On-Demand Documentation via Code Examples

Mathieu Nassif

School of Computer Science
McGill University, Montreal, Canada

August 12, 2024

A thesis submitted to McGill University in partial fulfillment of
the requirements of the degree of Doctor of Philosophy

Contents

Abstract

Software products have become increasingly reliant on smaller software components produced by third-party organizations to accomplish specialized tasks. In this context, documentation is a crucial component of software development to mediate the knowledge exchange between developers who create reusable software components and those who rely on the components for their projects. Creating and maintaining good documentation, however, is an effort-intensive activity that is often overlooked in favor of working on the source code. Prior work has addressed this issue by proposing various methods to generate or to retrieve relevant information from a documentation set based on a query. These techniques focus on improving the quantity, accuracy, or relevance of the documentation content.

In this thesis, we approach the multi-faceted challenge of documentation quality through another perspective: the interaction between readers and the documents. Improving ways to interact with documentation should help readers with varied backgrounds and subjective preferences apply their own optimal strategies to navigate to the information they need. Thus, considering the interaction between readers and documents as an essential aspect of documentation design should contribute to an efficient knowledge exchange between developers.

As a first contribution, we propose to automatically identify concepts that developers should be familiar with prior to reading a software component's documentation. This investigation led to a better understanding of techniques to capture and represent concepts with the use of knowledge bases such as Wikipedia. Our results benefit component users, who may find gaps in their knowledge to address before trying to read complex documents. Our results also benefit developers who want to review their software component or its documentation to minimize the number of conceptual dependencies.

As a second contribution, we describe variations points observed in the design of high-quality Android tutorials. We compared the content and organization of three tutorials to identify design decisions that writers must take based on their judgment. As we compared

only tutorials from reputable organizations and targeted at a similar audience, the identified variation points suggest aspects of documentation for which the impact of alternative decisions is not well understood. Thus, our results can help practitioners think more critically and systematically about various documentation design decisions. For researchers, our results highlight the need for further studies about specific documentation aspects.

As a third contribution, we designed a new format, called Casdoc, to present online documentation. Casdoc improves upon the static presentations influenced by the historical constraints of printed resources. Readers interact with documents to dynamically reveal and hide information based on their needs. This interactivity allows documents to contain more information without appearing bloated. In addition to creating a viable alternative format for learning resources, we also used Casdoc to study how developers interact with documentation to guide the development of further formats. The results demonstrate that our participants could leverage the benefits of Casdoc to mitigate recurrent limitations of static documents. These findings, together with the documentation design guidelines we elicited, motivate a wider exploration of alternative formats to complement content-focused research on documentation quality.

Résumé

Les produits logiciels dépendent de plus en plus de composants tiers pour accomplir des tâches spécialisées. Dans ce context, la documentation est un élément crucial du développement logiciel, afin de soutenir l'échange de connaissances entre les développeurs qui créent des composants logiciels réutilisables et les développeurs qui dépendent de ces composants dans leurs projets. Cependant, créer de la bonne documentation est une activité laborieuse, souvent ignorée en faveur du développement de code. La recherche a abordé ce problème en proposant diverses méthodes pour générer ou extraire des information pertinentes à partir d'un ensemble de documents. Ces techniques mettent l'accent sur l'amélioration de la quantité, de la précision, et de la pertinence du contenu de la documentation.

Dans cette thèse, nous approchons le défi de la qualité de la documentation selon une autre perspective : l'interaction entre les lecteurs et les documents. Améliorer les façons d'interagir avec la documentation devrait aider les lecteurs avec des expertises variées et des préférences subjectives à appliquer leur propre stratégies optimales pour localiser l'information dont ils ont besoin. Par conséquent, considérer l'interaction entre les lecteurs et les documents comme un aspect essentiel de la conception de document devrait contribuer à un échange efficace de connaissances.

Comme première contribution, nous proposons d'identifier automatiquement les concepts avec lesquels les développeurs devraient être familiars avant de lire la documentation d'un composant logiciel. Cette recherche nous a mené à mieux comprendre des techniques pour extraire et représenter des concepts à l'aide de bases de connaissances comme Wikipédia. Nos résultats bénéficient aux utilisateurs de ces composants, qui pourront déterminer des lacunes à leurs connaissances avant d'essayer de lire des document avancés. Nos résultats bénéficient également aux développeurs qui veulent réviser leur composant logiciel ou sa documentation pour minimiser le nombre de dépendances conceptuelles.

Comme seconde contribution, nous décrivons des points de variation observés dans la conception de tutoriels de qualité à propos d'Android. Nous avons comparé le contenu et

l'organisation de trois tutoriels pour identifier les décisions de conception que les auteurs de documentation doivent prendre en fonction de leur jugement. Comme nous avons comparé des tutoriels d'organisations réputées et qui s'addressent à un auditoire similaire, les points de variation identifiés suggèrent des aspects de la documentation pour lesquels l'impact de différentes décisions est flou. Ainsi, nos résultats peuvent aider les praticiens à réfléchir critiquement et systématiquement à diverses façons de concevoir la documentation. Pour les chercheurs, nos résultats illustrent le besoin d'études supplémentaires à propos d'aspects spécifiques de la documentation.

Comme troisième contribution, nous avons conçu un nouveau format, nommé Casdoc, pour présenter de la documentation en ligne. Casdoc diffère de présentations statiques dérivée des contraintes historiques due à l'impression des documents. Avec Cadsoc, les lecteurs interagissent avec des documents pour révéler et dissimuler dynamiquement l'information dont ils ont besoin. Cette interactivité permet aux documents de contenir plus d'information sans devenir trop volumineux. En plus de créer un format alternative viable pour des ressources d'apprentissage, nous avons aussi utilisé Casdoc pour étudier comment les développeurs interagissent avec des documents pour aider le développement de formats futurs. Nos résultats démontrent que les participants ont pu profiter des avantages de Casdoc pour atténuer les faiblesses de documents statiques. Ces résultats, avec les suggestions de conception concrètes que nous avons trouvé, motivent l'exploration de formats alternatifs pour complémenter la recherche sur la qualité de la documentation orientée sur le contenu.

Contributions

This thesis contribute to the field of software documentation. It is organized around three research projects, described in Chapters 3, 4, and 5.

Chapter 3 describes the investigation of conceptual dependencies, i.e., concepts related to the development and usage of a software system. In this project, we studied different techniques to extract, represent, and manipulate conceptual knowledge by associating software documentation with entries from a knowledge base. This project contributes:

- results about the performance of six wikifiers applied to software-related documents that can help researchers and practitioners select an appropriate wikifier and configuration parameters for specific applications;
- detailed insights about heuristics to improve the performance of wikifiers for software-related documents;
- a discussion of the combination of different community search techniques to estimate the smallest maximally-connected community within a knowledge graph;
- a novel approach, Scode, to automatically identify concepts that developers must be familiar with when contributing to a project, using a consistent and project-agnostic terminology to express concepts;
- the results of an extensive empirical evaluation of Scode, including its end-to-end performance and the individual performance of its components;
- a manually-curated list of over 6700 computing-related Wikipedia articles about software development.

The contributions of this project were published in two articles: *"Wikifying Software Artifacts"* (Empirical Software Engineering) [156] and *"Identifying Concepts in Software Projects"* (IEEE Transactions on Software Engineering) [159]. The author of this thesis was

the principal investigator of this project. He designed Scode with advice from his supervisor. He implemented Scode and the scripts necessary to collect and analyze data for the studies, designed jointly with his supervisor. The data annotation step was split between the author and his supervisor. The author wrote the original drafts of the articles, which were then reviewed and edited by both the author and his supervisor.

Chapter 4 describes the investigation of design variations found in high-quality documentation. This smaller project bridges the other two research projects that constitute this thesis. In this project, we compared the content and organization of three Android tutorials to identify notable design decisions that can affect how readers find information in the tutorials. This project contributes:

- a series of documentation design decisions, organized by variation points related either to the content or the organization of the tutorial, that writers can use to critically reflect on the design of their documents;
- guidelines that describe the rationale and trade-offs of each design decision;
- data-driven examples to illustrate and contextualize each design decision.

The contributions of this project were published in one article: *"A Data-Centric Study of Software Tutorial Design"* (IEEE Software) [15]. The author worked jointly on this project with another PhD student, Deeksha Arya. Both contributed equally, under the supervision of their supervisor. The author and D. Arya designed the study with advice from their supervisor, implemented the scripts to collect and analyze the study data, and discussed preliminary findings to iteratively expand and refine the data collection procedure. The author and D. Arya both contributed to the original draft of the article, which was then reviewed and edited by them and their supervisor.

Chapter 5 describes the investigation of design strategies to create interactive documentation. In this project, we designed and implemented a new format of documentation. We then used this format in a field study and a laboratory study to evaluate the impact of interactive design features on the documents' readers. This project contributes:

- the design of an innovative format, Casdoc, that embeds interactive annotations in code examples;
- an online service that transforms annotated Java files into self-contained HTML files that use the Casdoc format, to facilitate the creation of Casdoc documents;

- a discussion of five documentation design aspects of Casdoc that were synthesized from prior work on information needs and documentation usage behavior;
- a complete methodology for the design of a field study that maximizes the ecological validity and reliability of the results in a context where the investigators have authority over participants;
- a set of short programming tasks that elicit meaningful information needs, with the material necessary to use these tasks in a controlled experiment on documentation usage or programming behavior;
- guidelines for the design of interactive software documents, synthesized from the results of our field and laboratory studies.

The contributions of this project are described in four articles. The initial implementation of Casdoc was published in an article titled *"Casdoc: Unobtrusive Explanations in Code Examples"* (International Conference on Program Comprehension, demonstration track) [154]. Preliminary results of the field study were published in an article titled *"A Field Study of Developer Documentation Format"* (CHI Conference on Human Factors in Computing Systems, late-breaking work track) [158]. A comprehensive report on the design of Casdoc and the results of the field study is presented in an article titled *"Non Linear Software Documentation with Interactive Code Examples"* (ACM Transactions on Software Engineering and Methodology, accepted pending minor revisions, link to the ArXiv preprint in the reference) [160]. The design and results of the laboratory study are presented in an article titled *"Evaluating Interactive Documentation for Programmers"* (Empirical Software Engineering, under review) [162].

The development of Casdoc was led by the author. His supervisor provided insights about several design aspects and piloted several versions of Casdoc. Two undergraduate research assistants, Zara Horlacher and Emily Shannon, contributed to the implementation of Casdoc under the direction of the author and his supervisor (up to and including the version described in Section 5.7). The field and laboratory studies were designed jointly by the author and his supervisor. The author prepared the material for the laboratory study, conducted the study sessions, and analyzed the collected data. The creation of the Casdoc documents for the field study was split between the author and his supervisor. The author implemented the infrastructure to deliver documents and collected interaction data for the field study, and analyzed the collected data. For all articles, the author wrote the original draft, and both the author and his supervisor edited and reviewed successive versions.

Acknowledgements

Doing a PhD is hard. Doing my PhD was made many times easier, and more rewarding, thanks to the many people that I have had the chance to interact with throughout my studies.

First and foremost, I am eternally grateful to my supervisor, Martin, for his unwavering commitment to my studies, professional development, and love of research. From the first day I joined his group as a research assistant, he has been an extraordinary mentor, sharing his enthusiasm and care for meticulous research. I thank him for his encouragements to develop my areas of expertise, his patience through many cycles of slow progress, his useful insight into all aspects of the academic life, the time he devoted to our weekly meetings (and the walks on the Mount Royal), and the right dose of cynicism at the right moments.

I thank Christoph, who contributed to my initiation to research all these years ago. I am privileged to have worked under his co-supervision when he was a postdoc at McGill. He helped me grow from writing a few lines of code to query the Stack Overflow API, to writing my first research article. I also had the chance to work alongside Jin, who co-leads our lab. Despite not having her as a co-supervisor, I benefited from her devotion to creating and maintaining a great research environment. I also benefited from her ability to reveal deep discussion points about my research that eluded me for months when I seek feedback for an article or a presentation.

I am grateful for the exceptional environment we had in the Software Technology Lab (and its predecessors). I enjoyed the interesting discussions and social events with Alexa, Ziming, Alex, Cheryl, Kian, Fuyuan, Breandan, Justine, and all other current and past lab members. A special thank you goes to the current lab regulars: Deeksha, for being the best desk neighbor and sharing the high and (ridiculously) low points of academia; Bhagya, for the company on the way home; Sara, for the company during late nights in the lab; Lanese, for your generous food offers and great cooking skills; Linh, for keeping me humble about Quebec's culture; Jazlyn, for your insightful perspective on society, religion, politics, etc.; Avinash, for always checking up how it's going; and Divya, for accepting to go outside of your

comfort zone when I make suggestions. I thank you all for your strong support and for going out of your way to make the lab a lively and stimulating place to work.

I also had the chance to work with many smart and motivated research assistants who did a project in our lab. I thank Ashvitha and Alexa for their contribution to my master's work, DScribe; Zara and Emily for their work on the version of Casdoc presented in this thesis; Tristan and Vivian for their contribution to the future of Casdoc; and Muhammad for his diligent help with the collection of test convention data. You all provided invaluable help to achieve my research objectives.

I was privileged to meet and work with Walid during a research internship. I thank him for being a great host supervisor, for his concrete and useful advice on research, especially to reach the end of my PhD, and for sharing his enthusiasm about research. I also thank Lloyd, Abir, Tim, Volodymyr, Clara, and the other members of the MAST group for their warm welcome and for the lively exchanges of research ideas and perspectives.

I am grateful for the structures in the School of Computer Science to help PhD students. I thank the CSGS, which conveniently included many members of our lab, for promoting a sense of community among CS graduate students. I also thank Corey, Ron, and Andrew for diligently helping with any software or material issue, as well as Sheryl, Tricia, Ann, Diti, Kamini, and other staff members in the administrative office for their work supporting the entire department. Additionally, I thank the PhD Program Committee and my Progress Committee, Comprehensive Exam Committee, Proposal Committee, Defense Committee, and Thesis Examiners for their involvement in the evolution of my PhD. Beyond McGill, I had the chance to be supported by NSERC, FRQNT, and Mitacs.

On a more personal note, I am grateful to Max and Deeksha for their deep friendship throughout my PhD, supporting me in the difficult moments and celebrating with me the victories. I thank Jessie for humoring me during the morning climbing sessions, and for her help with the next stages of my career. I also thank Ariane, for making many hard periods easier, and some easy periods harder, because life without challenge would be boring.

Dernièrement, je remercie ma famille, qui m'a toujours encouragé et soutenu dans mes ambitions de recherche. J'ai une pensée particulière pour Louis, qui m'encourage à ne jamais arrêter. *"1, 2, 3, Go!"*

To all these people, I express my sincere gratitude for their contribution, direct or indirect, to my PhD.

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
C2W	Concepts to Wikipedia
Casdoc	Cascading documentation
CSS	Cascading Style Sheets
CVE	Common Vulnerabilities and Exposures
ESA	Explicit Semantic Analysis
HTML	HyperText Markup Language
JDBC	Java DataBase Connectivity
JSON	JavaScript Object Notation
LDA	Latent Dirichlet Allocation
Scode	Software conceptual dependency
SQL	Structured Query Language
UPGMA	Unweighted Pair Group Method with Arithmetic mean
URL	Uniform Resource Locator

Chapter 1

Introduction

Software documentation is a crucial aspect of software development. As projects become more complex, developers increasingly rely on components created by third-party organizations to accomplish common operations. For example, as of March 2024, Maven Central hosts over 13 million packages for Java projects, and PyPI hosts over half a million projects for Python. Developers use such software components through their application programming interface (API). Because the developers who *use* an API may not have access to the developers who *created* it, knowledge about the API must be transferred asynchronously, through documentation.

Creating good software documentation that is effective in facilitating this knowledge transfer is challenging. Many documents act only as passive containers of information. Under this assumption, a high-quality document is expected to contain exactly the information sought by its readers. Thus, past research on software documentation studied techniques to generate more information [35, 121, 135], or to retrieve it from various sources [56, 233]. Yet, given that readers from various background will use documents to accomplish various tasks, it is unsurprising that documentation still suffers from many issues, such as being too verbose, too technical, or lacking relevant background information [5, 216].

With the transition to online documentation, where a plethora of learning resources are accessible through search engines, the purpose of documents expands beyond being information containers. Documents should encourage active reading and information seeking behavior [221]. Thus, good documents should help readers navigate between document fragments, adapting to the evolving needs and preferences of individual readers as they learn about different topics.

This thesis explores how the design of documentation can capture the role of readers as more than passive consumers in the exchange of knowledge. This problem is challenging, as a departure from traditional documentation design strategies can increase both the creation cost for writers and the adoption cost for readers. Nevertheless, exploring new documentation design strategies can also reveal opportunities to create documents that better match how developers look for information [30, 208]. For example, allowing readers to select their preferred reading order and set of concept definitions can make the difference between a document that provides a *complete* description of a topic and one that appears overly *verbose*.

As documentation can take many forms, we scoped our research to online documents to learn about an unfamiliar software API, such as a tutorial that explains how to connect to a database using Java's JDBC API. More specifically, we restricted our studies to documentation distributed as text-based web pages, as opposed to, e.g., documentation delivered inside code editors and non-textual documents (e.g., video tutorials). We also excluded internal developer documentation (e.g., code comments, tickets from issue tracking system) and learning resources independent of specific technologies (e.g., basic programming courses). Aside from tutorials, our scope includes API reference documentation and programming forums, which are also online text-based resources used by programmers to learn how to use a new API.

We approached our investigation of documentation design from three angles. First, we studied the concepts that developers must know when using an API, and consequently when reading its documentation. Documentation often describe *technical* usage information, such as directives [33, 116] or usage scenarios [35, 185]. However, developers must also be familiar with relevant *concepts* to correctly navigate and understand the documentation. Thus, we refer to such concepts as *conceptual dependencies*. We studied techniques to identify conceptual dependencies, which can help documentation writers and readers, as well as researchers, reason about the match between the content of documents and the background of individual readers.

Although conceptual dependencies are useful to critically assess *what* information is or is not included in documentation, they provide little insight into *how* to structure the information in a document. Thus, in a second phase, we studied design variations in tutorial documentation. We observed that there is a lack of guidance about many decisions that occur when designing tutorials. For example, there are no widely accepted recommendations about the amount of content to include, the selection of topics to cover, or strategies to integrate code examples within the prose of a tutorial. This gap in design knowledge prevents researchers and tutorial writers from systematically reasoning about how readers can use

the documents. We compared three high-quality tutorials, all about the same domain and targeted at the same audience, to elicit design decisions that serve as a basis towards a standardized design framework for tutorials.

Finally, we explored in more depth the design space for a specific type of documents: interactive code examples used as API learning resources. Despite the variations in documentation design, most documents are delivered as (mostly) immutable web pages. Readers can only interact with these documents through standard HTML features, such as hyperlinks and the browser's native text search tool. This limited design overlooks the potential of digital documents to adapt to the needs of different developers [190]. Thus, we designed and implemented a new interactive format for code examples that embeds a hierarchy of explanations as annotations in the code. Readers access annotations on-demand by interacting with subtle markers that indicate the presence of information. In addition to the intuitiveness of the user interface for readers, we considered the authoring effort for writers as an important factor when designing the format. When then evaluated our new format through two user studies to understand how developers can benefit or be limited by interactive features in software documentation.

As a whole, this thesis demonstrates the potential of revisiting the design of documentation to address common limitations. We focus on intentional decisions that pertain to the design of entire documents, as opposed to the quality of a document's content. In particular, we consider the selection of a cohesive set of topics to cover as part of the design, but not decisions related to how individual topics are described. Thus, the recent developments in generative artificial intelligence technologies (e.g., ChatGPT) do not solve the problems studied in this thesis. Nevertheless, we discuss the impact of these technologies on documentation design in Section 6.3.

We contribute detailed empirical insights into techniques to capture the conceptual context of documents, and into current variation points of documentation that are not well understood. We also contribute design insights into, and the implementation of, a novel format of interactive documentation, which can serve as a starting point to investigate a currently unexplored portion of the documentation design space. The results of our user studies contribute to building an understanding of the strengths and limitations of interactive documents, thus further supporting the exploration of this design space. Thus, with this thesis, we hope to encourage documentation creators and researchers to investigate alternative design strategies to improve documentation quality.

1.1 Thesis Organization

The remainder of this chapter introduces each part of this thesis: identifying conceptual dependencies (Section 1.2), exploring design variations in tutorials (Section 1.3), and designing an interactive documentation format (Section 1.4). In each section, we present the research *goal* and its *motivation*. We then present an overview of the *research design*, and a *summary of the findings*. As each section is intended as a brief overview of the research, we relegate details about the research questions and contributions to later chapters, where they are most relevant. We also synthesize the research questions and contributions at the end of the thesis.

Chapter 2 presents a discussion of related work. We discuss the general context that motivates this thesis as a whole, as well as the work specifically related to each part.

Chapter 3 presents our investigation of wikification and community search techniques to identify conceptual dependencies. We start with a precise definition and motivation of the problem we address (Section 3.1). We then present the protocol we used to compare state-of-the-art wikifiers (Section 3.2) and the results of this comparison (Section 3.3). These sections are followed by a complete description of Scode, our approach to identify conceptual dependencies (Section 3.4). The last section of the chapter presents the design and results of our multi-faceted evaluation of Scode (Section 3.5).

Chapter 4 presents our investigation of variation points in Android tutorials. This smaller research project bridges the work presented in the previous and next chapters. The chapter starts with an overview of the design decisions and variation points we found (Section 4.1). We then elaborate on the methodology and evidence gathered to elicit decisions about the organization (Section 4.2) and the content (Section 4.3) of tutorials. The chapter concludes with a discussion of the results and their significance to promote further exploration of tutorial design alternatives (Section 4.4).

Chapter 5 presents our investigation of interactive presentation formats for API learning resources. The chapter starts with a description of our novel format named Casdoc (Section 5.1), followed by the key properties of the format that address limitations of traditional formats found in prior work (Section 5.2). The next two sections present the design (Section 5.3) and results (Section 5.4) of the field study. Similarly, the following two sections present the design (Section 5.5) and results (Section 5.6) of the laboratory study. At the end of the chapter, we present the improvements we made to Casdoc based on the results of our studies (Section 5.7).

Finally, Chapter 6 presents a discussion of our findings, and Chapter 7 concludes this thesis.

The data collected during the studies presented in this thesis, as well as the material necessary to replicate the studies, are available online [14, 155, 157, 161]. Appendices A, B, and C present the detailed content of each data artifact, for each part of this thesis. Relevant files from the data artifacts, such as the data annotation guides, are replicated in the appendices.

1.2 Identifying Conceptual Dependencies

We investigated the concepts that a developer must be familiar with when using a technology—and therefore when reading its documentation. For example, the documentation of a library to hash passwords is likely to mention concepts such as *cryptographic salt*, *hash function*, and *cipher*. Developers should be familiar with those concepts, otherwise they may not fully understand the documentation and misuse the library. We refer to these concepts as *conceptual dependencies*.

1.2.1 Goal

The goal of this part of the research was to study techniques to automatically identify conceptual dependencies of a software project, given its documentation as input. Generating a list of relevant concepts, however, is a challenging task due to the intangible nature of concepts and the often subjective and implicit understanding of prerequisite knowledge.

1.2.2 Motivation

Prior studies have found that developers without a sufficient background find it challenging to make effective queries and judge the relevance of the documents they find when looking for documentation [109], thus making the lack of background a common obstacle to learning a new API [187]. To illustrate the importance of conceptual dependencies, we consider the scenario of a novice developer reading a tutorial about Android development. As the developer learns about data storage APIs in Android, they stumble over unfamiliar concepts, such as *SQLite*, *DAO*, and *data entity*. For each term, the developer can guess its meaning from the context or ignore the corresponding passage entirely, in both cases risking to miss relevant information. Alternatively, the developer can stop reading the current tutorial and look elsewhere for a description of these concepts, which may not be as trivial as reading

the Wikipedia page for the target concept [191]. Doing so too often will thus disrupt the progression of the tutorial.

If, instead, the developer had access to a list of conceptual dependencies of the Android data storage APIs, they could address gaps in their knowledge before reading the tutorial and avoid those disruptions. Furthermore, such a list could be organized by domain (e.g., grouping *SQLite*, *DAO*, and *data entity* under the *database* domain). This structure could support a more natural introduction to related concepts than learning about each concept individually as they come up in the tutorial.

Such an explicit list of the conceptual dependencies could also allow writers to critically review the concepts introduced in a document. Writers could ensure that new concepts are defined in a prominent place, and insert links to other relevant documents that define concepts assumed as prerequisite for the current document.

The development of *concept linking* and *community search* techniques from the computational linguistics and network science domains, respectively, provides the building blocks to identify relevant concepts. The computational linguistics domain defines the *concept linking* task as identifying mentions of concepts in a natural language text and linking them to entries in a knowledge base [206, 217]. A common option is to use Wikipedia as the knowledge base, using articles as the knowledge base entries. In this case, the concept linking task is also called *wikification*, and automated tools to perform it are called *wikifiers*. In network science, the *community search* problem consists of finding, given a graph and initial query nodes, a densely-connected subgraph that contains all query nodes. Finding optimal solutions is particularly difficult on large graphs, as the search process becomes computationally expensive. However, researchers have proposed different techniques to approximate optimal solutions within practical computational limits [68].

We combined wikification and community search techniques to approach the problem of identifying conceptual dependencies. However, these techniques have been developed, and evaluated, for general-purpose domains, e.g., identifying concepts in news articles, or friend groups in social networks. Thus, it is unclear how well they can adapt to the idiosyncrasies of software-specific documents. Furthermore, several pre-trained wikifiers are available, each with different configuration parameters that affect their precision and recall.

1.2.3 Research Design

We started our investigation by comparing six state-of-the-art wikifiers, with multiple configurations for each wikifier, on a sample of software documents. We labelled each concept

identified by the wikifiers as "correct" (i.e., true positive) or "incorrect" (i.e., false positive). We considered the set of concepts missed by a wikifier but identified by any other wikifier, and labelled as "correct", as the set of false negatives. This data allowed us to compute precision and recall values to compare all wikifiers and configurations.

We then implemented state-of-the-art community search algorithms to expand the wikification results. The resulting tool, named *Scode*, takes as input the documentation of a software project and outputs a list of related concepts, organized by topic.

We evaluated the end-to-end performance of Scode by applying it to a sample of Java classes and manually validating its output. We compared its precision and consistency to two other concept identification techniques.

Additionally, we assessed the impact individual design aspects of Scode, including the cohesiveness of concept sets identified by the community search algorithms, the benefits of using an inclusion list to filter the wikifier's results, and the impact of using a project's documentation instead of its source code.

1.2.4 Summary of Findings

The comparison of wikifiers confirm the complexity of choosing an optimal wikifier and its configuration for specific applications. Each wikifier outperformed the others for different ranges of recall values. We observed that precision and recall was lower for software documents than the values reported in the original articles for evaluations with general-purpose documents. However, an analysis error patterns elicited some heuristics to improve their performance. For example, matching the output of wikifiers against an inclusion list of concepts helps to filter out false positives from common polysemous terms.

The evaluation of Scode revealed that it can consistently retrieve many concepts, but the unfiltered results include many false positives. However, most of these false positives are due to large communities surrounding general concepts, such as JSON. Removing large communities leads to practical configurations of Scode with precision values that range from 25% to 66%. By grouping concepts into a practical number of topics (i.e., communities), Scode can provide an overview of the conceptual dependencies of a project.

In addition to the design and evaluation of Scode, our results provide detailed insights into the potential and pitfalls of two techniques (i.e., wikification and community search) to capture the conceptual context of projects.

1.3 Exploring Design Variations in Tutorials

Due to the lack of empirically-validated guidance to evaluate and increase the quality of documentation, technical writers must rely on their judgment to design effective documents. As a result, the design of documentation from different organizations vary considerably. Understanding the decision points in the design of a document, both related to the selection of its content and its organization, is a necessary step to move towards a more systematic evaluation of quality aspects.

1.3.1 Goal

The goal of this part was to elicit variation points in the design of software tutorial documentation. We studied tutorials created by professional writers from reputable organizations to focus on high-quality documents.

1.3.2 Motivation

There are many approaches to design documentation. When creating a new tutorial, writers must take many decisions about what content to include and how to organize different parts within and across documents. Despite past research on documentation design (e.g., [29, 138]) and the desires and constraints of developers (e.g., [57, 141]), many of these design decisions are not well documented. Thus, it is difficult to find reliable guidance about the impact of different decisions to improve tutorials.

Some design decisions may be the consequence of practical constraints. For example, writing several blog posts that eventually culminate in a tutorial can ease the development of the tutorial by breaking it into smaller, self-contained documents. Blog posts can also focus on popular aspects of a technology to generate a faster return on investment and sustain the development of posts about less popular aspects. This strategy may be desirable for organizations who cannot commit to the larger effort investment required to plan a single, comprehensive tutorial. However, other design decisions may not be easily linked to development constraints. For example, how to interleave prose and code examples may not affect as much the effort required to create a tutorial, yet it can impact how readers use the tutorial. Thus, such decisions rely on the subjective judgment of expert writers.

Progressing towards a more systematic approach to tutorial design requires first to identify these variation points. This initial step is necessary to support a rigorous empirical assessment of the impact of different decisions on quality. An explicit list of variation points can help

writers consider more options when creating tutorials. Even without prescriptive guidelines, raising awareness of alternative designs encourages a critical analysis of (possibly unconscious) decisions.

1.3.3 Research Design

We conducted a case study of three high-quality tutorials on Android programming for novice developers. We chose Android as the target domain as it is a popular development framework with many available tutorials. Android development also requires to consider many factors, such as the variety of hardware to support and constraints such as battery usage and responsiveness. These considerations, which are typically less important in desktop applications, add variety to the topics that tutorial writers can choose to cover.

For each tutorial, we automatically extracted structural features based on HTML tags, such as the number of words, sections, code examples, and hyperlinks. We used Stack Overflow tags as a dictionary of topics to capture the coverage of the tutorials' content.

The data we collected highlighted key differences between the tutorials. We manually investigated these differences to elicit design decisions they linked to and understand possible causes and trade-offs of these decisions.

1.3.4 Summary of Findings

Our comparison of tutorials elicited eleven design decisions related to five variation points. The variety of design strategies motivates future work to improve and standardize the creation of tutorial documentation.

1.4 Designing an Interactive Documentation Format

We investigated the *presentation* of information in documentation. Although documentation writers must address a variety of information needs, including too much content in a document can overwhelm readers and hide or reduce the relative weight of the most important information about the API. Interactive presentation formats can alleviate the negative impact of verbose documents by showing readers only the information they need. However, to be effective, interactive formats must support intuitive navigation behavior from readers to justify the migration from the well-known static HTML pages. Additionally, they must not overly increase the authoring effort for writers to facilitate its adoption.

1.4.1 Goal

The goal of this part is to understand how interactive documents can support common navigation behavior reported in prior work, as well as to explore alternatives to structure information within a document. We pursue this goal through the iterative design and evaluation of a novel presentation format, called Casdoc.

1.4.2 Motivation

When reading tutorials, developers typically do not read every word. Instead, according to Information Foraging Theory [174], they will scan different parts of a document, looking for clues about the location of the information they need, and only read more thoroughly the parts of a document that they estimate is more likely to contain such information. Developers can even ignore the text of a tutorial entirely at first, starting directly at the code examples provided in the tutorial [30]. Hence, developers reading a tutorial need to perform a series of *navigation actions* (e.g., scrolling, scanning a paragraph, using a textual search tool) to reach their target information more effectively.

Documents can include different features and navigational cues [151] to support an effective navigation by its readers. For example, it can contain a table of content and hyperlinks to different parts of the document. However, those navigational aids can only improve to a certain extent the inherent navigability of its structure of information. As an extreme example, a large document presented as a long uninterrupted paragraph of natural language text is unlikely to be easy to navigate, even with many navigation features.

A more practical and common format consists of dividing the information into a sequence of ordered sections. Such a *linear* format retains the ordering of information to provide a clear starting point for readers, but breaks down the document into manageable fragments that can be scanned or skipped by a reader.Even when different sections are independent, they are presented in an explicit sequence that provides an inherent reading order. Other presentation strategies exist too. For example, the structure of Java API reference documentation (i.e., *Javadoc* documentation) matches the hierarchical structure of the API, with no fixed order across all API elements. However, non-linear formats are less common for learning resources such as tutorials.

The growing amount of documentation resources available online accentuates the importance of designing effective presentation formats, as developers will look elsewhere if it takes too much effort to locate the information they need in a document [174]. Thus, navigability issues constitute an important threat to the value of a document, as even if a document

contains a large amount of information. However, without empirically validated guidelines to structure documentation, it is challenging to detect and address these issues.

In particular, the digitalization of documentation created opportunities to design interactive features that printed documents could not support. Interactive documents can adapt to the needs of a diverse audience by showing tailored content to each reader. However, designing interactive interfaces is costly, and it can reduce navigability if the interface is not intuitive. Thus, eliciting guidelines to create interactive documents can help writers leverage the potential benefits of interactivity while limiting the threat of introducing navigability issues.

1.4.3 Research Design

We started with an exploration of the space of presentation strategies through the design and implementation of a novel format, Casdoc. The design of Casdoc was informed by prior work on common navigation patterns and information needs of programmers. We then conducted two studies to investigate the strengths and limitations of Casdoc to support programmers learning to use unfamiliar APIs.

We conducted a *field study* to collect quantitative evidence of the viability of Casdoc in a realistic context. The study took place during two sections of an undergraduate course on software design. Participants, i.e., students enrolled in the course, used Casdoc documents as part of the course material, alongside other material such as the course's textbook.

We instrumented the documents to asynchronously send interaction events when participants used any feature of the documents. The data collection procedure was minimally intrusive, both to preserve the ecological validity of the results and to avoid a potential negative impact on the students' learning objectives.

Throughout the field study, we collected over 18 000 interaction events from 326 participants. These events provided detailed insights into the usage of different features, highlighting aspects of the format that worked well and others that needed improvements.

We complemented the field study with a *laboratory study* to directly observe how (and why) programmers use different features of Casdoc when working on programming tasks. In this study, participants completed a set of tasks in a controlled environment. Each participant worked on the tasks during a single videoconference session with one of the investigators.

Participants were allowed to use any development technology they were used to, except for online documentation other than the documents we provided. The provided documents

used a combination of Casdoc and a traditional, non-interactive format, which allowed us to compare how well each format supported different navigation strategies.

We analyzed recordings of the sessions to elicit recurrent navigation patterns that participants used to find relevant information. We correlated these patterns to the usage of each format to compare the formats' strengths and limitations.

1.4.4 Summary of Findings

The field study showed the viability of Casdoc as a format for API learning resources. It confirmed the benefits of allowing readers to selectively reveal and hide content to mitigate the verbosity of documents. However, it also highlighted some aspects of interactive formats that must be carefully designed to avoid navigability issues. For example, aesthetic choices (e.g., the color of navigation hints) can impact the discoverability of information. Writers should also be careful to place important information in prominent places that do not require many user actions to locate. We synthesized these findings into five guidelines that can help writers improve the design of documentation formats.

The laboratory study revealed several contextual factors of an information search that make either an interactive or a non-interactive format more useful. For example, when trying to get familiar with the concepts of a new domain, participants typically preferred to use the non-interactive format, which allows them to scan surrounding paragraphs to estimate the document's coverage of the domain. In contrast, when trying to find a code example to use as an initial solution to perform a specific action, participants typically preferred Casdoc to look for short explanations about the details of the code example. Overall, the results showed that the combination both formats, even if it requires duplicating the document's content, is useful to support a large variety of search strategies.

Chapter 2

Background and Related Work

Researchers and practitioners agree that documenting software is both important and challenging. The lack of high-quality documentation is a common obstacle to learning the usage of new APIs [187, 189]. Yet, documentation is often missing, outdated, or untrustworthy [115, 141]. Even when up-to-date documentation is available, it can suffer from many other issues related to, e.g., its readability, correctness, or elements such as code examples and diagrams [5].

Recurrent issues with software documentation motivated the development of many techniques to reduce the burden of creating and maintaining documents (e.g., [36, 38, 84, 94, 243]), as well as many studies to understand what information needs to be documented and how best to document it (e.g., [54, 142, 148, 204]). This thesis contributes to this research effort on software documentation. As increasing documentation quality is a multi-faceted challenge, we focused on three aspects in our work: capturing and representing the intangible knowledge shared through documentation, studying the documentation needs of developers, and exploring different techniques and their impact to present information to developers.

Despite our focus on a specific type of documentation (i.e., API learning resources such as tutorials), we discuss prior work related to any relevant type of documents, as there are many cross-cutting challenges and concerns to improving documentation design. We present further related work specific to each contribution of this thesis in their respective chapter.

2.1 Capturing Concepts Relevant to Software Projects

Extracting the concepts that are related to a software system is useful to support many development tasks. In addition to using concepts for generating documentation [121], researchers have used different representations of knowledge to aggregate similar bug reports [171] and

feature requests [169], recommend third-party libraries [45], compare API types and methods [122], and categorize the technology landscape discussed on Stack Overflow [163]. Even closer to our work, prior work on requirements engineering proposed to use Wikipedia articles and categories to resolve language ambiguities [65, 72]. Researchers have also studied the problem of *concept location*, i.e., identifying locations in the source code of a software system that relate to a query concept, for example to help developers resolve defects [60]. In particular, Latent Dirichlet Allocation (LDA) is an unsupervised technique to map documents to a vector space based on latent topics, which has been extensively used in software engineering research to represent the (implicit) semantic information about many software artifacts, from source files to bug reports. For example, prior work has shown that LDA can help prioritize test cases [212] and find permission-related security issues in Android applications [59].

The variety of development tools that leverage semantic information about a software system demonstrates the potential that the identification of concepts can have on improving development methodologies. Thus, although this thesis focuses on conceptual dependencies for documentation, our results can also benefit other software engineering activities. In this section, we discuss prior work that address the extraction, representation, and manipulation of conceptual knowledge.

2.1.1 Concept Linking

In computational linguistics, research on the extraction of terms from an input document evolved from two related sub-tasks: *named entity recognition* (NER) and *named entity disambiguation* (NED). As the name implies, the two tasks focus on named entities, e.g., names of people, places, or organizations. Expanding this definition to the identification and disambiguation of any concept defines the *concept linking* problem. This task requires a knowledge base, whose entries constitute the dictionary of concepts that can be identified. Different knowledge bases can be used, such as WordNet [179], DBpedia [114], and YAGO [184]. When Wikipedia is used as the knowledge base, concept linking is also referred to as *wikification*.

A large number of different techniques exist to solve any of these problems [196, 236]. The popularity of concept linking is in part due to several challenges and shared tasks developed by the community, such as those of the Message Understanding Conferences (e.g., [205]), Senseval/SemEval (e.g., [143, 164]), and CoNLL (e.g., [213]). In particular, the popularity and extensive coverage of Wikipedia has motivated the development of many wikification techniques [206]. Many wikification tools (i.e., *wikifiers*) are publicly available, with the

parameters of their underlying models pre-trained on general-domain document corpora (e.g., [31, 149]).

The large amount of concept or entity linking techniques led to a community effort to generate a standard terminology to discuss the variants of the concept linking task [217]. The outcome of this community effort also includes a framework, GERBIL, to evaluate concept linking techniques with a consistent methodology and metrics. However, the methodology requires a gold standard against which to evaluate the results of the evaluated techniques. This requirement increases the cost of evaluating techniques on new domains, as we did as part of this thesis.

Despite the maturity of the research on concept linking, few software engineering researchers have investigated their potential to support development tasks. The lack of excitement for wikification techniques, relatively to other knowledge modeling techniques such as LDA, is possibly due to the unpredictability of wikifiers on software documentation. Software documents use a domain-specific terminology (e.g., without context, the term *tree* more likely refers to a data structure than a type of plant) and can include code terms (e.g., API method and type names). Both aspects are challenging to handle for natural language tools. Researchers have proposed named entity recognition techniques specific to the software domain to address these limitations [240, 241]. A particular instance of this problem consists of identifying code elements within natural language text [127, 186]. These techniques can identify software-specific entities, such as names of programming languages, of libraries, or of API types. Our work complements this effort by studying the problem of identifying software-specific concepts, such as common data structures or design patterns.

2.1.2 Knowledge Graphs

To address the need for knowledge-aware software engineering techniques, researchers have studied the construction and usage of knowledge graphs. Knowledge graphs can either be used directly by developers when searching for information about a software system [172], or they can support the design of other development tools (e.g., to support bug fixing [229, 246]). The value of constructing accurate and extensive knowledge graphs is not unique to software engineering [102]. Notable large knowledge graphs include Wikidata [226], DBpedia [114], and BabelNet [165]. However, because these knowledge graphs are not specific to software development, it is not trivial to adapt them for software engineering applications.

Researchers have proposed techniques to generate software-specific knowledge graphs using association rule mining [44] or heuristics based on grammatical dependencies identified

by natural language parsers [245]. Another recurrent strategy for to generate code-specific knowledge graphs is to extract relevant terms from source code identifiers and generate a semantic structure based on further information in source code [67, 183]. However, these techniques tend to generate an impractical number of concepts for large software systems. This limitation can be mitigated by filtering out irrelevant terms and clustering similar concepts in the output [1, 103, 239]. Related to knowledge graph construction is the problem of automated glossary construction, which attempts to identify relevant terms specific to a project [9, 227]. Glossaries provide a basis for developers to precisely express and discuss requirements and solutions of the software project. However, they are typically only relevant within the context of a single project. Furthermore, techniques that generate extract knowledge from software project artifacts tend to omit high-level concepts, such as relevant algorithms or design patterns used implicitly in the project.

Our investigation of conceptual dependencies can help extend existing knowledge graphs by linking software projects and their API to a project-independent source of knowledge, i.e., Wikipedia.

2.1.3 Community Search

Although large knowledge graphs are desirable to capture the semantic context of projects, they can be hard to navigate and manipulate in specific applications. In particular, the graph of Wikipedia articles is densely connected, but the meaning of a link between two articles is not precisely defined, other than a relatedness relation. As a result, two seemingly unrelated articles may be connected by only a few intermediate articles. For example, as of March 2024, the article Mozilla Thunderbird is connected to Julius Caesar though a single common neighbor (French Armed Forces).

Nevertheless, it is possible to derive meaningful information from the large-scale structure of the graph. In this thesis, we use *community search* algorithms to distinguish closely related sets of articles from spurious relations. The community search task consists of identifying, within a graph, a densely connected subgraph that surrounds a query node [68, 199]. As the identified communities contain many links between related articles, by definition, they are useful to exclude outlier relations, such as the link between an email client and a Roman dictator.[1] We discuss the technical details of the algorithms we used in Section 3.4.3, in the context of our proposed approach for identifying conceptual dependencies.

[1]We also identify a software-specific subset of Wikipedia to filter out irrelevant articles.

Community search algorithms, and other techniques from the network science domain, can help extract more knowledge from large knowledge graphs. In particular, graphs that rely on knowledge inputs from large crowds of volunteers, or those that are constructed using flexible approaches to identify arbitrary relations, typically contain a lot of information, but with a proportional amount of noise. Thus, our work can encourage further exploration of graph manipulation techniques to distinguish meaningful patterns.

2.2 Studies of Documentation Needs and Usage Patterns

During an observational study, Maalej et al. found that developers prefer to look at source code or ask their peers over consulting documentation, but attributed this preference to recurrent documentation issues such as sparsity and a lack of trustworthiness [129]. Knowledge elements such as rationale, intended usage, and real usage scenarios were also often missing from documents.

To improve current practices, researchers have sought to better understand how developers use documentation. Insights into developers' perspective on documentation can guide the effort spent on creating, maintaining, and researching software documentation. Prior work has investigated this problem from different angles. We discuss information needs expressed by developers, common actions to find information in documents, and quantitative techniques to capture documentation quality.

2.2.1 Information Needs

Understanding what information developers look for is necessary to ensure documentation address relevant needs. Thus, different researchers have collected evidence to synthesize needs from various sources [27].

With over 24 million questions and 22 million users as of March 2024, the popular question and answer forum Stack Overflow constitutes a valuable source of insights into those information needs [200]. Researchers have applied topic modelling techniques to investigate discussion trends on Stack Overflow [7, 192]. Their results can be used to derive taxonomies and automated classification approaches to manage different types of information needs [24]. Recently, Liu et al. manually analyzed a sample of 266 Stack Overflow posts to define a framework that categorizes information needs, subtypes of information to address those needs,

and how different API elements relate to those needs [120]. This line of research is useful to systematically evaluate the benefits of documentation.

Other researchers directly observed developers in action to elicit questions that arose during development tasks [62, 197]. They identified typical questions that developers tend to ask about unfamiliar APIs. These question templates can serve as a fine-grained, empirically-validated framework to evaluate and improve the coverage of documents. Contrary to studies based on data from public forums, studies with direct contact with developers are sensitive to information needs that are already addressed well in current documents. Other studies involved interviews or surveys to gather insights from developers (e.g., [4]). For example, Head et al. combined an analysis of log data and interviews with document readers and writers to study not only information needs, but also constraints that limited documentation writers from addressing these needs [88]. For example, they found that some writers preferred to exclude non-functional information to keep documentation minimal and avoid misinterpretations from readers.

There have also been studies to categorize the content of existing documentation to complement research on information needs [10, 128]. For example, Cogo et al. compared data from Stack Overflow and another question and answer forum with the official Rust documentation to study how well the language's documentation addresses demonstrable needs of developers [51]. They found that the content of official documents corresponds well to topics that developers ask about in general, but they also identified misalignments that suggest weaknesses in the documentation.

2.2.2 Documentation Usage Patterns

Over 30 years ago, Curtis et al. studied different strategies for documenting the control flow of small programs [55]. They found that using a constrained language was typically more effective than natural language or ideograms, but the arrangement of the content (i.e., whether it is shown sequentially, hierarchically, or using branches) did not have a considerable impact. Since then, both documentation [218] and software systems have evolved greatly, yet few researchers reassessed whether their findings still hold in the modern software development landscape. A recent study of architectural documents [64] generated observations consistent with Curtis et al.'s findings, but took into consideration modern formatting guidelines [50].

Aside from the impact of different document styles, prior work includes studies of the strategies used by programmers to navigate documentation [108, 119, 191]. For example, researchers have studied the opportunistic use of resources to satisfy specific needs that arise

when working on programming tasks [30]. They found that developers can use documents to fulfill different purposes, from learning new technologies to recalling exact information found in known locations. Such studies motivate the need to create flexible documents that can address multiple purposes, and accommodate developers with different personalities and preferences [106, 110].

Studies on document readability conducted outside the context of software development can also help improve software documentation. Hornbæk and Frøkjær studied reading patterns of students with scientific articles presented in three formats: a classic "linear" format, a "fisheye" format that allows readers to expand and collapse parts of a document, and an "overview+detail" format that shows a smaller version of the document in a left pane as an overview of the content [95]. They reported several reading patterns, from participants who preferred to carefully read through a document only once to others who jumped back and forth between different sections. They also observed that participants took more time on average to answer questions about the documents, but wrote better essays about the documents and showed higher satisfaction. In contrast, the fisheye format led to faster completion times, but also to more incorrect answers. These nuanced results suggest that the quality of a format is multi-faceted and likely task-dependent.

Other researchers have investigated several other aspects, such as optimal, and personalized, font choices [37]. Miniukovich et al. collected and synthesized guidelines described in the literature for increasing the readability of web documents, with a focus on accommodating dyslexia [147]. This effort produced a set of 61 guidelines, ranging from navigation features (e.g., "Use a breadcrumb trail [...]") to visual choices (e.g., "Avoid using italics in the main body of the text"). They further refined these guidelines to a set of twelve core guidelines supported by empirical evidence of their impact on the readability from average and dyslexic readers. In particular, they found that the two groups benefit from different sets of guidelines, demonstrating that the quality of a format is also reader-dependent.

2.2.3 Measuring Reading Behavior

More fundamental work can help build theories to better understand, measure, and predict the quality of software documentation. For example, Sharafi et al. used neuroimaging to identify differences between reading prose and code in brain activation patterns [195]. They also found similarities between programming tasks, such as understanding data structure manipulations, to seemingly unrelated mental exercises such as 3D spatial rotations. In another vein, Hu et al. compared automated metrics used in documentation generation

evaluations (e.g., ROUGE, BLEU, METEOR) to human judgment along six quality factors (e.g., naturalness, understandability) [98]. Although they found some correlation between the metrics and human judgment, the automated metrics are not sufficient to completely capture the quality of a document. Abid et al. demonstrated the usefulness of another technique, eye-tracking, to study how developers read unfamiliar source code [2]. They were able to distinguish, for example, that developers focus more on method calls than method signatures. This line of research is crucial to support the collection of precise and fine-grained data in software documentation research.

2.3 Designing Documentation Presentation Formats

There has been prior effort to improve the design of documentation. For example, Codelets is an approach to create databases of reusable self-documenting code fragments, designed in the context of web development [166]. A Codelet combines both the template code to implement a specific programming task and documentation explaining the solution. It can also be interactive, allowing programmers to update the code fragment using a form. This approach to documentation has notable similarities with the format we propose, Casdoc: both focus on code examples as the main documentation resource and rely on embedded annotations to provide additional explanations. However, Codelets requires a specific software infrastructure for developers to manage a database of information fragments and integrate them in their code. In contrast, Casdoc documents only use common web technologies by design, so they can be viewed with any web browser.

Closer to the context of our work, different techniques have been proposed to improve online developer documentation. The ability to execute and modify code examples within a document, without having to set up a suitable local development environment, encourages readers to actively engage with the content of a document [144, 222]. Guo designed a visualization tool to help novices follow step-by-step executions of programs [83]. The web-based tool includes a visual representation of the program's memory to help programmers understand and reason about the operations in the code. Other techniques focus on facilitating the comparison of code examples from various sources. For example, color highlights, distribution charts, and interactive selections can help programmers compare code examples from alternative libraries [79] or synthesize large sets of code examples [238] to accomplish a programming task.

More innovative formats have been studied to improve how documentation is presented, either for developers or other groups of users. For example, many researchers focused on new

documentation media, in particular video tutorials, as they became popular alternatives to text-based documents [49, 131, 150, 219, 220]. However, in the context of this work, we focus on primarily text-based documentation.

2.3.1 Interactive Formats

Many interactive formats proposed by researchers recently focus on the exploration and visualization of machine learning models (e.g., [139, 242]). For example, Symphony is a technique to create interactive interfaces to document machine learning models [22]. It relies on reusable components that allow users, e.g., to modify and visualize the data sets and to analyze properties of the generated models directly within the document. During three case studies, users found Symphony useful to explore data sets and to teach machine learning design principles. Similarly, Feng et al. found that live interactions with a machine learning model can ease communication between different groups of stakeholders [69]. Allowing data experts to influence the search space of hyperparameters through interactive dashboards can also improve the performance of optimization algorithms in complex situations [91].

Researchers have also studied techniques to help readers interact with data presented in documents through tables (e.g., [107]) and figures (e.g., [90]). Techniques such as Chameleon [134] and Charagraph [133] can overlay supplemental interactive figures on static images or generate interactive graphs to visualize data described in non-interactive documents, respectively. Such interactive figures can encourage readers to interact with the data and decrease the effort required to answer queries [133]. Wang and Kim also proposed to generate concise facts about data presented in documents to make such documents more relatable to blind and low vision individuals [228]. In a formative study, they found that participants had different preferences about the types of facts they are interested in, but participants were enthusiastic about the generation of personalized facts.

Aside from data visualization, Dragicevic et al. discuss the creation of interactive scientific articles that embed multiple data analyses, to allow readers to explore alternatives to what the researchers initially present [61]. Head et al. studied the use of techniques, including interactive ones, to help readers understand mathematical equations in documents [89]. Both of these studies focus on the design of the new format. Among other findings, they provide guidelines for creating, or facilitating the creation of, improved document formats.

2.3.2 Paper-Inspired Interaction Features

A recurring theme in prior work has been the integration of physical paper's affordances into electronic documents, such as the ability to annotate documents [96], synchronize bookmarks between printed and digital documents [18], or use natural pen-inspired gestures [78]. Chen et al. discuss how different types of devices (e.g., PC, tablets, and e-readers) support various reading actions, before proposing a system composed of multiple custom devices comparable to e-readers and tablets [47]. Their evaluation study with twelve participants showed that such a multi-device system can mitigate some limitations of electronic documents, but introduces new interoperability challenges, such as a low practical limit to the number of devices. Hybrid systems, which combine interaction on physical paper and on electronic devices, can address some of these limitations. Han et al. reported on a literature review of those systems, and discuss strategies to design and implement such systems [85]. Such work helps generate new ideas to improve the design of interaction between readers and electronic documents.

Chapter 3

Conceptual Dependencies of Software Documentation

Software documentation, in particular API reference documentation, often focuses on technical aspects of an API. It typically contains information about which objects and methods are related to a task or concept, for example, or about the constraints that the arguments of a function must satisfy. Regardless of the quality of the content, however, this technical knowledge is not sufficient to completely understand and correctly use an API. Developers must also be familiar with concepts related to the domain of the API, such as computer science theories, algorithms, and supporting technologies. For example, developers working on an e-commerce application may have to learn concepts related to database management systems and secure transactions, even if the application relies on third-party libraries to implement these requirements. Without this prerequisite knowledge, a developer may not understand the documentation at all, or miss important implicit assumptions of the domain concepts.

We refer to a domain concept that a developer must know to properly use a library or contribute to a software project as a *conceptual dependency*. Given the breadth of programming concepts, software developers must constantly learn about various technical concepts [30, 191]. However, identifying the conceptual dependencies of a software project, beyond the most prominent concepts, is a difficult task. It requires to delineate the extent of the knowledge involved in the project, partition this abstract and nuanced knowledge into distinct concepts, and express the relevant concepts with recognizable labels. Nevertheless, a precise mapping of concepts to the software project and its components can help developers

perform activities such as identifying the source of a feature [25, 60, 178], the cause of a bug [113, 181], or a third-party library for a specific task [45].

In this chapter, we describe an investigation of heuristics to automate the identification of the conceptual dependencies of a software project, based on computational linguistics [196] and network science [68] techniques. This investigation culminated in the design of a novel solution, named *Scode*,[1] to automatically generate a set of candidate conceptual dependencies for a project. Scode relies on *wikification* techniques to link mentions of concepts in natural language text to relevant Wikipedia[2] articles [196] and on *community search* techniques to identify further related concepts [68]. The design and implementation of Scode required several adaptations of these techniques to the software development context.

To better understand the merits and limitations of a concept identification strategy based on wikification and community search techniques, we performed an extensive evaluation of Scode. This evaluation includes a detailed comparison of state-of-the-art wikification tools on software-related documents. We also evaluated both the end-to-end performance of Scode and the performance if its components individually. The results of our investigation contribute to a better understanding of knowledge modeling techniques for software projects. They confirmed that parsing software documents has unique challenges that decreases the accuracy of wikification techniques developed and evaluated on general-domain texts. As a results, the tools' performance was lower than what was reported in their associated research articles and no tool was consistently better than any other, making it hard to choose an optimal option for specific applications. The end-to-end performance of practical configurations of Scode ranged from 25% to 66%, but identify recurring concepts more consistently than comparable approaches. In total, we make the following contributions:

1. an assessment of the performance of wikification tools on software resources;

2. empirically grounded insights to improve the performance of wikifiers for the software domain;

3. a novel approach to identify project-related concepts using a consistent and project-agnostic terminology;

4. the results of an extensive empirical evaluation of Scode that provide nuanced insights into concept identification strategies;

[1] Scode stands for Software conceptual dependency, and is used to represent both our novel approach and its output.

[2] https://en.wikipedia.org/wiki/Main_Page

5. a curated list of over 6700 computing-related articles that represent the extent of Wikipedia's coverage of the software development domain.

We motivate and precisely define the problem addressed by Scode in Section 3.1. We started our investigation with a comparison of wikification tools (Section 3.3). Based on the results, we developed and implemented our novel approach to identify the conceptual dependencies of a software project, presented in Section 3.4. We then describe our evaluation procedure and report on its results for both the end-to-end and individual components' performance of Scode (Section 3.5).

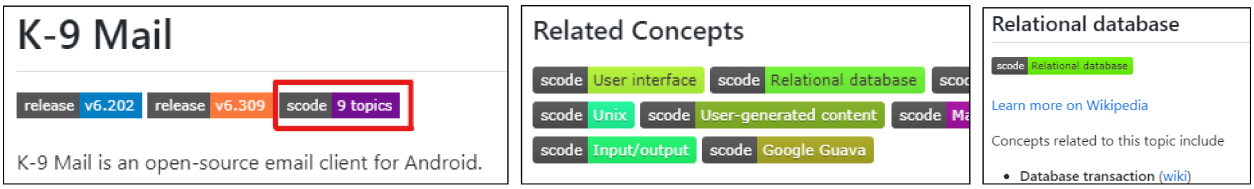
Publications Our investigation of conceptual dependencies was published in two articles. The comparison of wikifiers was published in an article titled *"Wikifying software artifacts"*, which appeared in *Empirical Software Engineering* [156]. The description of Scode and its evaluation was in an article titled *"Identifying Concepts in Software Projects"*, which appeared in *IEEE Transactions on Software Engineering* [159].

Study Data The data and material necessary to independently validate and replicate our findings is available as two data artifacts (one for each published article) [155, 157]. The artifacts contain the annotation guides, the raw data collected during the studies, and the lists of computing-related articles about software-related topics. Appendix A presents the content of the data artifacts in detail.

3.1 Motivation and Problem Definition

Prior studies have found that developers without a sufficient background find it challenging to make effective queries and judge the relevance of the documents they find when looking for documentation [109], thus making the lack of background a common obstacle to learning a new API [187].

To illustrate the importance of conceptual dependencies, we consider the scenario of a developer who wishes to contribute to the K-9 Mail Android application. This application is an open source mail client for mobile phones. As such, its development requires familiarity with concepts related to Android development (e.g., activities and intents, local data storage), email exchange protocols (e.g., IMAP, SMTP), and authentication (e.g., OAuth, multi-factor authentication), among others. If the developer is unfamiliar with some of these concepts, they may make incorrect assumptions while working on the system, misunderstand



(a) README File with Summary Badge (b) Section Listing All Topics (c) Topic Detail

Figure 3.1: Scode Badges Indicating the Relevant Topics for K-9 Mail

the documentation, or fail to recognize the importance of some constraints. Alternatively, the developer look for an explanation, e.g., on Wikipedia, when they encounter a new concept [191], disrupting their work flow. If, instead, the developer had access to a list of conceptual dependencies of K-9 Mail, they could address gaps in their knowledge before starting their contribution.

The inspection of conceptual dependencies as part of the onboarding process can be supported by showing then in a standard way on the project's web page. We envision the design of a simple application that inserts badges in a README file to represent concepts relevant to a project, grouped by topic. Figure 3.1 shows an example of the badges generated by this application for the K-9 Mail project repository. A summary badge that indicates the number of topics identified for the project is placed alongside other usual badges at the top of the README file (Figure 3.1a). The application also inserts one badge per topic in a dedicated section of the README file (Figure 3.1b). In a separate file, the application generates a section for each topic, listing the project-related concepts that belong to this topic (Figure 3.1c).

The badges fulfill a purpose similar to GitHub Topics.[3] However, whereas GitHub topics typically focus on project-wide concerns (e.g., application type, development and building dependencies), *scode* badges also capture concerns related to smaller portions of a project, for example that it includes a user interface controlled by a layout manager. The topic information file contains relevant concepts from each topic, with links to relevant Wikipedia articles, so that developers can learn about or recall unfamiliar topics [30, 191].

This scenario surfaces the limitations of techniques that extract concepts by analyzing recurrent terms in source code and documentation. First, using concept names recognized by Wikipedia, rather than the project-specific terms, ensures that they have a consistent meaning across projects. Thus, developers will know whether they need to learn more about a topic before having to read the project-specific description. Second, associating each concept

[3] https://github.com/topics/

with a Wikipedia article provides a natural way to learn about the concept when necessary. Finally, grouping concepts by topic, rather than presenting them as a flat list, helps developers navigate efficiently through the required knowledge even if a large number of concepts are presented.

3.1.1 Research Problem

We sought to retrieve concepts whose meaning is *recognized* beyond a specific project. We argue that it is easier for developers to interpret such recognized concept than project-specific concepts—or general concepts expressed using a project-specific terminology. We also sought to identify not only *explicit* concepts mentioned in a project's documentation, but also *implicit* concepts related to the project's domain. For example, a mention of the SHA-256 algorithm likely implies that the project is related to the more general concept Cryptography. Hence, Scode constitutes an alternative to prior work that synthesizes concepts from recurrent terms (e.g., [67, 103]) to build glossaries (e.g., [9, 227]) and ontologies (e.g., [1, 183]). Instead, Scode identifies concepts from an independent knowledge base, Wikipedia. We discuss the choice of Wikipedia as the information source in Section 3.1.2.

The development of efficient wikification techniques in the last decade enables the use of Wikipedia articles as proxies for concepts. *Wikification* refers to the task of linking mentions of concepts in free-form text to relevant Wikipedia articles [53, 137, 206]. It is a variant of the *named entity recognition and disambiguation* task, expanded to include unnamed entities, i.e., concepts [217]. Several tools, called *wikifiers*, are now available to perform this task on arbitrary texts. These tools are designed to retrieve the correct sense of a word in a text (e.g., associate "map" with the data structure or the visual representation depending on the context) and link synonyms to the same concept (e.g., associate "map", "dictionary", and "associative array" to the same concept).

Wikifiers, however, cannot identify concepts that are only implied by a document. For example, a text mentioning SHA-256 is also related to the implicit concept of Cryptographic hash function. Implicit concepts are important to reduce the impact of variations in the way software documentation is written when identifying related concepts. Community search algorithms can help identify such implicit concepts. The *community search* task consists of identifying, within a graph, a densely connected subgraph that surrounds a query node [68, 199]. Thus, using Wikipedia articles as nodes and hyperlinks between them as edges, we can apply community search algorithms to find further articles related to those

mentioned in the project's documentation. The *communities* found by these algorithms can also form the basis for aggregating concepts by topic.

Together, wikifiers and community search algorithms form the basis of our solution to identify recognized concepts. However, both techniques were developed for the general domain. Hence, **the goal of our investigation consists of assessing the performance of these techniques and designing heuristics to use them in a software-related context.** We ask the following research questions:

RQ 3.1 How do wikifiers compare to each other, in terms of performance, to wikify software documents?

RQ 3.2 How do the additional parameters affect the performance of each wikifier?

RQ 3.3 To what degree do different wikifiers identify similar sets of articles?

RQ 3.4 How accurate are concepts identified by Scode?

RQ 3.5 How consistent are concepts identified by Scode?

RQ 3.6 How effective are the internal mechanisms of Scode?

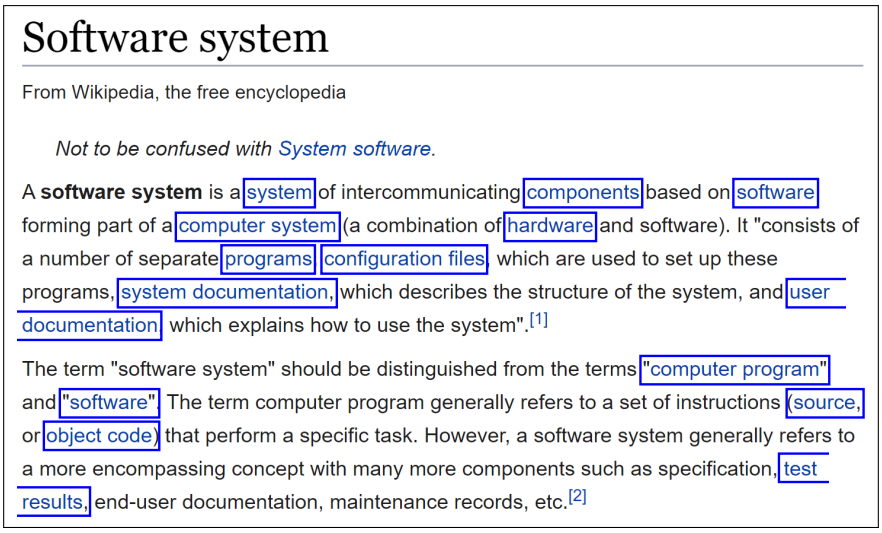
RQ 3.7 How do concepts extracted from documentation differ from those extracted from code identifiers?

We first studied RQs 3.1 to 3.3 by applying different configurations of six wikifiers to Stack Overflow[4] posts. We used the results of this comparison to inform the design of Scode. RQs 3.4 and 3.5 evaluate the end-to-end performance of Scode along two metrics, accuracy and consistency. Finally, RQs 3.6 and 3.7 assess several design decisions of Scode. Their results provide deeper insight into automated techniques to extract, represent, and manipulate software project knowledge.

3.1.2 Wikipedia as a Source of Information

With over six million articles, the extensive coverage of Wikipedia makes it a valuable knowledge base of recognized concepts. Although there are software-specific knowledge bases and glossaries (e.g., ISO/IEC/IEEE's vocabulary of systems and software engineering [101]), the subset of Wikipedia relevant to computing offers a more extensive and up-to-date coverage, thanks to its vast community of contributors.

[4] https://stackoverflow.com/questions



Software system

From Wikipedia, the free encyclopedia

Not to be confused with System software.

A **software system** is a system of intercommunicating components based on software forming part of a computer system (a combination of hardware and software). It "consists of a number of separate programs, configuration files, which are used to set up these programs, system documentation, which describes the structure of the system, and user documentation, which explains how to use the system".[1]

The term "software system" should be distinguished from the terms "computer program" and "software". The term computer program generally refers to a set of instructions (source, or object code) that perform a specific task. However, a software system generally refers to a more encompassing concept with many more components such as specification, test results, end-user documentation, maintenance records, etc.[2]

Figure 3.2: Excerpt from the SOFTWARE SYSTEM Wikipedia Article with Links to Other Articles in Blue Boxes (as of December 2023).

Wikipedia is also a popular resource within the software development community. Developers often include links to Wikipedia articles in Stack Overflow posts [20], source code comments [86], and commit messages [235]. The practice of linking to Wikipedia articles is common to help readers of a document find definitions of important concepts [191].

Aside from Wikipedia, Stack Overflow is also commonly used as a knowledge base in software engineering research [45, 163]. Similar to Wikipedia, Stack Overflow is well-known within the development community and actively maintained. Its set of over 65 000 tags (as of August 2024) covers a wide range of software technologies and programming concepts. However, the description of Stack Overflow tags is often missing [163], and they do not contain as much structured information as Wikipedia articles, such as categories and hyperlinks between articles.

3.1.3 Potential and Limitations of Wikification

Wikification is a prolific sub-area of computational linguistics that matured over the past decades. The term *wikification* refers to the process of adding links to relevant Wikipedia articles from a natural language document, so that the reader of the document can easily find description of pertinent related concepts. The result of this process is similar to existing Wikipedia articles (hence the name) which contain internal links to other articles. The wikification process was originally a manual process: authors of Wikipedia articles would explicitly add links to other articles. As the popularity of Wikipedia grew, and authors of

non-Wikipedia documents started to link to Wikipedia, e.g., from Stack Overflow and Reddit[5] posts [224], researchers designed wikifiers to help with this activity. Wikifiers automatically identify concepts mentioned in the text and associate them with a relevant Wikipedia article—or, more generally, with an entry in a knowledge base. Organizations use wikifiers, for example, on news articles to refer readers of the document to contextual or prerequisite knowledge helpful to understand a news item.

For example, Figure 3.2 shows the first two paragraphs of the Wikipedia article SOFTWARE SYSTEM,[6] which contain fourteen mentions of eleven unique articles.[7] In the context of wikification, a *mention* represents the fragment of text in the document that is associated with the Wikipedia article. In Figure 3.2, the first mentions in the first paragraph are *system*, *components*, and *software*. Mentions can consist of multiple words, such as *computer system*. In some cases, the Wikipedia article associated with a mention has a title that differs from the mention, as is the case for the mention *components*, associated with SOFTWARE COMPONENT (itself redirecting to COMPONENT-BASED SOFTWARE ENGINEERING). The objective of a wikifier is to automatically discover the same mention–article links as those identified by humans when given the raw text of the article as input.

Software engineering techniques can leverage mature wikification techniques to automatically link technical documents to supporting resources, either to help readers better understand the resource, or to improve other information retrieval techniques. However, wikification research targets well-written and general-domain documents [31, 48, 92, 145],[8] and it is unclear how well it can address the peculiarities of software-specific documents. Recent work proposed software-specific techniques related to wikification [241], but no end-to-end wikification technique exists yet for software resources, or even evaluations of general techniques on a software-related dataset.

Software engineering terminology adds complexity to the wikification task, because many common terms have a specific technical sense. For example, a "lock" in the software domain can refer to a concurrent programming concept, or a file access restriction, but less likely to the physical security device. Similarly, "Python" more often refers to the programming language than a type of snake. The fast-growing list of technologies, many of them named using common terms, makes the problem of terminology even more challenging [163].

A second challenge originates from the peculiar format of software resources. They often include code fragments, either in distinct blocks or inserted directly in the text. Source

[5] https://www.reddit.com/

[6] https://en.wikipedia.org/wiki/Software_system

[7] We mark titles of Wikipedia articles with a different FONT.

[8] There are a few exceptions of tools that target specific types of documents, such as Twitter messages [42].

code identifiers (e.g., names of variables and types) can also appear in the text without any special formatting, and often in different morphological forms [46], which makes them hard to distinguish from natural language words [240]. This mix of code and natural language leads to an uncommon syntax and many out-of-vocabulary tokens, two challenging aspects of natural language processing. Thus, the impact of the peculiar format of software resources on wikification is hard to reliably estimate, which motivates a domain-specific evaluation of the available wikifiers.

Typically, to evaluate wikification results, it is not sufficient to identify a set of related Wikipedia articles. Wikifiers must also associate these articles to the correct mention. For example, in Figure 3.2, a wikifier must associate the article SYSTEM with the mention *system*, and not other terms such as *intercommunicating*. However, Scode requires a variant of the wikification task that focuses only on the identification of a set of related articles, and disregards the mentions themselves. Thus, we consider the outcome of wikifiers as a set of articles (e.g., SYSTEM SOFTWARE, SYSTEM, and so on), rather than a list of mention–article pairs (e.g., <*system*, SYSTEM>, <*components*, SOFTWARE COMPONENT>, and so on). Meij et al. originally described this variant of wikification [137], and Cornolti et al. named it *C2W*, for *Concepts to Wikipedia* [53].

We focus on C2W because it maps more naturally to potential applications in software engineering. Once the wikifier identifies relevant articles, the exact mentions are not useful to understand the concepts related to the software-related document. Therefore, these mentions should not affect the study's results. Nevertheless, when discussing the results of wikifiers, we occasionally refer to the mention associated with an article, when the context requires it (e.g., to interpret a result).

3.2 Comparison Protocol for Wikifiers

The preparation for the comparison of state-of-the-art wikification techniques required the selection and preprocessing of a sample of Stack Overflow posts, as well as the selection of the wikifiers to compare, and which of their parameters to experiment with.

3.2.1 Types of Software Resources

Software-related resources can take many forms, such as source files, code comments, various forms of developer communications (e.g., emails, bug reports, forum posts), and technical or end-user documentation. These forms differ widely in many aspects, such as the level of

explicit structure in the document, the formality and quality of the language, and the ratio of natural language to code.

At one end of the spectrum, well-written and highly edited software resources aimed at a general audience (e.g., end-user manuals) are similar to the general-domain documents, such as news articles, used to develop and train wikification techniques. Thus, we can expect the performance of wikifiers on these documents to be similar to the performance reported for general-domain documents.

At the other end of the spectrum, software resources composed entirely of source code are clearly outside the intended scope of wikifiers, which take natural language as input. Hence, although the wikifiers may identify a few relevant concepts from identifiers in code artifacts, source code does not constitute an appropriate input for evaluation. The results would reflect the effectiveness of the preprocessing steps (e.g., identifier tokenization, aggregation into sentences) rather than the performance of wikifiers.

As a middle ground, the evaluation set consists of Stack Overflow posts, which are mainly written in natural language, but also contain code fragments. Some of the posts are long and well-structured documents, edited many times by the community to improve the quality of the language and add thorough descriptions of related concepts. These posts are similar to smaller versions of technical documentation and end-user manuals. At the other end of the spectrum, some posts are closer to fragments of informal discussions, with short replies and grammatically incorrect sentences. Posts also vary in their natural language-to-code ratio, from some that contain only text to others with only code, including posts with code formatted as natural language. In terms of information content, Stack Overflow covers virtually all computing domains. Finally, software engineering research often leverage Stack Overflow as a source of knowledge [21, 175, 215], which makes the results of this study directly applicable to techniques that use Stack Overflow posts as input data.

3.2.2 Sample Selection

Our evaluation set contains 500 Stack Overflow posts from the September 2019 Stack Exchange archive.[9] This set is a uniform random sample from all 44 016 828 posts (questions and answers alike) with a nonnegative score,[10] considering only the most recent version of each post. We

[9] https://archive.org/details/stackexchange

[10] The score of a Stack Overflow post is visible next to the post on Stack Overflow, and represents the number of "upvotes" minus the number of "downvotes" attributed by Stack Overflow users based on the usefulness and quality of the post.

Table 3.1: Properties of the 500 Selected Stack Overflow Posts

Property	**Value**
Type of post	186 questions, 105 accepted answers, 209 non accepted answers
Score	min: 0 Q1: 0 Q2: 1 Q3: 2 max: 96 avg: 2.5
# Words	min: 0 Q1: 29 Q2: 54 Q3: 102 max: 459 avg: 72.9
Creation year	08-09: 21 10-11: 74 12-13: 95 14-15: 118 16-17: 112 18-19: 80
Edit (days)	min: 0 Q1: 0 Q2: 0 Q3: 0 max: 3282 avg: 108

min = minimum value; Qn = n-th quartile; max = maximum value; avg = average;
Edit = number of full days between the creation of the post and its last editing

chose to discard posts with negative scores (3.9% of the population) to avoid the bias from documents explicitly flagged as problematic.

Typically, in quantitative research, the sample size ensures that statistics computed on the sample generalize to the population within a known error margin, at a predefined confidence level. This was not possible in our study, because the statistics used to compare wikifiers (precision and recall) measure proportions of Wikipedia articles, generated by wikifiers, rather than posts.[11] Because the articles are neither independent nor randomly generated, they do not meet the necessary assumptions for statistical generalization. This situation is not uncommon when evaluating wikification approaches. Prior work often reuses standard annotated corpora, which are not random at all, but instead allow for direct comparison of different approaches (e.g., [31, 149]). We discuss the implications of the sampling procedure and the results of a sensitivity analysis with the threats to validity (Section 3.3.6).

Nevertheless, we designed our sample to be *representative* of the population, in the sense that insights gained from this sample should apply to the whole population. With this regard, despite its small size relative to the population, the sample is sufficiently large to contain a non trivial number of posts related to the popular topics discussed on Stack Overflow, as well as posts that exhibit common characteristics, such as long and short posts, posts with and without code fragments, and posts that are more or less well written and formatted. Hence, the sample size of 500 posts gives us reasonable confidence that our findings do not merely reflect spurious relations.

Table 3.1 shows summary statistics for the 500 posts in the sample: the first row indicates the number of questions and answers, the second and third rows, the distribution of score and number of words (excluding code blocks) per post, and the fourth and fifth rows, the

[11] One alternative is to aggregate each statistic per post first, then average them over all posts. However, this alternative gives more weight to small posts with few related articles, which would be detrimental to the interpretation and generalizability of the results.

year in which the post was created, and the last time it was edited, in days, relative to its creation date. Each type of post amounts to a non trivial proportion of the dataset. Almost half (42.8%) of the posts have a score of 0. Posts contain an average of 72.9 words, with six of them having no word (i.e., they contain only a code fragment). The sample contains posts reasonably distributed among the years, although the early years of the forum unsurprisingly contain fewer posts. Finally, a majority of posts (86.6%) were not further edited past one day after their creation, with one notable outlier edited almost nine years after its creation. Overall, these statistics show that the sample consists of posts that vary in length (from a few words to multiple paragraphs) and time (they do not only reflect a specific period). Most, but not all, posts are not heavily edited or highly scored, as would be, for example, popular blog posts.

3.2.3 Preprocessing of Posts

The Stack Overflow archive stores the post bodies as HTML-formatted documents. Although some wikifiers can take as input HTML documents, not all of them can. To perform an equivalent comparison, we used the jsoup[12] library, version 1.11.2, to convert the HTML posts to plain text, and provided this text as input to all wikifiers.

Because wikifiers take natural language as input, not code, we removed code blocks (identified by HTML pre tags) from the post bodies, but kept inline code fragments (identified by HTML code tags without pre tags). The rationale behind this decision was to avoid breaking up sentences, and because users do not consistently format inline code fragments as such: some users use the "inline code" format option for terms other than code, such as names of technologies, whereas others do not use this formatting option at all, even for legitimate code fragments.

We excluded all other information, such as comments or edit messages, from the input. In particular, we did not include question titles, to avoid inconsistencies between questions and answers, and because most question bodies are self-contained.

3.2.4 Selection Procedure for Wikifiers

For this study, wikifiers must be able to generate a list of Wikipedia articles about concepts mentioned in an input Stack Overflow post. Prior work has proposed a large number of wiki-

[12] https://jsoup.org/

fication techniques, but only a small subset of these techniques come with an implementation that can wikify custom texts. We used the following criteria to select wikifiers:

1. The wikifier must be usable immediately, without an intervention from the user (other than installing the necessary software and possibly writing short driver code). In particular, the wikifier must not require the user to provide their own training set.

2. The wikifier must identify Wikipedia articles from an input consisting of plain text. If the wikifier links concepts to another knowledge base, it must be possible to associate (most) entries of this knowledge base with Wikipedia articles in a straightforward manner.

3. The wikifier must be freely available (for non-commercial use), either as a web service, packaged executable, or source code. For web services, requests can be limited to a reasonable rate.

4. The wikifier must provide an interface that allows it to be programmatically integrated as a component of a larger software system. In particular, the wikifier must not be a tool for demonstration only or a replication package for a specific experiment. The interface must be sufficiently well documented to be usable with reasonable effort.

5. The wikifier must not be deprecated by a more recent wikifier developed by the same group.

3.2.5 Selection Procedure for Configuration Parameters

Each wikifier has a number of configuration parameters, used to tune their performance. Not all parameters, however, should be tuned by the end users: for example, some hyperparameters affect the training phase of the wikifier, and other parameters have their values optimized during the training phase. For each wikifier, we carefully examined each of their parameters to understand whether the parameter would likely be tuned by an end user, or be fixed to an appropriate default value.

All wikifiers have at least one parameter that roughly estimates the confidence that an article is mentioned in the post, and is expressed as a number in the unit interval. Although this parameter has different names in different wikifier implementations, for consistency, we refer to it as the *confidence threshold*, except when referring to a specific wikifier. The confidence threshold is the main parameter that end users employ to tune the wikifier performance, and our comparative analysis focuses primarily on variations of this parameter.

Table 3.2: Wikifiers Compared in this Study

Wikifier	Knowledge Base	Confidence	Additional Parameter
Ambiverse	YAGO	confidence	NER: knowner/stanford
Babelfy	BabelNet	score	MCS: on/off
DBpedia	DBpedia	confidence	support
Illinois	Wikipedia	score	–
JSI	Wikipedia	1-pageRankSqThreshold	pageRank
WAT	Wikipedia/Wikidata	rho	tokenizer: opennlp/lucene

Some wikifiers also have a *secondary numeric parameter* that allows to filter results based on a numerical property different from the confidence threshold. Other wikifiers have a *binary option* to select which of two components to use to perform one of the wikification subtask, or to enable or disable an optional algorithm. We studied the impact of these parameters on the wikifiers performance as well.

Finally, some of the configuration parameters that must be tuned by the end user relate to concrete and objective properties of the wikification task, such as the input language (e.g., English). Such parameters also include options to disable key components of the approach, to perform ablation studies on the wikifier. For those parameters, we selected the most appropriate value based on the sample of posts: we did not disable any key component, and we selected values that fit the properties of Stack Overflow posts (e.g., they are written in English).

We relied on the wikifier's documentation and preliminary experimentation to distinguish between parameters that should be tuned, parameters that should receive a fixed, appropriate value, and training hyperparameters or parameters that should keep their trained value.

3.2.6 Selected Wikifiers

We identified six tools that respect the criteria presented in Section 3.2.4. Each tool can be used on a computer with 16 Gb of memory and 400 Gb of storage. Table 3.2 shows, for each wikifier, the knowledge base it resolves mentions to and the names of the confidence threshold and additional tuning parameters studied. The following paragraphs describe each wikifier and their parameters in more details. The versions of both the wikifiers and their corresponding knowledge base is given when it is available. We also relate the performance measures reported in the original articles that present each wikifier.

AmbiverseNLU[13] (Ambiverse) is an open source natural language understanding Java suite that resolves entities to the YAGO knowledge base [184]. It can be added as a Maven dependency to Java projects, or run from a Docker container. Its entity recognition component, KnowNER [194], uses several knowledge base derived features in a linear chain conditional random field (CRF) model to improve the state of the art. Its disambiguation component, AIDA [92], uses a linear combination of a prior probability based on popularity, a similarity score, and a coherence metric. In this study, we used the Maven artifact, version 1.1.0, with the database dump version aida_20180120_cs_de_en_es_ru_zh_v18. It is the most resource-consuming wikifier, requiring 16 Gb of memory to parse English texts and a little under 400 Gb to import the YAGO database. The performance of AIDA and KnowNER were evaluated separately. Hoffart et al. reported a precision of 82% at the 100% recall level for AIDA [92], and Seyler et al. reported F1 scores between 88% and 91% for KnowNER [194].

The confidence threshold of Ambiverse is named *confidence*. Ambiverse also has a secondary numeric parameter, *salience*, which indicates the relevance of the named entity to the document. However, because the salience is always 0 on non-named entities (i.e., concepts such as COMPUTER and DEBUGGING), which constitute most of the mentions, we did not consider it. Ambiverse also has a binary option: it allows users to use an alternative NER component (*stanford*), instead of KnowNER (*knowner*).

Babelfy[14] is an online wikifier with a REST API that resolves entities to BabelNet, which also offers a REST API [149, 165]. To wikify a document, Babelfy first identifies a large set of candidate entities for each (possibly overlapping) mention using string matching heuristics, then leverages a densest subgraph algorithm to filter among the entities, so that the entities retained form a coherent set. Both Babelfy and BabelNet APIs share a daily quota of 1000 daily requests by default (after registration). This project also has a commercial counterpart, Babelscape. For this study, we used the non-commercial service. The version of the Babelfy REST API is not available, but the website states that it uses version 3.0 of BabelNet. Moro et al. reported an accuracy between 72% and 82% for Babelfy on the entity linking task,[15] and an F1 score of 87% for the disambiguation task only [149].

The confidence threshold of Babelfy is simply named *score*. Babelfy also offers the binary option to activate a fallback strategy, termed "most common sense" (MCS). When this option is enabled, if the main disambiguation algorithm fails to identify a Wikipedia article for a

[13] https://github.com/ambiverse-nlu/ambiverse-nlu
[14] http://babelfy.org/
[15] They do not provide an explicit definition for accuracy.

mention, it uses its most common sense (i.e., the article with the highest prior probability). All articles linked with the MCS strategy have a confidence value of 0.

DBpedia Spotlight[16] (DBpedia) [58, 140] is an online wikifier that resolves entities to DBpedia [114]. DBpedia Spotlight also identifies candidates using simple string heuristics, then disambiguates between the candidates with a vector space model to represent knowledge base entities. The wikifier uses a custom measure derived from tf-idf, called tf-icf (for "inverse candidate frequency") and the cosine similarity metric to weigh and compare vectors. DBpedia Spotlight's REST API does not impose any rate limit, but during our study, the server was unstable and would often return HTTP 502 or 503 errors for valid requests. The website does not mention the version of the wikifier, nor the backing knowledge base. Mendes et al. reported a precision-recall plot of DBpedia, where precision varies between 40% and 82% for recall values between 12% and 61%, with a maximal F1 score of 56% [140].

Similar to Ambiverse, DBpedia's confidence threshold is named *confidence*. However, contrary to all other wikifiers, DBpedia's confidence score must be set a priori. We used values between 0 and 1, with 0.1 increment (i.e., 0, 0.1, 0.2, ..., 0.9, 1.0) to sample the parameter space. DBpedia returns a secondary numeric parameter, *support*, which is a positive integer that is a property of the entity (i.e., independent of the associated mention or input text).

Illinois Wikifier[17] (Illinois) is an open source Java wikifier that resolves entities to Wikipedia [48, 182]. To disambiguate mentions to articles, Illinois includes several local and global features into a linear combination. Local features, which are defined between a mention and a Wikipedia article, use the cosine similarity on tf-idf vectors of the mention and the article. Global features use the pointwise mutual information (PMI) and normalized Google distance (NGD) metrics on the Wikipedia link graph between all pairs of entities. Finally, Illinois improves the results by using semantic relations, extracted from the input text, between entities. Due to many dependencies to outdated maven artifacts, we experienced some difficulty in importing the wikifier into our project. To solve our issues, we downloaded a compiled distribution, version 3.0, that includes all dependencies. Cheng and Roth reported F1 scores between 81% and 93% on four datasets for Illinois [48].

Similar to Babelfy, Illinois' confidence threshold is named *score*. Illinois also offers several predefined configurations, of which *STAND_ALONE_NO_INFERENCE* was the most

[16] https://www.dbpedia-spotlight.org/

[17] https://cogcomp.seas.upenn.edu/page/software_view/Wikifier

appropriate for custom text (the "with inference" configuration requires the commercial Gurobi Optimizer software).

JSI Wikifier[18] (JSI) is an online wikifier with a REST API that resolves entities to Wikipedia [31]. At its core, JSI uses an augmented mention–entity graph of Wikipedia: first, JSI constructs the bipartite graph with anchor text (mentions) on one side, and articles (entities) on the other, and edges linking each anchor text to the article they point to. Then, it augments this graph with edges between articles that are similar, according to a similarity metric based on internal links. JSI computes pagerank values [168] on the nodes of this graph, and returns the set of articles with the highest pagerank values. JSI's REST API requires registration, but does not impose any rate limit. It does not provide version information on either the software or backing knowledge base. Brank et al. reported an F1 score of 59% for JSI [31].

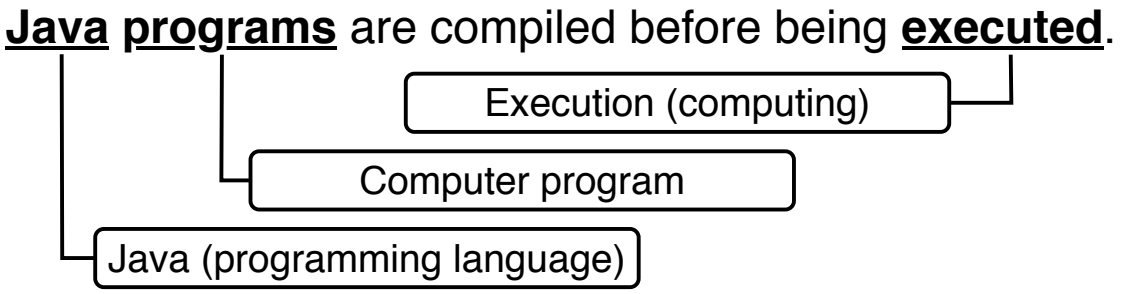
Instead of using an absolute pagerank value as its confidence threshold, JSI computes the threshold as a proportion of the sum of the squares of all pagerank values. Thus, the user-defined proportion constitutes the confidence threshold (named *pageRankSqThreshold*). Because a lower proportion leads to a lower recall, for this study, the confidence threshold is actually *1 - pageRankSqThreshold*. JSI also returns *pageRank* values with the results, so we considered them as a secondary numeric parameter.

WAT[19] is an online wikifier with a REST API that resolves entities to Wikipedia and Wikidata [173]. WAT is the successor of another wikifier, TagME [71]. To identify mentions and candidates, WAT trained a binary classifier with features from Wikipedia articles, such as redirect titles and anchor text of internal links. To disambiguate between candidates, WAT leverages two algorithms. The "voting" algorithm developed for TagME computes the sum of a relatedness measure between pairs of extracted entities, weighted by a priori probabilities. The other algorithm is based on a mention–entity graph similar to that of JSI, but computed only over mentions and entities extracted in the first step. Similar to JSI, WAT imposes no rate limit, but requires registration. It also does not provide any version information. Piccinno and Ferragina reported precision and recall values respectively between 46% and 49%, and between 51% and 59%, for different configurations of WAT [173].

WAT's confidence threshold is named ρ (*rho*). The only other parameter (apart from the input language) is the *tokenizer* to use: *opennlp* or *lucene*.

[18] https://wikifier.org/

[19] https://services.d4science.org/web/tagme/wat-ap



Titles in round boxes indicate the expected Wikipedia article that wikifiers should identify.[20] Exact mentions (in bold and underlined) are shown for better understandability, but are not evaluated.

Figure 3.3: Sample Sentence with a Gold Standard Wikification

A notable tool missing from this list is Milne and Witten's **Wikipedia Miner** [145, 146]. This is one of the first available wikifiers, and it has served as a baseline for the evaluation of many other wikifiers. However, the web service is no longer operational, and we could not find the source code with pre-trained models.

3.2.7 Data Annotation

The evaluation of wikifiers consisted of executing all of them on the posts from the dataset, collecting all generated pairs of articles and posts, and manually annotating each pair as correct or incorrect. This procedure generated a *reference set* to assess the precision and recall of each wikifier.

The usual method to evaluate a wikifier is to compare its output with a gold standard (e.g., [31, 149]), which allows to automatically discriminate false positives (FP) from true positives (TP) and list all false negatives (FN). Figure 3.3 shows a sentence with a corresponding gold standard.[20] Three articles are expected to be found by the wikifiers. If a wikifier outputs the two articles Java (programming language) (correct) and Compilation (album) (incorrect), there would be one true positive, one false positive, and two false negatives. From these counts, it is possible to compute the precision and recall.[21] The wikifier in the previous example would have a precision of 0.5 (i.e., 1/2) and a recall of 0.3 (i.e., 1/3). F1 scores[22] are also commonly reported for wikifier evaluations, as a single objective function that balances precision and recall. However, in the context of our study, a single arbitrary

[20] At the time of writing, Wikipedia did not have an article specifically on source code compilation, but only an article about Compiler. For this study, we considered the distinction between the two concepts (a process and tools to perform it) significant enough to reject the article Compiler for the mention *compiled*.

[21] precision $= \frac{TP}{TP+FP}$; recall $= \frac{TP}{TP+FN}$

[22] F1 score is the unweighted harmonic mean of precision and recall, or $\frac{2 \times \text{precision} \times \text{recall}}{\text{precision}+\text{recall}}$

objective function would only hide the more precise information captured by precision and recall, so we chose to present the latter metrics separately.

With such a method of evaluation, the most effort-intensive and critical task is to generate the gold standard. Creating a high quality evaluation dataset is hard, especially when such concepts are not named entities, as humans do not always agree on the concepts that should be linked. Because of this difficulty, wikifiers are commonly evaluated against standard golden datasets, such as AIDA-CoNLL, which consists of 1393 news articles manually annotated by Hoffart et al. [92].[23] For example, Brank et al. [31] used this dataset as a gold standard, and Moro et al. [149] used six different gold standards, including AIDA-CoNLL.

None of these available gold standards are specific to software engineering, and in particular, none of them uses Stack Overflow posts. To mitigate the cost of creating a new, high quality evaluation dataset, we used a slightly different approach. We generated a *reference set*, rather than a gold standard, to compare the relative, rather than absolute, performance of wikifiers. We executed all wikifiers on each post, with parameters that maximize recall. We then manually annotated each association between an article and a post as correct or incorrect (without knowing which wikifier produced which associations).[24] The reference set generated with this procedure is sufficient to discriminate false positives from true positives, and have a consistent list of false negatives across all wikifiers.

The precision values computed with this procedure are equivalent to those that would be derived from a gold standard. However, recall values are only *relative* to all true positives found by any of the six wikifiers, instead of all theoretically possible true positives. Thus, the ratios of recall values between wikifiers is consistent with the ratios that would be derived from a gold standard, but their absolute magnitudes are not.

Using this procedure, it is possible to accept multiple articles for the same mention, which would not be possible with a gold standard. For example, for the sentence *"Paypal is a payment system."*, standard datasets would associate the mention *payment system* with either Payment system or E-commerce payment system. But, using our reference set, both of them are correct associations.

[23]The AIDA-CoNLL dataset only links proper nouns (i.e., named entities), but other datasets exist for both named entities and concepts.

[24]The term "annotate", in the wikification community, often refers to the wikification process itself, or a variant. In this article, we use "annotate" (and its derivatives) only when referring to the manual annotation of human experts to assess the correctness of an article–post pair.

3.2.8 Annotation Task

The annotation task consisted of answering, for each article–post pair, the single question

> *Is the concept represented by the Wikipedia article related to computing and explicitly mentioned in the Stack Overflow post?*

The above question is the result of an iterative process over a pilot set to mitigate the subjectivity of the task as much as possible, so that the results would be objective and lead to an unambiguous interpretation. It requires two conditions for an article–post pair to be *correct* (i.e., for a positive answer to the question): the article must be *explicitly mentioned* in the post, and it must be *related to computing*. The following paragraphs discuss in details why we needed to include these conditions.

Condition 1 (Explicit Mention). Some concepts do not have an associated Wikipedia article. For example, there is no article dedicated to the HTTP GET method (it is described in a section of Hypertext Transfer Protocol). When concepts with no Wikipedia article appear in a document, wikifiers sometimes link the mention to a related article (e.g., linking *GET* to Hypertext Transfer Protocol). However, for the annotation task, we require that related articles (including articles about hypernyms and holonyms) be marked as "incorrect". Only articles about concepts that are actually mentioned in the post can be "correct". This condition is necessary to avoid an ambiguous or subjective definition of relatedness threshold, and favors wikifiers that link specific mentions (e.g., *Java*) to specific articles (e.g., Java (programming language)) instead of general ones (e.g., Programming language).

Condition 2 (Related to Computing). The pilot annotation task revealed that wikifiers often correctly identify mentions that are nonetheless irrelevant to the context, such as phatic expressions (e.g., *Thanks*), figures of speech (e.g., idioms and metaphors), common functional phrases (e.g., *There exists*), and terms related to the Q&A nature of Stack Overflow (e.g., *question* and *answer*). Typical wikification guidelines, such as those from the Wikipedia Manual of Style on linking,[25] suggest to only link relevant mentions, and avoid "everyday words understood by most readers in context" [231]. However, relevance is often subjective. To avoid this subjectivity, an article is considered relevant if and only if it is related to

[25] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking

computing.[26] Consequently, we reject genuinely relevant but non-computing articles, such as AZIMUTH in a post about astronomy-related libraries,[27] but such articles are rare in a computing-related forum.[28]

To help annotators, we designed an extensive annotation guide that discusses corner cases (e.g., different parts of speech, synonyms, and antonyms) and what exactly is *related to computing*. Elements of this guide are motivated in part by the structure and conventions of Wikipedia, and in part by the lessons learned during the pilot annotation task. The guide also contains a curated list of five Stack Overflow posts from the pilot set, with all associated concepts annotated jointly by the authors, to serve as an example for annotators. This annotation guide is available in our online appendix, and replicated in Appendix A.1.

3.2.9 Annotators

Despite the extensive annotation guide, the annotation task still required a considerable effort from the annotators. The difficulty of the annotation task arises from quality issues from the Stack Overflow posts,[29] as well as potentially misleading Wikipedia titles.[30]

These difficulties related to the annotation task result in a steep learning curve for the annotators, and an increased threat of quality degradation that would be hard to detect if a careless external annotator wishes to finish the task too quickly. Consequently, we could not efficiently distribute the annotation task to many external annotators, and the two authors annotated all pairs. After the initial learning curve, the authors were able to annotate roughly 1000 pairs per hour.

[26] We express our criterion in terms of *computing*-related articles, rather than only those specific to *software engineering*, because the precise boundary of software engineering is less well defined than that of computing, especially among Wikipedia articles.

[27] https://stackoverflow.com/questions/2348415

[28] There are concepts that can conceivably be related to computing in some contexts, but part of a general body of knowledge in others. BLOG is such a concept: We consider that, when referring to a specific post, the concept is not a technical term, but when discussing the creation of a blogging platform, this term becomes related to computing.

[29] In addition to making grammatical errors, post authors often use natural language shortcuts such as abbreviations, acronyms, omissions, and ambiguous terms, which require additional effort from annotators to resolve. The misuse of formatting options, such as formatting code blocks as inline code, also makes posts harder to understand. For example, the scope keyword "until successful" can easily be misread as a preposition and an adjective if it is not formatted as inline code (https://stackoverflow.com/questions/41998618).

[30] Annotators cannot assume the topic of a Wikipedia article only by its title. For example, the article JAVA does not describe the programming language, but the Indonesian island. Also, although most titles that consist only of three capital letters lead to disambiguation pages, the article URL does not. Redirect titles further add to the possible disconnect between titles and article content. Thus, annotators must make the effort to scan the article to verify what it actually describes.

Table 3.3: Agreement Between Annotators Using Cohen's κ Statistic

	Author A	**Author B**	**External 1**	**External 2**
Author A	—	0.83 (940)	0.72 (1007)	0.74 (1007)
Author B	0.83 (940)	—	0.60 (569)	0.64 (569)
External 1	0.72 (1007)	0.60 (569)	—	0.67 (1576)
External 2	0.74 (1007)	0.64 (569)	0.67 (1576)	—

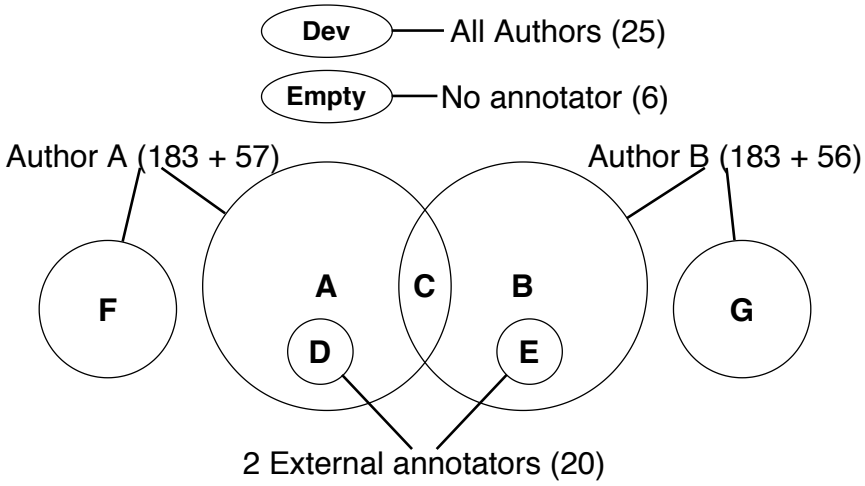
The number of common article–post pairs are shown in parentheses.

Investigator bias, common in situations where the investigators perform the annotation, is minimal in this study, because it is an independent evaluation of already existing tools. Nevertheless, we took great care to mitigate even the threat of this bias as much as possible. Annotators were not able to discern which concept was produced by which wikifier, preventing unintentional biases. Furthermore, the set of pairs to annotate was split into two mostly disjoint sets, for efficiency reasons, but both annotators annotated a common, unmarked set, to estimate their agreement on the complete set. Finally, we hired two additional annotators, external to our research group, to re-annotate a subset of pairs, to better understand the difficulty and subjectivity of the task, and further control any biases that could arise. Table 3.3 shows the agreement between each annotator, using Cohen's κ statistic [52]. The agreement between external coders and the authors are between 0.60 and 0.74, which Landis and Koch consider "substantial" [112], and the agreement between both authors, who annotated the complete dataset, is "almost perfect" (0.83).

Following the computation of the agreement scores, the authors jointly resolved all conflicts in the overlapping sets by discussing each conflicting pair and deciding on the correct annotation.

3.2.10 Annotation Sets

The wikification of all 500 posts produced 41 124 article–post pairs to annotate. We performed the annotation in two phases. In the initial phase, we annotated the pairs from 385 posts. To gain additional confidence in the reliability of our results, we then conducted a second annotation phase, in which we annotated the pairs from the remaining 115 posts. These two phases allowed us to assess the generalizability of our conclusions by observing the variation in the performance of wikifiers when augmenting the sample by 29.9%. Indeed, we observed very little variation (less than 2 percentage points of precision and recall).



The letters identify each partition, and the total number of posts for each annotation set is shown in parentheses.

Figure 3.4: Annotation Sets with their Respective Annotator(s)

Pairs formed four (non disjoint) sets: one *development* set, two *main* sets, and one *external validation* set.

To improve the efficiency of the annotation task, all pairs from the same post are put in the same set, so that each annotator has fewer posts to read. Therefore, in the following, we describe annotation sets by referring to the Stack Overflow posts they contain, rather than the article–post pairs. Figure 3.4 summarizes these sets.

Six of the 500 posts (four from the first phase and two from the second phase) happened to contain only a code block, which is removed at the preprocessing stage. Thus, these six posts produced no article for any wikifier, and were naturally not considered when creating the annotation sets.

Of the remaining 494 posts, 25 constituted the development (dev) set, used for the iterative pilot. Both authors annotated this set to refine the annotation task and guide. In contrast to many other evaluations, where development sets are discarded from the final results, the results include the final annotations of this set, as they constitute valid information that does not threaten the validity of the results. In fact, these annotations are possibly of higher quality due to the many iterations to reach a consensus.

The empty and development sets are disjoint from the other sets. We split the remaining 471 posts into seven partitions: two large partitions of 163 posts each (A and B), three small partitions of 10 posts each (C, D, and E), and two additional partitions of 57 and 56 posts for the second phase (F and G). Author A annotated the first main set, consisting of partitions A, C, and D in the first phase, and partition F in the second phase. Author B annotated the second main set, consisting of partitions B, C, and E in the first phase, and partition G in

the second phase. Each external annotator annotated the same validation set, consisting of partitions D and E. We designed this strategy so that the overlap between any two annotators consists of all pairs from at least ten distinct posts, to compute the inter-rater agreement scores shown in Table 3.3.

3.3 Wikifiers Comparison Results

Of all 500 posts, 41 (8.2%) did not link to any correct article, and the two posts with the most links have 37 and 28 linked articles[31] (average: 6.1, median: 5). We present the results relating to RQ 3.1 in Section 3.3.1, RQ 3.2 in Section 3.3.2, and RQ 3.3 in Section 3.3.3.

A considerable artifact of this study is the outcome of the annotation task, a manually verified list of 1098 computing-related Wikipedia titles, and 10 854 negative examples.[32] Section 3.3.4 explores possible ways to use this list to further improve the performance of wikifiers.

3.3.1 Wikifiers Performance

This section describes how wikifiers compare to each other, with the explicit objective of helping software engineering researchers select a wikifier that best suits their need. For this comparison, only the confidence threshold of each wikifier varies. Additional parameters are set to optimal values, described in Section 3.3.2. To compare wikifiers, we use the precision and relative recall metrics (see Section 3.2), which have an intuitive interpretation and are not affected by true negatives, irrelevant in the context of wikification.

Figure 3.5 compares the performance of all six wikifiers, in terms of precision and recall, for all confidence thresholds between 0 and 1. Each marker indicates an increment of 0.1 of the confidence. In the case of Babelfy, no confidence score was below 0.7, which explains why only four markers appear on the plot. The performance of all six wikifiers shows the difficulty of wikifying software resources, with precision levels hardly reaching 70% even for low relative recall values (10%), and rapidly decreasing under 50% for higher recall values. Interestingly, no single wikifier completely outperforms another: each one achieves the highest precision for a portion of the recall range. Therefore, before choosing a wikifier, it is important to decide on the acceptable precision or recall values for the specific application, as the optimal wikifier (and confidence threshold) depends on this decision.

[31] https://stackoverflow.com/questions/55893389 and https://stackoverflow.com/questions/21646135

[32] The number of distinct articles is slightly less than 1098 and 10 854, respectively, because some titles are actually redirect pages to other articles.

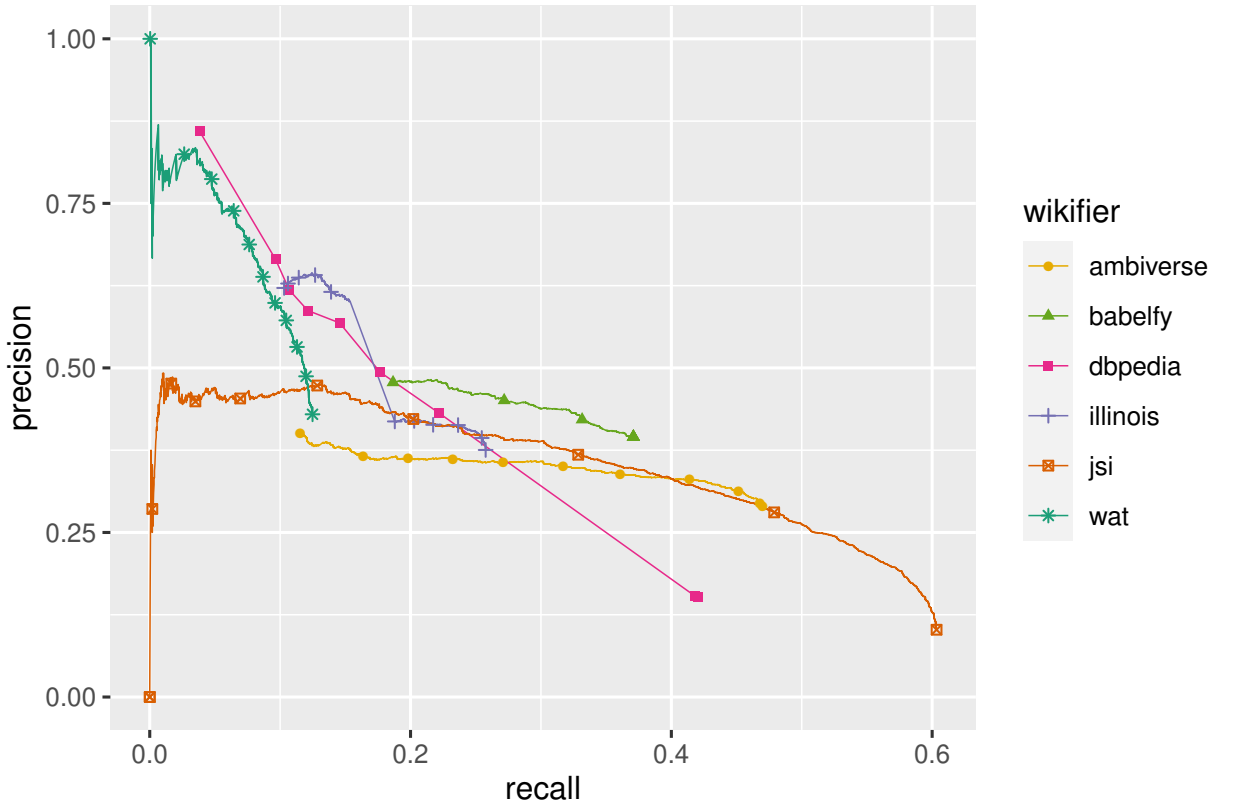


Figure 3.5: Precision-Recall curves of all six wikifiers. Markers indicate each 0.1 increment of the confidence threshold (some markers overlap). For DBpedia, a linear interpolation estimates precision and recall values for confidence values that are not exact multiple of 0.1.

Another interesting observation is the difference in the precision and recall ranges. Variations in the confidence threshold of some wikifiers, such as DBpedia and JSI, will greatly affect the recall, with the expected degradation in precision as recall increases, whereas other wikifiers, such as Babelfy and Illinois, have a much lower degradation in precision, but cannot achieve the same recall. The former wikifiers can be used in more varied contexts, but the latter wikifiers perform more consistently.

Finally, some wikifiers, including Illinois and JSI, exhibit a decline in precision for the highest confidence values. This counterintuitive drop in precision is possibly due to these wikifiers typically giving higher confidence values only to concepts from popular domains, due to their high prior probability. For example, two of the articles associated with the highest confidence values by JSI are Worldwide Exchange (about a television business news program) and Midfielder (a position of football/soccer). Such concepts are more likely to appear in news articles targeted to a general audience, which are used to train wikifiers, than

Table 3.4: Comparison of the Precision of Each Wikifier for Selected Recall Values

	Ambiverse		**Babelfy**		**DBpedia***		**Illinois**		**JSI**		**WAT**	
Rec.	CT	Pr.	CT	Pr.	CT	Pr.	CT	Pr.	CT	Pr.	CT	Pr.
5	–	–	–	–	0.98	**82**	–	–	0.56	47	0.78	76
10	–	–	–	–	0.87	**65**	–	–	0.44	46	0.36	59
15	0.94	38	–	–	0.59	56	0.52	**61**	0.37	46	–	–
20	0.79	36	0.99	**48**	0.45	46	0.41	42	0.30	42	–	–
25	0.66	36	0.93	**46**	0.39	39	0.13	41	0.26	40	–	–
30	0.54	36	0.85	**44**	0.36	32	–	–	0.22	39	–	–
35	0.42	34	0.75	**41**	0.33	25	–	–	0.18	36	–	–
40	0.32	33	–	–	0.31	18	–	–	0.15	**33**	–	–
45	0.20	**31**	–	–	–	–	–	–	0.12	30	–	–
50	–	–	–	–	–	–	–	–	0.08	**26**	–	–
55	–	–	–	–	–	–	–	–	0.05	**22**	–	–
60	–	–	–	–	–	–	–	–	0.00	**13**	–	–

For each recall level (Rec.), the corresponding confidence threshold (CT) and precision (Pr.) are shown. Dashes (–) indicate that the wikifier did not achieve such recall. DBpedia values (marked by an asterisk *) are linear interpolations. For each recall, the best precision among all wikifiers is shown in bold. All precision and recall values are percentages.

computing concepts, so they are more likely to receive favorable biases during the training phase of wikifiers.

Table 3.4 shows the value of the confidence threshold needed to achieve different recall values (from 5% to 60%, in steps of 5%), with the associated precision. The highest precision in each row is indicated in bold. For example, a user who wants to wikify Stack Overflow posts with 25% recall should use Babelfy with a confidence of 0.93, and expect a precision around 46%. Conversely, if the user requires a precision of at least 60%, they should choose Illinois, with a confidence around 0.52, and expect a recall around 15%.

Findings: All wikifiers have a precision under 50% for relative recall values over 20%. These values greatly differ from the performance reported in the original articles that describe each wikifier. These differences are possibly due to the peculiarities of software documents, such as the presence of polysemous terms with domain-specific definitions, of technologies named after common terms, and of mentions of code elements within paragraphs. For different recall values, all wikifiers in turn achieves the highest precision, which means that no wikifier is irrelevant.

3.3.2 Effect of Additional Parameters

The previous section compares all wikifiers, varying only the confidence threshold. Additional parameters allow to further fine-tune the wikifiers, but, as the result will show, at least in the context of Stack Overflow posts, there is generally little incentive to modify the default parameter values.

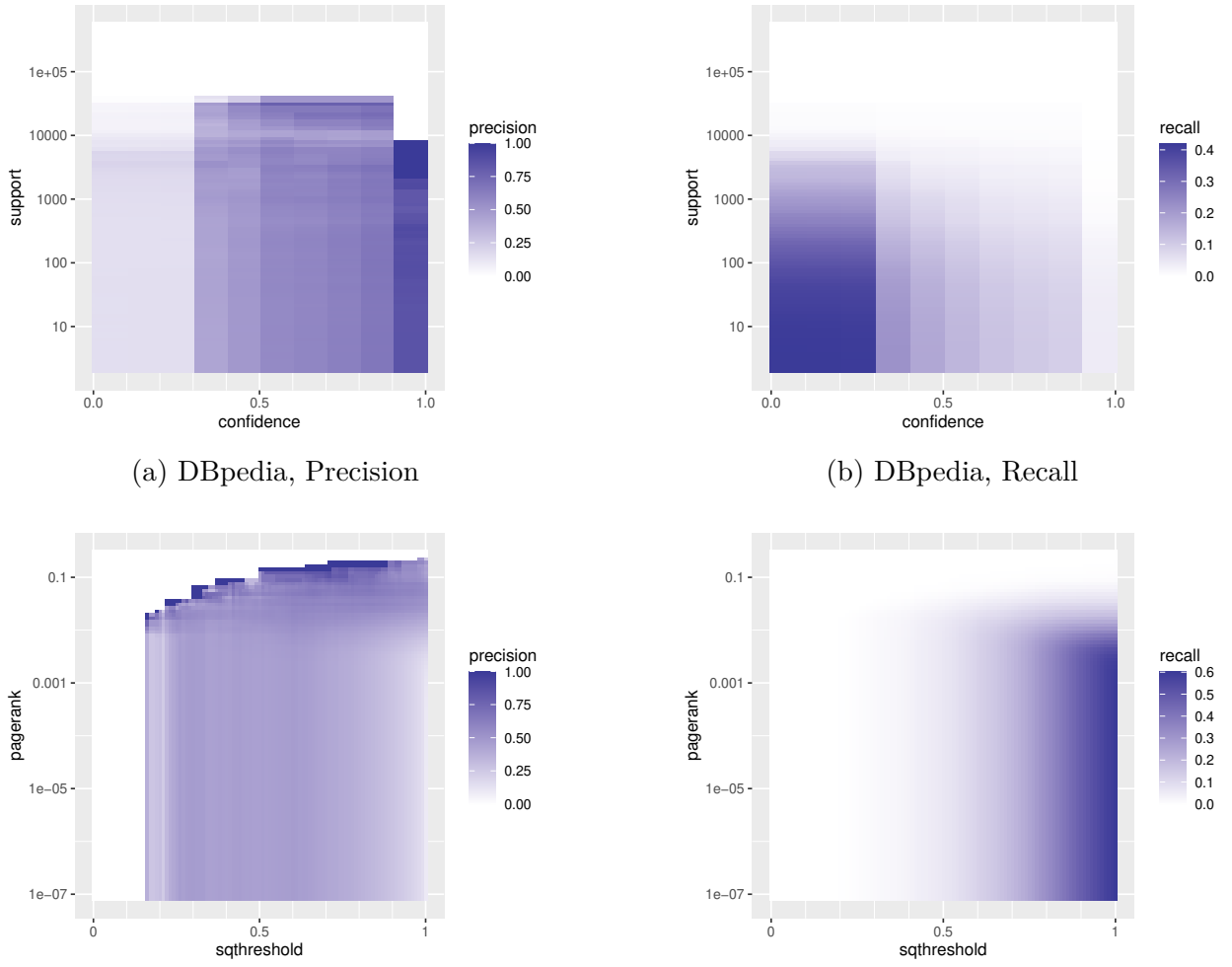
Secondary Numeric Parameters (DBpedia/support and JSI/pageRank). Both DBpedia and JSI have a secondary numeric parameter, respectively *support* and *pageRank*, that can further influence the balance between precision and recall. To visualize the effect of these parameters, Figure 3.6 shows their effect, combined with the confidence threshold, on precision and recall. When the wikifier did not return any Wikipedia article for all Stack Overflow posts, we defined the precision (typically undefined) to be 0. The axes range from the lowest to the highest value that impacts the results.

In the case of DBpedia (Figures 3.6a and 3.6b), we observe that applying a minimum support threshold has little impact on the precision, but gradually reduces the recall. Around a minimum support value of 10 000, the precision slightly increases, but at the cost of near-zero recall. Similarly, for the highest confidence level, increasing the support threshold slightly increases the precision, which reaches almost 100%, but again leads to a very low recall. A possible explanation for the ineffectiveness of support to increase precision is that the distribution of the support values of computing-specific articles, which are the focus of this study, is too similar to the distribution of more general domain articles to be an effective discriminating factor.

The case of JSI differs. Below 0.001, the minimum pageRank threshold barely affects the precision and recall. However, between 0.1 and 0.001, increasing the pageRank threshold leads to the expected increase in precision, with a corresponding decrease in recall. This suggests that some combination of minimal pageRank and pageRankSqThreshold could allow JSI to achieve precision levels similar to the other tools, without entirely sacrificing recall.

Figure 3.7 gives a different view of these phenomena by showing the effect of the secondary parameters on the precision-recall curves from Figure 3.5. To keep the graphs readable, only the most interesting portion of the secondary parameter range is shown, rather than the same range as in Figure 3.6.

DBpedia's curves strengthen the interpretation from the heatmaps. Increasing the minimum support brings the curve more to the left (i.e., decreases the recall), without considerably lifting it up (i.e., increasing precision). In the case of JSI, increasing the



(a) DBpedia, Precision

(b) DBpedia, Recall

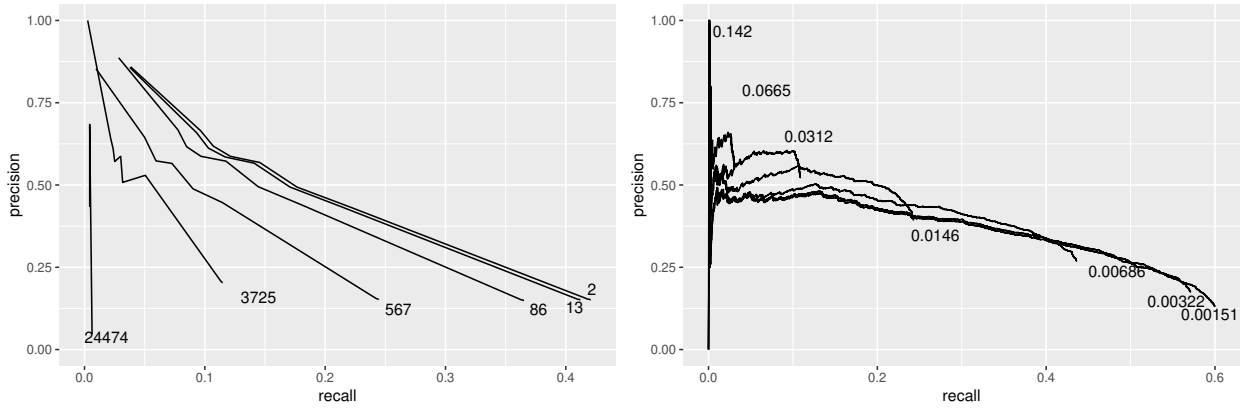
(c) JSI, Precision

(d) JSI, Recall

Figure 3.6: Impact of the Main and Secondary Numeric Parameters on the Precision and Recall of DBpedia and JSI

pageRank threshold moves the curve closer to the left and up. This makes JSI a very flexible wikifier, that can adapt to various needs. However, comparing Figure 3.7b with Figure 3.5, the gain in precision is generally not worth the loss in recall, as other tools can achieve higher recall for similar levels of precision.

Findings: A higher threshold for DBpedia's *support* parameter does not increase precision in most cases. We suggest keeping this value to 0.



(a) DBpedia (secondary: *support*)

(b) JSI (secondary: *pageRank*)

Figure 3.7: Precision-Recall curves of DBpedia and JSI for various values, equidistant on a logarithmic scale, of their secondary numeric parameter. The numeric labels on the graph indicate the parameter value associated with each curve.

Findings: Using a minimum *pageRank* threshold for JSI, in addition to *pageRankSqThreshold*, makes JSI a very flexible wikifier. However, when optimized for precision (non zero *pageRank* threshold), JSI is still less precise than other wikifiers. We suggest either using JSI with a *pageRank* minimum of 0, or other wikifiers.

Babelfy (MCS on or off). Because Babelfy's MCS option is a fallback strategy that links unmatched mentions to their most common sense, they do not affect results previously matched by the main algorithm. Furthermore, because MCS matches do not have an associated confidence score, it is impossible to filter among them: a user must either take all or them, or none.

With MCS disabled, and with the lowest value for the confidence threshold, Babelfy identified 2842 articles for the 500 posts, with a precision of 40% and a recall of 37%. Enabling MCS adds an additional 2599 articles,[33] nearly doubling the number of articles. However, the precision of the MCS articles is only 5.9%. This means that the large decrease in precision (40% to 23%) is balanced by only a small increase in recall (37% to 42%). Furthermore, with MCS enabled, increasing the confidence threshold will actually decrease precision, which will converge towards the precision of only the MCS matches, 5.9% (with a confidence threshold

[33]An additional 15 previously identified articles become MCS matches, due to unpredictable factors of the wikification algorithms.

of 1, almost all articles are identified by the MCS strategy). Therefore, in most cases, the slight increase from the MCS strategy is not worth the loss in recall.

Findings: Babelfy's MCS option has very low precision. Because there is no way to distinguish between the MCS matches, we suggest not to use this option.

Ambiverse (Stanford or KnowNER recognition). Ambiverse includes two components to perform entity recognition: a Stanford NER tool and the KnowNER tool by the Ambiverse authors. We observed that KnowNER leads to a better performance, but the difference is minimal: for any given recall value, the precision with KnowNER is between 0.3 and 2.6 percentage points above that of Stanford NER. The superiority of KnowNER is consistent over the range of recall. Both components achieve a maximum recall just under 47%.

Findings: Ambiverse's KnowNER component performs slightly, but consistently, better than the Stanford NER alternative.

WAT (OpenNLP or Lucene tokenizer). WAT offers a choice between two components for the tokenization step: OpenNLP, which is the default option, and Lucene, which is described as better suited for ill-formed text. Figure 3.8 shows the precision-recall curves of both tokenizers. Contrary to the choice of NER component for Ambiverse, the choice of tokenizer has a noticeable impact on the peformance of WAT. Lucene allows WAT to achieve much greater recall, but quickly drops in precision, whereas OpenNLP generally retains a higher precision, but at a much lower recall. Therefore, depending on which of the precision or recall is preferred, both tokenizers can be relevant. However, when compared to other wikifiers, Lucene's gain in recall is not worth its loss in precision. Therefore, in most cases, either OpenNLP is more appropriate, or another wikifier is.

Findings: WAT's OpenNLP tokenizer achieves higher precision, whereas the Lucene tokenizer leads to higher recall. However, the gain in recall is less impressive when compared to other wikifiers. We suggest using WAT with OpenNLP, or other wikifiers.

3.3.3 Correlation Between Wikifiers

Although all six wikifiers claim to solve the same general task, wikification, it is not clear whether all approaches converge towards the same results, or the results of one can cover the

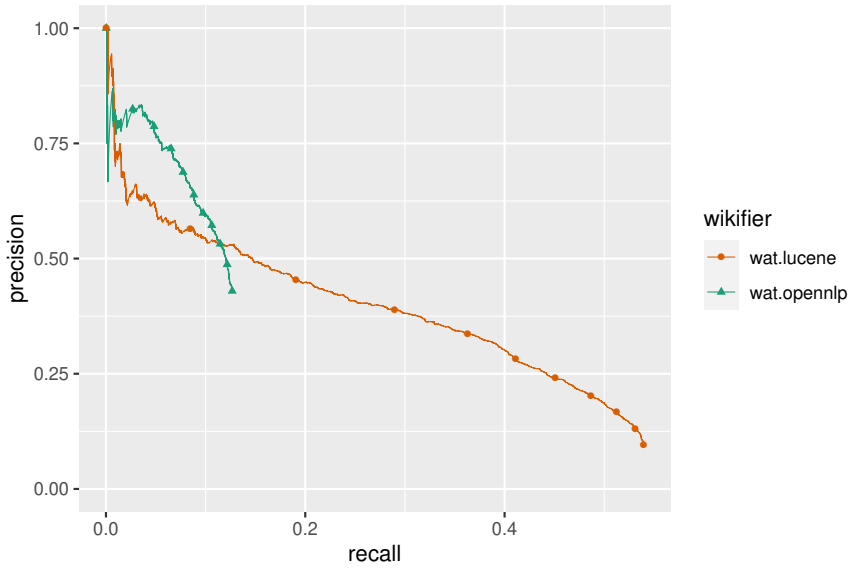


Figure 3.8: Precision-Recall Curves for WAT with Two Different Tokenizers: Lucene and OpenNLP

Table 3.5: Overlap Between the Correct Results of Each Wikifier

Ambiverse	**Babelfy**	**DBpedia**	**Illinois**	**JSI**	**WAT**	
1423	703 (533)	586 (605)	353 (370)	965 (867)	206 (179)	**Ambiverse**
–	1123	596 (477)	432 (292)	824 (685)	252 (177)	**Babelfy**
–	–	1274	475 (332)	898 (777)	261 (161)	**DBpedia**
–	–	–	780	529 (475)	229 (98)	**Illinois**
–	–	–	–	1827	324 (230)	**JSI**
–	–	–	–	–	378	**WAT**

Parentheses show the expected overlap assuming a uniform random sampling.

blind spots of another. Understanding such correlation between the results of wikifiers can help to improve each individual approach, as well as develop ensemble methods to mitigate the blind spots of different wikifiers.

Table 3.5 presents, for each pair of wikifiers, the number of articles correctly identified by both wikifiers (with a confidence threshold of 0), as well as the *expected* number of articles, in parentheses. The expected value assumes that all wikifiers select independently and uniformly randomly articles from the set of correct matches, keeping the recall constant. For example, Ambiverse correctly identified 1423 of the 2997 articles (recall = 47.48%), and Babelfy correctly identified 1123 articles (recall = 37.47%). Therefore, assuming independence, the number of articles identified by both Ambiverse and Babelfy should be $47.48\% \times 37.47\% \times 2997 = 533$.

Table 3.6: Kendall's τ_b Correlation Coefficient and 0.95-Level Confidence Interval Computed over the Overlap of each Pair of Wikifers

Babelfy	**DBpedia**	**Illinois**	**JSI**	**WAT**	
0.43 (±0.05)	0.37 (±0.05)	0.44 (±0.06)	0.19 (±0.04)	-0.21 (±0.10)	**Ambiverse**
–	0.33 (±0.06)	0.36 (±0.07)	0.22 (±0.05)	-0.07 (±0.09)	**Babelfy**
–	–	0.04 (±0.07)	0.36 (±0.04)	0.01 (±0.09)	**DBpedia**
–	–	–	0.09 (±0.06)	0.02 (±0.09)	**Illinois**
–	–	–	–	-0.06 (±0.07)	**JSI**

Typically, one could expect the actual overlap between wikifier results to be greater than the expected overlap under the assumption of independence, because they attempt to solve the same task. However, Ambiverse with both DBpedia and Illinois actually has a smaller overlap than expected. This could suggest that Ambiverse's results complement those of DBpedia and Illinois, and that combining the ideas from Ambiverse and Illinois or DBpedia into a mixed approach would result in a higher improvement than combining other approaches. For other wikifiers, with an overlap larger than the expected value, a simple ensemble approach, such as majority voting, can improve individual performances.

To further understand the correlation between wikifiers, Table 3.6 shows Kendall's τ_b statistic between each pair of wikifiers, computed over the overlap between the results of the two wikifiers.[34] Kendall's τ coefficient is a non parametric rank correlation statistic that estimates the probability that the two wikifiers will agree on which of two random articles have a higher confidence score [104]. The probability value is rescaled from $[0, 1]$ to the $[-1, 1]$ range to produce a correlation score. Therefore, $\tau = 0$ indicates an agreement probability of 50% (no correlation), $\tau = -1$ indicates a 0% probability of agreement, or 100% of disagreement (perfect negative correlation), and $\tau = 0.43$, as for Ambiverse and Babelfy, indicates a probability of 72% of agreement.[35]

Table 3.6 provides a more detailed insight into the complementarity of different wikifiers. WAT shows a peculiar behavior, with correlation scores hovering around zero for most wikifiers, except for Ambiverse, where the correlation is negative. This surprising result

[34]Instead of restricting the correlation to the overlap, another possible approach would have been to use all articles, and assign a confidence of 0 to articles not found by wikifiers. This approach, however, is sensible to noise due to differences in the knowledge bases (and their version) used to train the wikifiers. Using this approach, we observed correlation scores all near zero. Therefore, we present the more useful results using only the overlaps.

[35]The τ_b variant of the statistics is explicitly tuned to account for ties, which is especially important for the discretized confidence scores of DBpedia's results.

indicates that among articles correctly identified by both Ambiverse and WAT, those with a high confidence Ambiverse score tend to have a low WAT score, and vice versa.

Apart from WAT, almost all pairs of wikifiers have a significant positive correlation, ranging from 0.19 to 0.44. The only exception is Illinois, with DBpedia and JSI, which has a near-zero correlation. In particular, Ambiverse with DBpedia and Illinois show a strong positive correlation, despite having a smaller than expected overlap. These results are encouraging, as they suggest that different wikifiers (with the exception of WAT) actually have a similar objective. Therefore, choosing one over another is less likely to introduce unwanted biases.

Findings: The confidence score used by all wikifiers is positively correlated, with the notable exception of WAT. Confidence scores between Ambiverse and WAT are even negatively correlated, which is a surprising result.

3.3.4 Validated List of Computing Concepts

The list of validated computing-related Wikipedia titles, obtained as a result of the annotation task, is an important contribution of this study. Because we generated the list using six state-of-the-art wikifiers configured to optimize recall, with a representative sample of a large programming forum, it represents the current range of Wikipedia articles on computing and can serve as a basis to precisely identify the full extent of computing articles.

We studied two approaches to use this list as a filter to improve the output of wikifiers. The first approach uses the validated computing titles as an inclusion list, and rejects other articles. The second approach instead uses negative examples (i.e., articles marked as "incorrect" by the annotators) as an exclusion list of articles to reject. To allow others to use the same strategies to improve the performance of wikifiers, we distribute the complete annotated dataset in our online appendix (see Appendix A).

Computing Titles as an Inclusion List. The first strategy is to remove from the output of wikifiers all articles not on an inclusion list, i.e., the validated list of computing-related articles. In the ideal case, where the inclusion list contains all computing articles of Wikipedia, this strategy will increase precision with no impact on recall, because it filters out only non-computing, thus incorrect, articles.

To evaluate this strategy, we used the list of 1098 articles to filter the output of all six wikifiers. Figure 3.9 shows the result of this strategy on the precision-recall curves. The

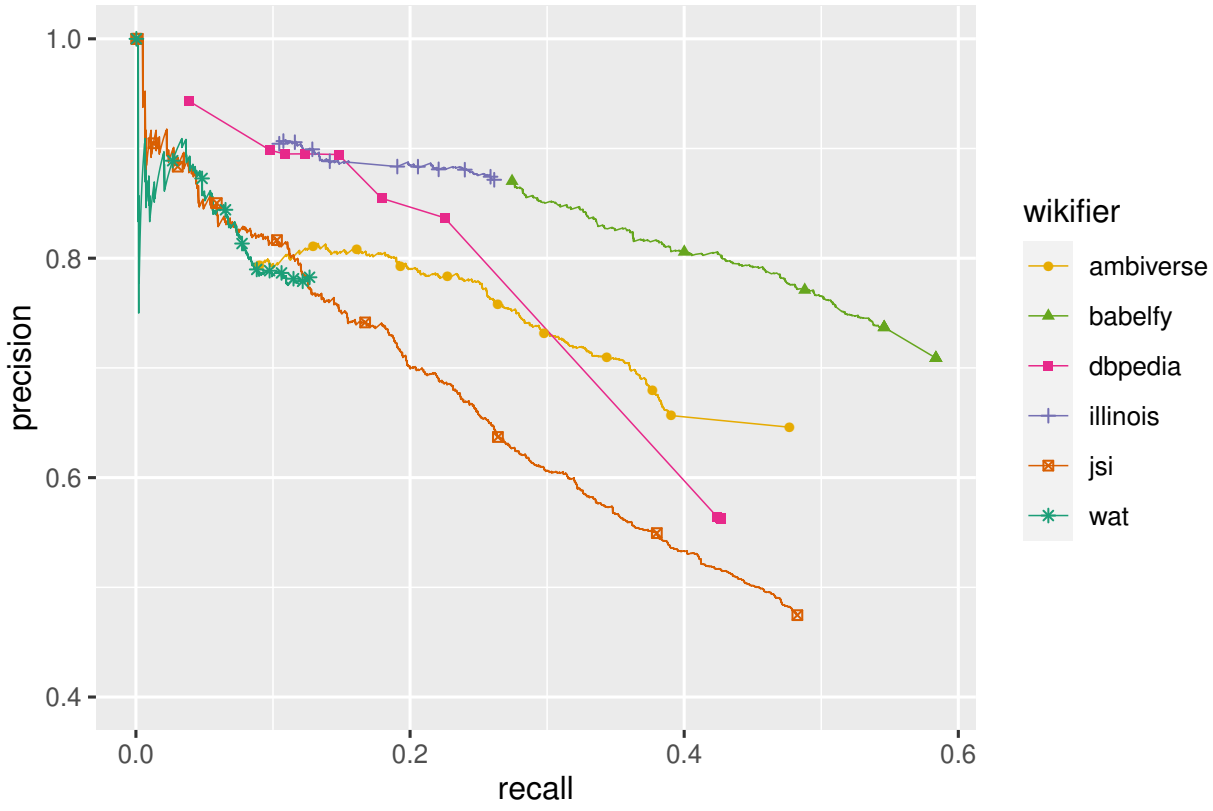


Figure 3.9: Precision-Recall Curves of All Six Wikifiers with the Inclusion List Strategy to Improve Precision[36]

inclusion list considerably augments the precision.[36] For example, for a recall threshold of 20%, the precision of all wikifiers goes from less than 50% to over 74% (except for WAT, which does not achieve such recall). Nevertheless, there is still a non trivial proportion of computing-related false positives, especially for low confidence values (e.g., to achieve 60% recall, JSI's precision drop to 47%). For example, although SOURCE CODE is often correctly linked to the mentions *source* and *code*, it can be linked to misleading terms, for example when *source* indicates the provenance of information in a post.[37]

A limitation of this approach is the need for an extensive inclusion list. Although our list of 1098 articles is a good start, it may not contain articles about rarer computing concepts or technologies. For example, we took a random sample of 30 articles marked as "incorrect" only once, and found four computing articles among them: MESSAGE AUTHENTICATION CODE (matched to HTML codes), 32-BIT (matched to the number 32), ARITHMETIC OVERFLOW

[36]Note that the y axis in Figure 3.9 differs from Figure 3.5, to improve readability.
[37]https://stackoverflow.com/questions/16516936

(matched to HTML content overflow), and Graphics Environment Manager (matched to Ruby's gem command). To avoid the possibility of rejecting such articles, users can combine two configurations (or two wikifiers): one optimized for recall that uses the inclusion list, and the other optimized for precision which does not use the inclusion list.

Findings: Using the list of articles as an inclusion list showed a precision improvement, reaching values between 74% and 89%. However, computing-related false positives are still present in non-negligible ratios for configurations that optimize recall.

Negative Examples as an Exclusion List. To circumvent the limitations of the inclusion list strategy, another strategy is to only reject articles from an explicit exclusion list. An extensive list of all non-computing articles would be equivalent to an extensive inclusion list. However, the exclusion list can focus on articles that are especially problematic for a wikifier, i.e., articles that are most often false positives. The objective of this strategy is to maximize the increase of precision by excluding common irrelevant terms (e.g., Canning, which is often matched to the modal auxiliary verb *can*) and artifacts of the training phase (e.g., Burmese language[38] and On Your Toes[39]).

In addition to the list of 1098 computing articles, the annotation task generated a set of 10 854 distinct articles marked as "incorrect" at least once. Articles that were rejected the most often (e.g., 781 articles were rejected at least ten times) can serve as a basis to create the exclusion list. However, even some of these most often rejected articles are related to computing. For example, although This (computer programming) is a computing concept, WAT with the Lucene tokenizer often associates it with the demonstrative pronoun *this*, creating many false positives. Including these articles in the exclusion list can improve precision at the cost of known blind spots, but this trade-off must be tailored to each wikifier (e.g., associating This (computer programming) is not a common issue for other wikifiers than WAT).

[38]The ISO two-letter code for the Burmese language is "my", a homograph of the possessive determiner, which appears in many posts.

[39]This musical is often linked to the word *your*, an obvious incorrect artifact from the training phase.

Findings: An exclusion list does not prevent rarer articles from being detected. It optimizes the gain in precision by targeting the weaknesses of wikifiers. However, some computing articles are often confounded with other terms, and thus often marked as "incorrect". With a careful configuration, the exclusion list strategy can optimize the gain in precision at the cost of known blind spots for such articles.

3.3.5 Discussion

Overall, the unpredictable relative performance of the six wikifiers confirm the difficulty of selecting an optimal wikifier and its configuration for a given usage scenario. The wikifiers vary considerably in the range of precision and recall values they can achieve for different values of their confidence parameter. For example, modifying the confidence parameter of Ambiverse will have a large impact on its recall, but not on its precision, while the inverse is true for WAT.

We observed, however, that there are few benefits to modifying the default values of the additional parameters of the wikifiers. In most cases, the default value already provides the optimal performance. Thus, developers and researchers looking for an optimal configuration can focus on the primary confidence parameter. To this end, Figure 3.5 provides a useful starting point to find the optimal wikifier for a specific task.

Nevertheless, the low precision and recall limit the wikifiers' usefulness in concrete applications: To identify at least one out of every five related concepts (i.e., at least 20% recall), no wikifier was able to produce more true positives than false positives (i.e., at least 50% precision). This observation motivates the development of specialized techniques to adapt wikification approaches to the software domain. In particular, the use of a list of accepted software-related concepts to filter the output of wikifiers showed a promising improvement to the precision, with a minimal impact on the recall.

3.3.6 Threats to Validity

Internal Validity. The dynamic nature of Wikipedia affects the internal validity of the results. Because Wikipedia is in constant evolution, identifying an article by title is ambiguous, as it can refer to multiple versions. This evolution impacts wikifiers trained with older versions of Wikipedia, or with other knowledge bases which themselves rely on old Wikipedia versions. In addition to being oblivious to more recent articles, these wikifiers may refer to articles that were significantly changed. As an example, the article Database management system was

merged in 2013 with the article Database.[40] Therefore, a wikifier trained on a Wikipedia version older than 2013 will consider Database and Database management system as two different articles, whereas more recent wikifiers will only consider them as one. This divergence, however, is inevitable, as researchers who may wish to use a wikifier without having to re-train it will be similarly affected.

By the same argument, the evolution of Stack Overflow posts between the wikification and the completion of the annotation task causes another threat. To mitigate it, all data used in this study was only a few months old, greatly reducing the threat of divergence.

External Validity. Threats to external validity depend on the level of generalization considered. The first level of generalization is from the sample of 500 posts to all Stack Overflow posts. The simple random sample of posts is sufficient to support a statistical generalization of proportions *computed over posts* to the whole population within an error of 0.05 with 95% confidence. However, most of the results of this study are proportions *computed over article–post pairs*. This situation benefits us, as proportions of pairs represent a much larger number of annotations (41 124 article–post pairs in total), but statistical generalization from these proportions is more complex, because the sample of pairs is not random. Despite the lack of statistical generalization, the unbiased sampling of posts and the absence of evident imbalance in the wikifications (e.g., there was not a small subset of posts that received all correct wikifications) give sufficient confidence in the representativeness of the findings. Furthermore, the objective of this study is not to identify the best performing wikifier, but to understand their relative performance to help potential users make informed decision on which wikifier to use. In such context, small performance differences, statistically significant or not, should not influence the choice of a wikifier the way larger differences would.

We performed a sensitivity analysis to further assess the threat of spurious results. We generated 300 trimmed samples by removing 5%, 10%, and 20% of the 500 posts, 100 times for each proportion. For each trimmed sample, we computed the precision and recall of all wikifiers, for 21 confidence thresholds ranging from 0 to 1, in increments of 0.05. We then compared those precision and recall values to the precision and recall computed on the full sample, for the same confidence thresholds. For all six wikifiers, the maximum variation in recall, for all three proportions, was 1.8%. In terms of precision, the maximum variation was of 4.2% for Ambiverse, Babelfy, DBpedia, and Illinois for all three proportions. We

[40] https://en.wikipedia.org/w/index.php?title=Database_management_system&diff=544579037&oldid=544577010

observed larger variations in precision for JSI and WAT when the confidence threshold was near 1, which can be expected, as very few results are returned (and thus, the effect of one additional true or false positive is much larger). However, for lower thresholds (0.70 or less for JSI and 0.95 or less for WAT), the variation is similar to that of the other tools (below 7.5%). Considering that these values represent the *maximum variation* over 300 independent perturbations of up to 20% of the sample, the small observed variations suggest that the results can reliably be generalized to the population of Stack Overflow posts within a small margin of error.

The study design supports an analytic generalization from the sampled population, i.e., Stack Overflow posts, to the concept of a "software resource". In contrast to other types of documents, a software resource contains a mix of natural language and code, either in blocks or embedded in the main text, software-specific jargon, and references to a fast growing list of technologies. The considerable variance of posts in a number of dimensions, such as the formality and quality of the language, the length of a post, the sub-area of computing, and the intent of the post (e.g., question, description, explanation, step-by-step guide), allows this analytic generalization. Nevertheless, because this study is the first to focus on the wikification of software resources, further replication studies with resources from other sources are required to strengthen this generalization. In particular, Stack Overflow posts exhibit a number of idiosyncrasies. For example, posts edited to add content often include the new content under an "Edit" header, and questions sometimes end with a "Thank you", which wikifiers would link to the Wikipedia articles Editing and Gratitude, respectively. Although such phrases are unlikely to appear in other types of software resources, it is reasonable to assume that any source has its own idiosyncrasies, and wikifiers should be able to ignore them.

Construct Validity. Construct validity concerns the relation between the metrics used (precision and recall) and the constructs under study (wikifier performance). The two metrics are commonly used in software engineering research, and both have an intuitive interpretation that allows the reader to understand what exactly is measured (as opposed to, e.g., F1 score). The more important threat to construct validity comes from the formulation of the annotation task. The details of the task, discussed in Section 3.2.8, may diverge from others' interpretation of wikification, especially regarding the rejection of relevant, but non-computing articles. This decision was however necessary to mitigate the subjectivity of the task and the influence of irrelevant results.

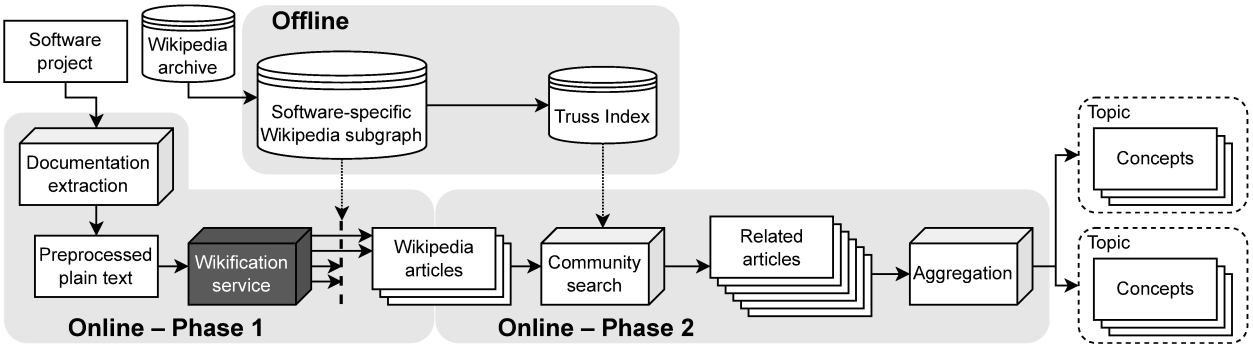


Figure 3.10: Scode Approach to Identify a Set of Concepts Grouped by Topic from a Software Project

Conclusion Validity. The findings of this study suggest different strategies to use wikifiers. These suggestions are backed by empirical evidence, but there are many other factors that can influence the choice of a wikification procedure, such as the time budget, the number of documents to wikify, and the amount of human effort that can be devoted to the task (e.g., to experiment between different configurations or manually validate some of the output). To mitigate this threat, we discussed findings from all wikifiers, not just the best performing ones.

3.4 Concept Identification Approach

Our approach to identify relevant Wikipedia-based concepts works in two on-line phases, summarized in Figure 3.10. The first phase identifies concepts that are *explicitly* mentioned in the documentation using a *wikification* service (Section 3.4.2). The second phase uses explicit concepts as seeds to identify additional *implicit* concepts related to the same domain. For this phase, we implemented a set of *community search* algorithms (Section 3.4.3). The outcome of the community search algorithms is also used to group concepts into topics, each characterized by a single representative concept.

These two phases rely on data processed once during a third, off-line phase. This phase consists of preparing the Wikipedia data archive to extract a graph of computing-specific articles (Section 3.4.1). We also precompute properties of this graph to support the community search algorithms.

3.4.1 Off-Line Preparation

Both on-line phases of Scode rely on a graph of concepts derived from wikilinks, i.e., hyperlinks between Wikipedia articles. The edges between concepts in this graph provide the basis to identify related implicit concepts with the community search algorithms. The concepts included in the graph also act as an inclusion list to filter the explicit concepts identified by the wikification service. Thus, the accuracy with which the graph models the computing domain plays an important role in improving the accuracy of Scode.

Although it would be possible to use the entire wikilink graph, this graph is extremely large and highly connected. We used a semi-automated approach to extract this software-specific subgraph from the April 2020 archive of the English-language Wikipedia. We only retained articles in the main namespace of Wikipedia, i.e., excluding pages such as categories and article discussions. We also retained only wikilinks that are inserted directly in the code of an article, i.e., not via a template. This filter discriminates wikilinks that are intentionally chosen by Wikipedia editors to relate two specific subjects from those added automatically by, e.g., *navboxes* and *sidebars*.[41]

Wikipedia contains *redirect pages*, which automatically redirect the reader to another target article when accessed. These pages are often used to add alternative titles to an article. We removed those pages from the set of nodes, but resolved and retained wikilinks passing through these redirect pages. That is, if an article A links to a redirect page R, whose target is B, we add an edge from A to B in the graph. Finally, we discarded *disambiguation pages*[42] and *set index articles*,[43] which are only used to list the different senses of polysemous terms.

After applying these filters, we obtained a pruned graph with 5.65 million nodes and 129 million edges (average node degree of 45.7). We considered all edges as undirected and unweighted, which is consistent with the interpretation that a wikilink indicates an unquantifiable and reciprocal relatedness relation between the source and target concepts.

We further filtered the set of articles using a systematic manual procedure to retain only those related to computer science and software technologies (Figure 3.11). The manual procedure requires as input the automatically pruned Wikipedia link graph W and a seed set C of candidate articles potentially related to computing. To produce the initial candidates, we gathered all articles listed in four Wikipedia navigation pages containing notable computing-

[41] https://en.wikipedia.org/wiki/WP:NAVBOX
[42] https://en.wikipedia.org/wiki/WP:DAB
[43] https://en.wikipedia.org/wiki/WP:SIA

Input: W: Wikipedia link graph
Input: C: Seed candidate set
Output: R: Subset of computing-related Wikipedia articles

```
1:  D ← ∅        // unrelated articles to discard
2:  while C ≠ ∅ do
3:      c ← arbitrary element from C
4:      C ← C \ {c}
5:      if c is related to computing then
6:          R ← {c} ∪ R
7:          for all n ∈ neighbors(c) \ (C ∪ R ∪ D) do
8:              N ← neighbors(n)
9:              if |N ∩ R| ≥ |N \ R| then
10:                 C ← {n} ∪ C
11:             end if
12:         end for
13:     else
14:         D ← {c} ∪ D
15:     end if
16: end while
```

Figure 3.11: Identification of Computing-related Articles

related articles,[44] as well as the 1098 articles identified as *correct* at least once during our comparison of wikifiers (Section 3.2.8). The resulting *candidate* set C contained 2045 distinct articles covering different areas of computing.

For each candidate, an annotator scanned the content of the article to determine whether it was related to computing (line 5), in which case the candidate was added to a *related* set R (line 6). When adding an article to R, each of its neighbors not yet considered (line 7) and that have more neighbors inside than outside R (line 9) became a candidate (line 10).[45] Rejected articles were added to a *discard* set D to avoid considering them again as candidates (line 14). As the set of candidates is updated every time a candidate is accepted, we could not divide the annotation task into parallel subsets annotated independently by each author. Thus, the first author performed the entire task, accepting 6746 articles and rejecting 2811 others. To measure the subjectivity of the annotation task, the second author independently annotated a sample of 800 articles taken at random with equal probability from the final content of R and D. As this exercise only served to assess the subjectivity

[44]The articles are *Glossary of computer science*, *Index of object-oriented programming articles*, *Index of computing articles*, and *Index of software engineering articles*.

[45]This criterion on line 9 prevents the set of candidates from growing too quickly and rendering the manual validation impractical.

of the reliability of the first annotator's results, we did not resolve conflicts. We observed a substantial [112] inter-rater agreement (Cohen's $\kappa = 0.71$ [52]), suggesting that the selected articles are a reliable representation of the computing domain on Wikipedia.

The resulting graph of 6746 computing articles contains 102 854 edges (average node degree of 30.5). This graph excludes general-domain concepts, such as Language and Mind. It also excludes specific end-user applications (e.g., names of video games), learning resources on computing (e.g., textbooks), and low-level hardware technologies (e.g., models of microprocessors). It includes general computing concepts and domains (e.g., Data parallelism, Artificial intelligence), programming languages (e.g., C++), and development technologies (e.g., IntelliJ IDEA, Spring Framework). We make this graph, as well as the annotation guide and annotations, publicly available in our on-line appendix.

In addition to extracting a relevant subgraph of concepts from the Wikipedia link graph, the off-line phase involves preparing a *truss index* to support one of the community search algorithms. We reimplemented the procedure to compute this index as described by Akbas and Zhao as part of their community search technique [6]. We define trusses and describe the use of the truss index in Section 3.4.3, together with the other algorithms we used to expand concepts into topics.

3.4.2 Explicit Concept Identification

Scode uses a wikification service to identify explicit concepts in documentation. To design this phase, we relied on the results of a prior study in which we compared six state-of-the-art wikification tools (Section 3.3). We found that the performance of all tools decreased when applied to software documents, typically due to technical terms being confused with their domain-general sense (e.g., Tree resolved as the type of plant). Using an inclusion list of computing-specific concepts is an effective strategy to counter this limitation (see Section 3.3.4).

We chose the Babelfy tool [149] for Scode. Babelfy is an on-line wikification service that achieved good precision for reasonable recall levels, relatively to the other tools. To produce its natural language input, Scode aggregates all Javadoc comments in a source file, keeping them in their original order within the file. Each comment consists of an initial description of the type or method being documented, followed by a list of tag-fragment pairs. For example, the documentation of a method can start with its general purpose, followed by fragments describing each parameter, marked by the `@param` tag. All elements can use HTML and

in-line Javadoc-specific syntax. Before aggregating these comments, Scode removes the HTML and Javadoc syntax, and excludes any comment description or fragment that is too short.[46] Excluding short descriptions and fragments prevents the input from containing incomplete phrases (e.g., *"Used for nontrivial settings upgrade"*) and irrelevant information (e.g., the authors of a class).

Like most wikification tools, Babelfy is more precise when processing large texts, because text fragments of only a few words lack sufficient context to disambiguate the sense of polysemous words. Thus, Scode aggregates all the comments of a project into a single input, to provide as much context as possible.[47] The order in which Scode aggregates the documentation from different files is arbitrary.[48]

Babelfy returns a list of concepts, identified as entries in the BabelNet knowledge base [165]. BabelNet entries can be mapped to corresponding Wikipedia articles. Scode removes from the output any concept that is not in our curated list of computing-specific Wikipedia articles. This filter can produce false negatives, for example due to non-computing concepts related to the problem domain of the project. However, the negative impact of these missing concepts is outweighed by the larger increase in precision observed in our prior work when using an inclusion list. The explicit concepts output by this phase seed the search for additional implicit concepts in the next phase.

3.4.3 Implicit Concept and Topic Identification

Edges in the computing-specific Wikipedia graph indicate relatedness between articles, and thus can be used to find implicit concepts. However, despite the heuristics described in Section 3.4.1 to reduce the number of edges, the graph remains densely connected. Trivial heuristics, such as taking all concepts within an arbitrary distance of the seed concepts, produce too many concepts.

To circumvent this issue, we used community search techniques to identify a practical number of concepts. Informed by Fang et al.'s survey of community search algorithms for large graphs [68], we reimplemented three algorithms to identify implicit concepts, as shown in Figure 3.12.

[46] We used an arbitrary minimum of ten words.

[47] Due to a limitation of the API, Scode makes several requests if the aggregated comments contain more than 10 000 characters. Scode avoids splitting comments to the extent possible.

[48] Wikification techniques are noisy and sensitive to the order of their input. In an ancillary experiment, we confirmed this sensitivity, but found no obvious way for the input order to systematically improve the quality of the result. Babelfy introduced a similar amount of noise for the arbitrary order of Scode as for other random permutations.

Input: Set C of explicit concepts
Output: Set of topics T

```
1: M ← empty map
2: R ← ∅
3: for all concept c ∈ C do
4:     T ← LargeTruss(c)        // T is itself a set of concepts
5:     T ← ReduceTruss(T)
6:     if |T| ⩾ 100 then
7:         T ← ReduceECC(T)
8:     end if
9:     r ← Representative(T)
10:    if r ∈ keys(M) then
11:        T ← T ∪ M[r]
12:        R ← R ∪ {r}
13:    end if
14:    M[r] ← T
15: end for
16: for all r ∈ R do
17:    M[r] ← ReduceTruss(M[r])
18: end for
19: return values(M)
```

Figure 3.12: Implicit Concept Identification Procedure

The procedure starts by identifying a separate community around each explicit concept c. An initial community (or topic) T is generated using Akbas and Zhao's algorithm [6] to find the *largest k-truss* around c, for the highest possible value of k (function LargeTruss of Figure 3.12). Within a graph, a k-truss is a subgraph in which any two neighbors (i.e., nodes connected by an edge) share at least $k-2$ other common neighbors. Akbas and Zhao's algorithm is particularly efficient for repeated queries over a large graph, as it precomputes the key information about the potential trusses and stores it in a *truss index*.

As we favor smaller communities, Scode applies Huang et al.'s algorithm [100] to reduce the size of the k-truss while maintaining the query concept c and the value of k (function ReduceTruss of Figure 3.12). Their algorithm approximates the subset of the initial k-truss for which the maximum distance from the query concept c to any other concept is as small as possible.

We observed a bimodal distribution in the number of concepts per topic. In many cases, the reduced topic has fewer than 25 concepts. However, when the initial query concept is a popular computing concept, such as JSON or SQL, this minimal community still

contains several hundred concepts. Scode further reduces the size of any community with over 100 concepts by relying on an alternative definition of connectivity for communities: *k-edge-connected-components*, or k-ECC. A k-ECC is a subgraph that requires at least k edges to be removed for the graph to become disconnected (i.e., it is no longer possible to find a path between any two nodes). For example, a set of k nodes that are all pairwise connected is a (k-1)-ECC. This connectivity requirement is looser than k-trusses, which allows to reduce the size of a (k-1)-ECC derived from a k-truss while maintaining the value of k. We used Hu et al.'s algorithms [97] to perform this reduction for large k-trusses (function REDUCEECC of Figure 3.12).

The algorithms described so far generate a separate community for each explicit concept. The next step is to merge communities that describe similar topics. As the size of communities varies considerably, many intuitive heuristics (e.g., merging communities with a large overlap) have a detrimental effect on the output. For example, they can void the information captured by a small specialized topic by merging it with a large generic community. Trivial merging heuristics can also lead to arbitrary topics that depend on the order in which the communities are merged.

We implemented the community aggregation step by seeking a *representative* concept for each community. Following the insights from Lizorkin et al. [124], we chose the representative as the concept with the highest *pagerank* value [32] among the community (function REPRESENTATIVE of Figure 3.12).[49] Scode merges all communities that have the same representative into a single community, and applies the k-truss reduction algorithm once more on the new communities to further reduce their size.

This final set of communities forms the outcome of Scode. Each community represents a single topic of related concepts, identified by one representative concept.

3.4.4 Implementing the Sample Application

We implemented the application described in Section 3.1 to demonstrate the usefulness of Scode in a practical scenario.

After collecting a set of concepts grouped by topic, the application separates small topics (i.e., containing at most 25 concepts) from larger ones. The summary badge inserted at the top of the README file indicates the number of small topics. For each small topic, the application generates one badge using the name of the topic's representative concept

[49]The representative concept is typically more general than the explicit concept. In our evaluation with Java classes, we observed that 15% of topics have an explicit concept as its representative.

as the value of the badge. The color of the badge is based on the representative concept: we automatically assigned to each concept in the computing-related subset of Wikipedia a unique color, giving similar colors to similar concepts based on a hierarchical clustering of the Wikipedia graph. The application uses the Shields.io service to generate the badge image with the desired name (*scode*), value, and color.

The application also creates a new file to describe the topics in more detail. In this file, we add one section for each small topic, and each badge in the README file links to its corresponding section. The section body contains the explicit concepts from this topic first, as they are more likely to be of importance to the developer. Further implicit concepts are placed in a collapsible list under the explicit concepts. As each concept is a Wikipedia article, the application creates a hyperlink from each concept to the associated article.

Large topics are not included. We observed that they tend to contain less cohesive groups of general programming concepts. Hence, developers are less likely to find the extent of such topics useful, especially if they must scan hundreds of concepts. Hence, we only retain *explicit* concepts included in large topics, and list them all under a *"General Programming Concepts"* section in the new file.

3.5 Scode Evaluation

The objective of our evaluation was to understand the strengths and limitations of an approach based on wikification and community search, as represented by Scode, to identify concepts relevant to a software project. We centered this evaluation around the research questions RQ 3.4 to 3.7 presented in Section 3.1.1.

We first evaluated the end-to-end performance of Scode and compared it to two baseline techniques. To measure the performance of concept identification techniques, we used two metrics: The *precision* of a technique corresponds to the proportion of the concepts it identifies that are actually related to the project (RQ 3.4, Section 3.5.2). Conservative techniques can achieve a high precision by returning few or very specific concepts. To balance this dimension, we used the *consistency* of the identified concepts as the second metric (RQ 3.5, Section 3.5.3). A technique should consistently identify the same concept for any project related to this concept.

To gain a better understanding of the promising and limiting components of our approach, we evaluated the internal mechanisms of Scode separately (RQ 3.6): the inclusion list of computing concepts (Section 3.5.4), the wikification service (Section 3.5.5), and the community search algorithms (Section 3.5.6). Finally, as it is a major design decision for Scode, we

studied the impact of extracting concepts from documentation rather than source code (RQ 3.7, Section 3.5.7).

3.5.1 Study Design

We generated a list of concepts related to a sample of Java classes and manually validated the concepts' relatedness to their associated class. We compared Scode to two state-of-the-art techniques as baselines, following the same procedure. We computed the *precision* and *consistency* of all techniques to answer our first two research questions.

We selected as baselines tools that are publicly available and that extract concepts from the documentation of a project. The first baseline is a technique to identify concepts related to Java classes in order to build a comprehensive API knowledge graph [121, 122]. The technique relies on various natural language processing heuristics specifically tailored for API concept extraction. In contrast to Scode and the other baseline, the concepts are not intended to have a recognized meaning in an external knowledge base. For our evaluation, we extracted the concepts from the API knowledge graph, which we refer to as *PengKG*, instead of re-implementing the concept extraction technique.

We used explicit semantic analysis (ESA) [75] as the other baseline technique. ESA associates arbitrary text inputs with entries of a knowledge base, such as Wikipedia. We used the EasyESA implementation of ESA in this evaluation [41]. To produce the input for EasyESA, we used the same processed documentation as for Scode, described in Section 3.4.2.

We limited the size of the subjects in our evaluation sample, so that the evaluators could understand each subject in a reasonable amount of time. Thus, instead of using whole projects, we considered individual Java files as independent subjects. Because PengKG did not include concepts for interfaces, we only sampled files for which the top level type is a class. Hence, we refer to each subject as a Java class, with the understanding that it may include nested classes.

We randomly sampled 100 Java classes, each with equal probability, from the standard Java Class Library, version 15. We used the standard library due to its coverage of many areas and the high quality of its documentation. These classes were also part of PengKG.

Scode identified a total of 84 to 774 concepts per class (average of 459), except for five classes that were associated with no concept due to their insufficient documentation. To avoid overloading the annotators, for classes associated with more than 100 concepts, we randomly sampled 100 concepts among Scode's output for the annotation task, with each concept having an equal probability of being selected.

For each class, PengKG includes one to 42 related concepts (average of 10). We included all of them in our evaluation. For EasyESA, as it takes a number of concepts to generate as part of its input, we generated five more concepts than the number of concepts identified by Scode. We did not use a constant number of concepts to account for classes that are inherently related to very few or many concepts. Because EasyESA's results are ranked, we selected the 100 highest ranked concepts for the annotation task.

In total, we generated 20 005 class–concept pairs to evaluate. For each pair, the two authors judged whether the concept may be relevant to a developer working with the class. We did not rely on external annotators for this procedure as it requires a significant effort to read and understand the code of each class and peruse the content of Wikipedia articles before judging whether the concepts are related. To avoid a negative bias against baseline techniques, the technique that generated each pair was not identifiable during the annotation task. We also put all concepts in lower case, and removed parentheses from Wikipedia titles to further reduce the distinctions between Wikipedia-based concepts and the free-form concepts from PengKG.

The author and his supervisor first annotated all concepts associated with two classes and discussed the results to refine the annotation procedure. We found that it was unreliable to grade relatedness on a scale. Thus, the annotators annotated each pair as a binary variable, i.e., whether the concept is related to the implementation of the class, its usage, or the abstraction it represents. During the preliminary phase, we also found that some general concepts could be relevant to virtually all developers for any class. These concepts include, for example, fundamental computer science concepts such as Byte and Conditional branching. Annotators assigned the special annotation general to these concepts and did not further assess their relatedness to specific classes.

After refining the annotation guidelines (see Appendix A.3), each annotator independently annotated 54 of the remaining 98 classes, so that the concepts associated with ten unmarked classes would be annotated by both annotators to assess the inter-rater reliability. The two annotators achieved a substantial agreement [112] (Cohen's $\kappa = 0.74$ [52]). Conflicts were resolved by a discussion between both annotators.

3.5.2 Precision of the Identified Concepts

Table 3.7 summarizes the results of the annotation task to answer our first research question. The number of concepts identified by each technique and those that we annotated are shown in the "Concepts" and "Sample" columns, respectively. The sample size varies based on the

Table 3.7: Concept Identification Precision for 100 Java Files

Technique	Concepts	Sample	Gen.	Relevant	Prec.
PengKG	1006	1006	187	584	71.3%
EasyESA	44 069	9455	674	340	3.9%
top 1	97	97	9	20	22.7%
top 3	291	291	36	44	17.3%
with inclusion list	1160	1160	439	79	11.0%
top 1 inclusion list	93	93	26	22	32.8%
top 3 inclusion list	265	265	96	38	22.5%
Scode	43 584	9663	6432	146	4.5%
small topics	1453	288	159	32	24.8%
representative	548	115	93	9	40.9%
explicit	1071	306	208	65	66.3%
implicit	42 513	9357	6224	81	2.6%

Table 3.8: Examples of Concepts Identified by PengKG, EasyESA, and Scode

Technique	Sample Concepts (original capitalization)
PengKG	PREFERREDSIZE, Minimum Size, Property Change Listener, utc, detail message, CompositeData value
EasyESA	Byte stream, GUI widget, Stream (computing), Endianness, Home directory, Aspect ratio
Scode	Concurrent computing, Character encoding, Serialization, Layout manager, Dialog box, Parsing

number of concepts identified by each technique (capped at 100 per class) for the 100 classes. The "Gen." column shows how many concepts were marked as general. Finally, the last two columns report the number of class–concept pairs marked as related and the precision of each technique, computed as the ratio of relevant to non-general concepts. We excluded general concepts from the precision as they are related to virtually any class, but capture little information about them. This conservative decision impacted Scode the most. For each technique, the top row in bold indicates the global values for the technique, and subsequent rows show values for different variations as described below.

PengKG. PengKG is the most precise technique with a precision of 71.3%. It returned a number of true positives comparable to but smaller than the other techniques.

However, while inspecting the concepts provided by PengKG, we observed recurrent patterns of concepts that matched the class name and its members. For example, the class ServerSocket, which contains the methods getLocalPort() and getInetAddress(), was associated with the concepts Local port and Inet address. Although relevant, these concepts do not reveal additional information beyond the interface of a class. In particular, PengKG associated 44 of the 100 classes with a concept that exactly matches the class name. PengKG also contained many overly specific concepts such as CompositeData value (see Table 3.8). Contrary to concepts with an externally recognized definition, these concepts are meaningless without prior knowledge of the class implementation.

EasyESA. In contrast to Scode and PengKG, EasyESA ranks concepts from most to least relevant to the input text. Considering only the top concepts increases the precision of EasyESA. However, we found that even when considering only its highest ranked result, the precision was only 22.7%. As more concepts are considered, the precision decreased rapidly, already to 17.3% when considering the top three concepts (see rows *top 1* and *top 3* of Table 3.7).

Scode. Scode produced a large number of concepts for each class, with a considerable amount of noise. The high proportion of false positives can be explained by explicit general concepts, which tend to gather large, ill-defined topics during the community search phase of Scode. Concepts from such large topics were also pervasive in the annotation sets due to the uniform sampling across topics. For example, if Scode produces for a class three small, cohesive topics of 20 concepts each, and one large, low-quality topic of 500 concepts, sampling 100 concepts uniformly from the 560 identified by Scode will select mostly concepts from the large topic, despite most topics being more specific to the class. Specifically, only 1453 of the 43 584 concepts (3.3%) identified by Scode were in topics with at most 25 concepts.

When considering only concepts included in topics of 25 or fewer concepts, we observed that Scode's precision was significantly higher, up to 24.8% (see row *small topics* of Table 3.7). We used a threshold of 25 concepts for the size of a small topics based on our preliminary observation of a gap in the distribution of topic sizes, between 25 and 75 concepts. Although the precision remains modest, within a topic of 16 concepts, approximately four would be directly relevant to the project, thus warranting a look by developers unfamiliar with the domain.

We also assessed the quality of the topics by considering their single representative concept, as defined in Section 3.4.3. The precision of these concepts reached 41% (see the *representative*

row of Table 3.7), although a disproportionately large number of concepts were marked as general and thus excluded from this computation.

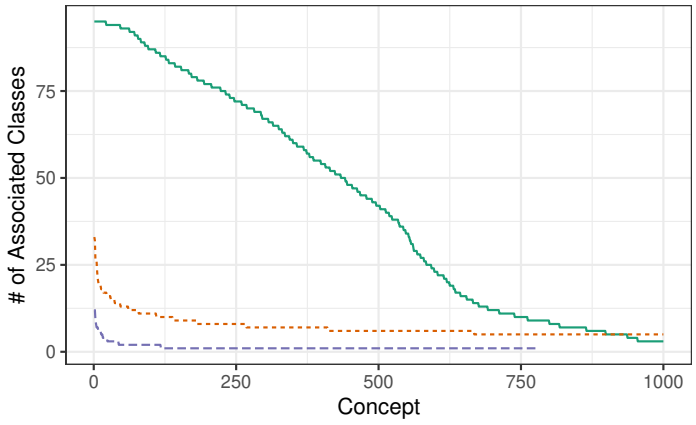
Without constraints on the number of topics, we found that most of the time, Scode produced a manageable number of topics. For our sample of 100 Java classes, Scode produced at most 15 topics per class. For 90 classes, it even produced ten or fewer topics (including the five classes for which no concept were identified). The number of topics for an entire project is also small: When using 227 Android projects as input to Scode (see Section 3.5.6), it produced on average 16 topics per project, with 90% of the projects having fewer than 26 topics. Thus, it is possible for humans to get an idea of the conceptual context of an entire project by reviewing only a small number of topics.

Findings: Scode achieves a low precision when considering all implicit concepts (4%). However, when filtering out large general topics, the precision rises to 25%. Scode is even more precise for topic representatives (41%). In any configuration, Scode outperforms EasyESA for a comparable output, but EasyESA generates fewer general concepts. PengKG is the most precise concept identification technique (71%), but returns many trivial concepts. In all cases, the modest precision levels suggest that concept identification techniques are better suited to support semi-automated applications, where a human judge can filter false positives.

3.5.3 Consistency of the Identified Concepts

Our second research question focuses on Scode's ability to generate recognized concepts consistently across projects. We looked at the concepts marked as related to get a sense of each technique's performance, and report illustrative examples in Table 3.8. Although PengKG is more precise than Scode and EasyESA, the concepts it identified were specific to each class. For example, PengKG identified that many classes of the Java Swing library define a PREFERREDSIZE, but such a concept has no meaning outside this library. Other concepts, such as CompositeData value, are obscure outside the specific context of the class. In contrast, Scode and EasyESA's concepts have by design a meaning recognized in a popular independent knowledge base.

To empirically assess the consistency of the three techniques' output, we measured how often each technique identified the same concept for different classes. Although some concepts are genuinely relevant to few classes, a technique should not only identify concepts so specific



The x axis shows each concept, in decreasing order of their recurrence across classes. For readability, the x axis is limited to the 1000 most recurrent concepts.

Figure 3.13: Distribution of Concept Frequencies for Scode, EasyESA, and PengKG

that they are only related to a single class. Otherwise, these concepts will be of limited value when comparing the conceptual context of different classes.

Figure 3.13 shows the recurrence of concepts in our sample of 100 Java classes. We used all concepts returned by the three techniques to produce this graph, without limiting them to 100 concepts per class to avoid the bias of a random filter on the output. Therefore, the figure includes false positives, as not all results were annotated.

As the figure shows, Scode identified the same concept for multiple classes more often than EasyESA and PengKG. Some general concepts were associated with almost all classes, whereas other concepts were more specific to a few classes. This distribution of classes per concept effectively supports a comparison of the domain of multiple classes.

In comparison, PengKG's concepts were rarely reused across classes. Specifically, the 1006 class–concept pairs from PengKG contained 776 distinct concepts, 84.9% of which were associated with a single class. The most common non general concept, PREFERREDSIZE, was associated with only eight classes. In comparison, Scode's 43 584 class-concept pairs involved only 1281 distinct concepts. Only 10.9% of them were associated with a single class, and the median number of classes associated with each concept was 18.

Despite also using a finite set of concepts, i.e., Wikipedia articles, EasyESA also rarely identified the same concept for multiple classes. Of the 27 943 distinct concepts identified by EasyESA, 71.9% were associated with a single class. The most common concept, METHOD, was only associated with 33 classes, even though such a general concept is related to virtually any Java class. This result suggests that using an external knowledge base is not sufficient to identify consistent concepts, and shows the importance of finding a strategy to identify implicit concepts.

Findings: Concepts identified by Scode are more consistent than those identified by EasyESA and PengKG. They were linked to a varying number of the 100 sampled classes, from over 90 classes to a single one, with a median of 18. In contrast, the median EasyESA concept was linked to two classes, with the most common concept, METHOD, only found for 33 classes. The majority of PengKG's concept (85%) were linked to a single class.

3.5.4 Inclusion List of Computing Concepts

One of our contributions is the manually curated list of 6746 Wikipedia articles about computing-related concepts. Scode uses this list to remove false negatives from the output of the wikification service. We investigated whether this inclusion list could also improve the precision of other concept identification techniques by applying it to EasyESA.[50]

When using the inclusion list to filter the top 100 results from EasyESA, we observed that it removed 87.7% of the original concepts, increasing the precision up to 11.0% (see row *with inclusion list* of Table 3.7). When considering only the highest ranked result that is also on the inclusion list, EasyESA achieved its highest precision of 33%, but this precision dropped rapidly, reaching 22.5% for the first three concepts (see rows *top 1 inclusion list* and *top 3 inclusion list* of Table 3.7).

An inclusion list of concepts also reduces the number of unique concepts that a technique can return, limiting the number of concepts associated with a single class. In the case of EasyESA, among the 1070 unique concepts both in the top 100 results for a class and in the inclusion list, only 38.5% were associated with a single class, and the median concept was associated with three classes.

Findings: Applying the computing-specific inclusion list of Scode to EasyESA improves its precision from 23% to 33% (considering only the highest ranked concept). Although better, it remains below the precision of Scode (41% for topic representatives or 66% for explicit concepts).

3.5.5 Wikification and Community Search Algorithms

We measured the performance of the wikification service by considering only the subset of explicit concepts in our annotated data set (see *explicit* row of Table 3.7). We found that

[50]As PengKG does not use Wikipedia articles as concepts, we cannot apply our inclusion list to this technique.

Babelfy identified on average 10.7 concepts per class, with a precision of 66.3% for non general concepts.

The precision is consistent with what we had observed in the prior comparative study of six wikification tools (Section 3.3). In this prior study, Babelfy achieved a precision of 75% with the same configuration as used in this current work. For this precision level, Babelfy achieved a recall of approximately 35%. This value is close to the maximum performance of other wikification techniques, which hardly surpass 40% recall, motivating the use of community search to recover missing concepts.

An important difference between the comparative study of six wikifiers and this study is the sample of documents. The comparative study used Stack Overflow posts as a sample of software documents about a variety of technologies. In contrast, this study relies on the documentation comments (i.e., Javadoc) of Java classes, consistently with the application domain of Scode. Although they share some similarities (e.g., varying in their length and editorial quality), the two types of documents differ in their nature, which could affect the performance of wikifiers. Nevertheless, we observed a similar performance of Babelfy in both studies. This promising result suggests that the performance of wikification approaches may generalize to different types of documents within the software domain.

Excluding the explicit concepts, we observed that the precision of the community search algorithms was low, 2.6%, for identifying implicit concepts (see *implicit* row of Table 3.7). However, as discussed above, most false positives are due to large, generic topics. When considering only small topics, which mostly contain implicit concepts, the precision of Scode is well above the 2.6% value. Thus, the drop in precision is a necessary cost to achieve additional benefits, such as identifying further implicit concepts, grouping the results by topic, and discriminating specialized concepts from generic ones (based on the size of the community).

Findings: Babelfy's precision (66%) is almost on par with PengKG (71%), the most precise technique in our evaluation. In contrast, the community search algorithms introduce a relatively large amount of noise when identifying implicit concepts (precision of 2.6%), especially due to concepts that generate large communities.

3.5.6 Topic Cohesiveness

We evaluated the ability of the community search algorithms, developed for general network science applications, to generate cohesive topics in the context of software-specific knowledge

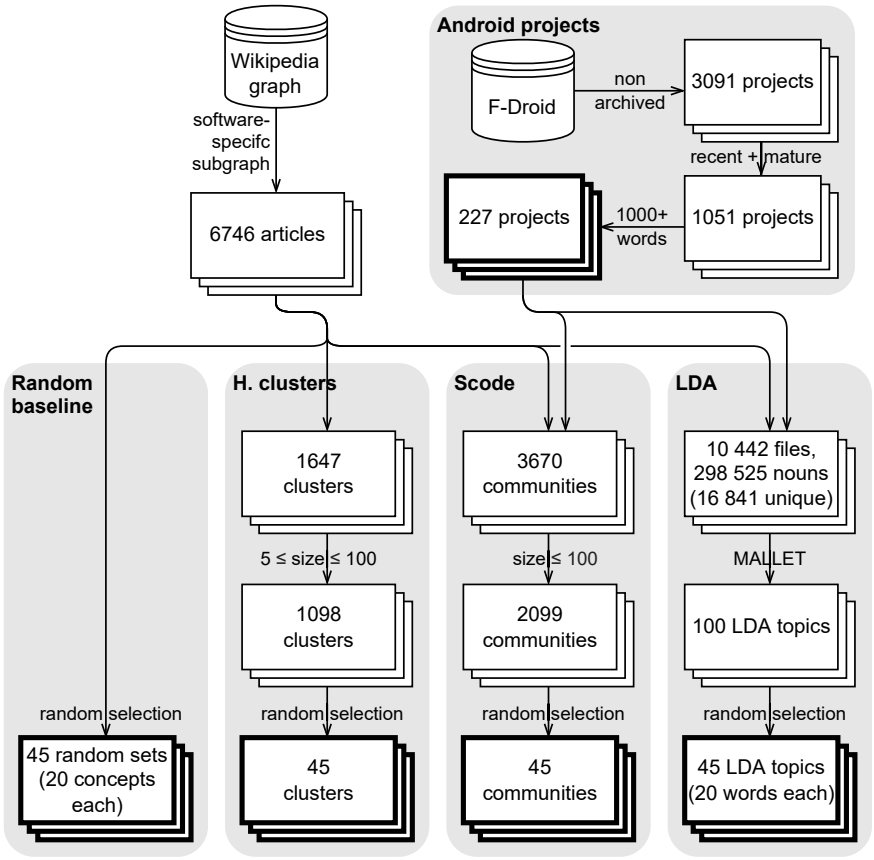


Figure 3.14: Sample for the Topic Cohesiveness Evaluation

graphs. We compared the community search algorithms to three baselines: a hierarchical clustering algorithm [198], latent Dirichlet allocation (LDA) [26], and random groups of concepts as a soundness check. Because LDA requires a large training set, we sampled entire Java projects for this evaluation.

Project Sample

Figure 3.14 summarizes our project and topic sampling procedure. We sampled Android applications from F-Droid [66]. We considered all 3091 applications that were not archived at the time of the evaluation. We excluded from them applications for which the last version was released five or more years ago (*recency filter*) or for which the time difference between their first and last released versions was less than one year (*maturity filter*). The recency filter is relevant to ensure that recent concepts have a chance to be included in our sample. The maturity filter excludes trivial applications.

After applying these two filters, we obtained a set of 1051 projects. We removed projects for which the entire documentation contained less than 1000 words, computed as described in Section 3.4.2 (*documentation filter*). We only considered classes and interfaces in the package matching the application ID of the project, or one of its subpackages. After applying this filter, we obtained our final sample of 227 Android applications, listed in Appendix A.5.

Topic Sample

We sampled 45 topics from each approach. This sample size is sufficient for a statistical analysis when comparing ordinal score distributions, but small enough to limit the threat of annotation fatigue (see the *Evaluation Metrics*).

We applied Scode to the 227 Android projects, generating a total of 3670 topics. We removed those that contained more than 100 concepts. We randomly sampled 45 of the remaining 2099 topics as the sample set from Scode.

Generating hierarchical clusters requires a distance metric between any two concepts, which we defined as the length of the shortest path between them in the undirected and unweighted Wikipedia subgraph. We then applied R's implementation of the unweighted pair group method with arithmetic mean (UPGMA) algorithm [198] to aggregate concepts into hierarchical clusters. As the outcome of this algorithm is a binary tree by design, it is sensitive to noise in the distance function. To mitigate this sensitivity, we rounded the distance function during the merging phase of the UPGMA algorithm to produce a tree that is not binary and less deep. After trying different rounding functions, we chose to round distances to the nearest non-exceeding quarter (e.g., 3.1 and 3.2 were rounded to 3 and 3.3 was rounded to 3.25). We excluded clusters if they either contained fewer than five concepts or were contained in larger cluster of ten or fewer concepts. We also removed clusters of more than 100 concepts. This procedure generated 1098 topics, from which we sampled 45 at random for the evaluation.

To generate LDA topics, we used the MALLET implementation of the algorithm for Java [136]. LDA is an unsupervised learning technique that takes as input a set of documents and generates latent topics represented as lists of words [26]. As LDA requires a training set with many documents to generate a good model, we considered each of the 10 442 files across all projects as a distinct document. For each document, we only retained nouns, for consistency with Wikipedia titles which are predominantly noun phrases. We further removed stop words and converted all nouns to lower case. This preprocessing produced a total of 298 525 terms across the 10 442 documents, forming a vocabulary of 16 841 unique nouns. We

generated 100 topics, with the initial value of parameters α and β set to 0.01, and using 2000 iterations. We left other parameters to their default values. The resulting topics comprise between 62 and 670 terms (average of 295).

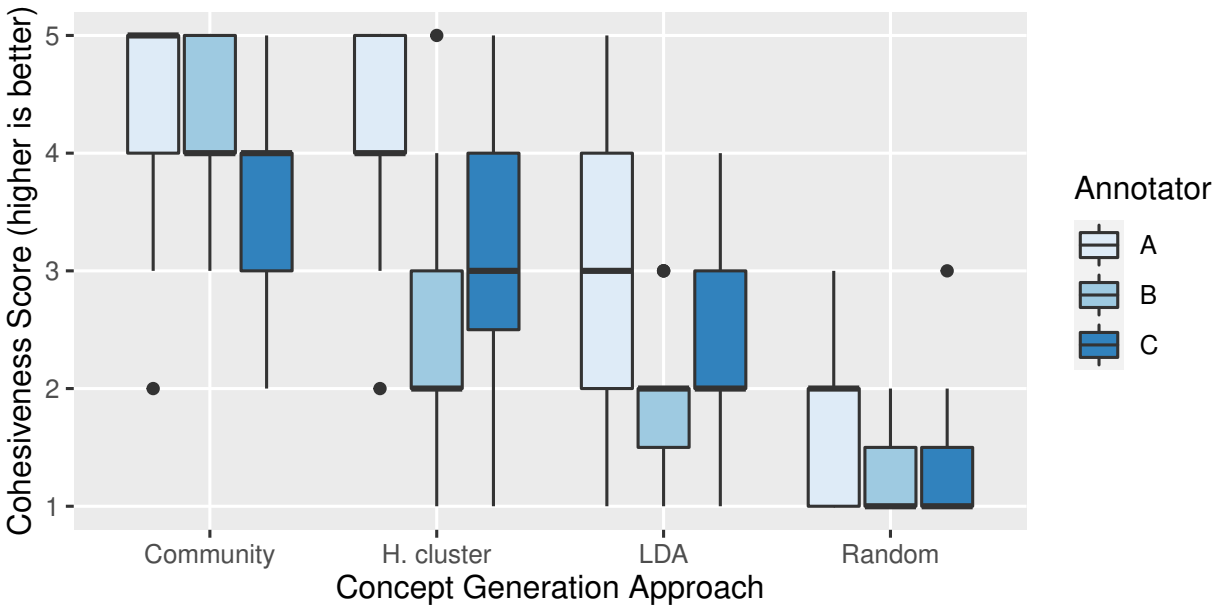
Finally, to support a soundness check, we generated 45 sets of 20 random concepts selected using a uniform probability distribution over the 6746 computing-related Wikipedia articles, with replacement between sets.

Evaluation Metrics

We used two complementary metrics to evaluate the cohesiveness of the topics. The first metric is a direct, subjective assessment on a five-point ordinal scale from 1 (least cohesive) to 5 (most cohesive). We asked three graduate students that were not involved in the project to provide this assessment. For each topic, we showed a maximum of 20 concepts to avoid overwhelming annotators with very large topics. We split the topics so that each annotator evaluated 15 topics from each approach, with no overlap.

We used a second metric to mitigate the subjectivity of the direct assessment. This metric is the success rate of the *word intrusion* task, described by Chang et al. [43]. In this task, an annotator is shown a set of $N + 1$ concepts in random order. N of these concepts belong to the same topic and the last one, the *outlier*, is chosen randomly among concepts not in the topic. The annotator must then identify the outlier among the $N + 1$ concepts. If a topic is cohesive, identifying the outlier should be easy, resulting in a high success rate. However, concepts from a vague topic will be indistinguishable from the outlier. The success rate for such topics should decrease towards $1/(N+1)$. For this evaluation, we chose the value $N = 5$. We generated a word intrusion task for each topic, and asked the same three annotators to perform the task, dividing topics evenly among them and ensuring that the same annotator did not provide a subjective score on the same topic for which they performed the word intrusion task.

For both measures, annotators were unaware of the technique that generated each topic. They did not know how many techniques were being compared, or that some concepts were Wikipedia titles (which were shown in lower case) whereas others were simply terms from the vocabulary of the LDA model. When selecting the evaluation metrics, we also considered synthetic measures such as the Silhouette index [193]. Although these metrics are arguably more objective than those we selected, their abstraction of human judgment poses a threat to construct validity.



Each boxplot aggregates the 15 topics evaluated by one annotator.

Figure 3.15: Distributions of the Cohesiveness Scores on a Five-Point Ordinal Scale for each Combination of Approach and Annotator

Results

The community search algorithms generated topics that were more cohesive than those from the three baselines. Figure 3.15 shows the distribution of cohesiveness scores given by each annotator to topics generated by all four approaches. Each boxplot aggregates the scores of exactly 15 topics. The results show a consistent trend across annotators: communities are generally more cohesive than hierarchical clusters, followed by LDA topics, and finally the random baseline. As expected, random topics were the least cohesive, and validate that annotators can discriminate spurious topics by giving them very low scores, most often one or two.

Table 3.9 confirms this observation. It shows the median cohesiveness score of each group of topics, as well as the average scores, in parentheses, as this measure is more commonly used to approximate the center of a distribution. However, because cohesiveness scores are not on an interval scale, the averages cannot be used for further statistical analysis or interpretation. We used one-tailed Mann–Whitney U tests [132] to test the hypothesis that the cohesiveness differences are not due to chance. This test compares two groups of ordinal values and evaluates the probability that a value randomly selected from the first group is larger than a value randomly selected from the second group. The stars beside each value indicates

Table 3.9: Success Rate of the Word Intrusion Task with Median (and Average) Cohesiveness Scores for Concepts Generated by the Four Techniques

Annotator	**Community**	**H. cluster**	**LDA**	**Random**
A	60%; 5 (4.3)	40%; 4 (4.2)**	60%; 3 (3.0)**	20%; 2 (1.7)
B	67%; 4 (4.3)***	60%; 2 (2.5)	60%; 2 (1.9)**	13%; 1 (1.3)
C	73%; 4 (3.8)*	60%; 3 (3.1)*	73%; 2 (2.5)***	13%; 1 (1.3)
All	67%; 4 (4.2)***	53%; 3 (3.3)***	64%; 2 (2.5)***	16%; 1 (1.4)

Cohesiveness scores are rated on an ordinal scale from 1 to 5, with higher scores given to more cohesive groups of concepts.
Stars indicate the p-value of a one-tailed Mann–Whitney U test comparing the distributions of cohesiveness scores in a column to those to its right: less than 0.05 (*), 0.01 (**), or 0.001 (***).

the significance of the tests between consecutive approaches in the sequence {communities, hierarchical clusters, LDA, random}.[51]

All but two comparisons are statistically significant at the 0.05 level. The difference between hierarchical clusters and LDA topics annotated by B has a p-value of 0.080, and the difference between communities and hierarchical clusters annotated by A has a p-value of 0.30. These inconclusive results can be explained by the small group sizes (15 topics), as well as the high cohesiveness scores (capped at 5) given by A, which limits the discrimination of the most cohesive topics. When comparing all 45 topics from each approach, all tests are significant, with p-values all lower than 0.00092, confirming our initial observation (communities > hierarchical clusters > LDA topics > random concepts).

Table 3.9 also includes the proportion of outliers correctly identified in the word intrusion task. Interestingly, although communities remain ahead of the other approaches, LDA topics appear to be more cohesive than hierarchical clusters. One possible explanation for the higher success rate with LDA topics is that outliers are selected from a different vocabulary (i.e., Wikipedia articles) than the terms that form the topics (i.e., nouns).[52] Due to the binary result for each topic (success or failure), the only statistically significant differences are between the random baseline and the other approaches. Detecting a significant difference between the other approaches would have required an impractical sample size, i.e., over eight times larger to reduce the confidence interval radius to 5% at the 95% confidence level.

[51]P-values for non-consecutive approaches are smaller than the minimum of the p-values shown in the table. For example, the difference of *community* and *LDA* score distributions for annotator A is significant with a p-value lower than 0.01.

[52]We chose this conservative evaluation approach because it favors the baseline. The alternative, i.e., selecting an outlier among the much larger, non computing-specific set of nouns used to train the LDA model, would have had a negative impact on the LDA baseline.

Findings: Three external annotators found topics identified by community search algorithms significantly more cohesive in a direct evaluation than topics generated by alternative techniques (hierarchical clustering and LDA). This subjective preference was confirmed by a corresponding performance on a word intrusion task.

3.5.7 Documentation or Source Code as Input

In this work, we considered the documentation of a software project as input for identifying concepts. This decision contrasts with prior approaches that extract concepts from source code identifiers (e.g., [1, 103]).

Although using identifiers is the only option when documentation is lacking, using documentation as input can leverage more diverse forms of information. We hypothesized that the two strategies should generate different concepts, because information in source code is at a lower level of abstraction than the information found in documentation. To validate this hypothesis, we compared the output of Scode to concepts generated from source code identifiers.

We used LDA to generate topics from the identifiers in all the source files of the Java 15 standard library.[53] We considered each file as a separate document. We split each identifier based on camel case conventions and underscores and removed terms with less than three characters. Then, for each group of terms with the same stem according to the Porter stemming algorithm [177], we replaced all terms with the most common form. We set the number of topics to 100, the number of training iterations to 2000, and left the other hyperparameters to their default values in MALLET.

We compared the top LDA topics with Scode's concepts that are included in small topics (see Section 3.5.2). As it is difficult to compare LDA topics to concepts represented by Wikipedia articles, we focused on the *vocabulary* used by both strategies. We used the union of the top 20 terms from the top 10 LDA topics and compared it to the set of all terms from all Wikipedia titles, excluding terms in parentheses. We used their overlap coefficient, i.e., the ratio of the sets' intersection size to the size of the smaller set, to quantify this comparison. As MALLET's implementation of LDA is not deterministic, we repeated this procedure 100 times.

[53] An alternative would be to use Scode to identify concepts directly from source code. However, this comparison would be unfair as the wikification service expects natural language inputs. Most prior work uses language models such as LDA to identify concepts.

We found that the overlap between Scode's concepts and LDA topics varied depending on the class. Excluding classes for which Scode did not find any small topic,[54] the average overlap coefficient for each class, across the 100 trials, ranged from 0 to 0.19 (standard deviation ranges from 0 to 0.11). Inspecting the LDA topics confirmed that they capture low-level concepts. In addition to implementation-specific topics with terms such as *length*, *attribute*, and *string*, other topics representing domain abstraction are expressed with low-level concepts. For example, a topic about time zones includes terms such as *central*, *america*, and *gmt*.

Findings: Concepts extracted from recurrent terms in code identifiers differ from those extracted using Scode: their vocabulary overlaps on average by less than 20%. This is explained by the identifier-based concepts typically being at a lower level of abstraction than Scode's concepts.

3.5.8 Discussion

Beyond evaluating the performance of a single tool, the goal of our evaluation was to explore the theoretical potential and limits of a new concept identification strategy. The findings demonstrate the difficulty of this problem. Both phases of Scode achieved good performance in isolation, but together, the precision remains low. Although there are more precise concept identification approaches, they typically represent concepts using a project-specific terminology, limiting the comparison and understandability of the concepts outside the project's context. Even for Scode, configuration details matter: Using only concepts from small topics, topic representatives, or explicit concepts has a considerable impact on the precision and coverage. Hence, there is currently no single technique to solve all concept identification tasks.

Scode improves the state of the art with regards to identifying recognized concepts from documentation. The evaluation showed the potential of an approach like Scode to identify concepts more consistently. It also elicited challenges to solve to continue improving our approach. In particular, the density of links in the Wikipedia graph, despite our pruning steps, introduced noise in the community search algorithms and generated very large topics that are impractical for developers. Although itself a challenging knowledge engineering task, constructing a graph of Wikipedia articles with fewer, more meaningful links should improve the precision of Scode while reducing the size of its output.

[54]These classes were mostly custom exception types and poorly documented classes.

Another aspect to consider when improving concept identification strategies is the different relations between concepts and a project. Some concepts that are not related to the current implementation of a project may still be relevant to know for developers. For example, developers who work with the MD5 cryptographic algorithm should likely be aware of alternative algorithms, such as SHA-256. There may also be a distinction between core concepts implemented by a project, and concepts implemented in third-party libraries. Precisely identifying such categories of relatedness would be a promising avenue for future work.

Finally, there is no inherent reason for Scode to use only header comments as input. Future work could include investigating more types of documentation sources, such as requirements files and issue trackers, and more preprocessing strategies to improve the precision of Scode.

Solutions to these challenges can be integrated within Scode without having to modify the other components. Hence, Scode provides a framework to study different aspects of the concept identification and knowledge graph construction problems.

3.5.9 Threats to Validity

The intangible nature of concepts and of the relatedness relation introduced threats to the *internal validity* of our results. Judging whether a concept is related to a class is a subjective decision. To control this subjectivity, we prepared a strict coding guide, which in particular avoided speculations about concepts that may be relevant to be aware of, even though they are not directly used by a class. The authors also performed the annotation task entirely. Although this may introduce a bias in the annotations, judging the relevance of a concept to a class requires a considerable effort to understand the class and the concept, as well as their context. Thus, we favored this threat over the one that would have been introduced by less motivated external annotators. We further mitigated the threat of investigator bias by removing any indication of the technique that identified each concept during the annotation task.

Performing the end-to-end evaluation using classes as a substitute for projects also poses a threat to validity. This was necessary to design a practical evaluation, as it would be infeasible for annotators to read and understand all components of entire projects. The impact of this threat was mitigated by the fact that, in the context of this work, a concept that is relevant to even a single class in a project is by definition relevant to the project. This property shows a beneficial aspect of Scode: it can be used to analyze projects at different granularities, from a project to a class level.

Similarly, most of our results focus on concepts rather than topics. We made this decision as prior work have already rigorously evaluated the properties of the topic modeling techniques used by Scode, including the communities' cohesiveness [6, 68, 97, 100] and the use of *pagerank* to select a topic's representative [124]. To ensure these previous results would apply in a software development context, we performed additional evaluations to measure the cohesiveness of topics and the precision of topic representatives.

The evaluation of the topic cohesiveness relied on the judgment of external annotators. In this case, we found that external annotators were a better option as the annotation tasks were better encapsulated and required less contextual knowledge. Nevertheless, the background of each annotator can affect the internal validity of the results. To mitigate this threat, we triangulated results from two complementary metrics—a direct but subjective assessment of the cohesiveness and the success rate of the word intrusion task—and from three annotators working independently. To assess the consistency of the external annotators, we tested the hypothesis that topics for which an annotator succeeded at the word intrusion task would receive higher cohesiveness scores by the other annotators. We observed that all tests were significant at the 0.05 level using Mann–Whitney U tests, except the one that compared cohesiveness scores attributed by annotators B and C to topics for which the word intrusion task was performed by annotator A (p-value of 0.086). This suggests that the results of the evaluation are reliable, despite the subjectivity of cohesiveness.

The different vocabulary used by LDA topics and PengKG's concepts can also introduce a bias in our evaluation. We mitigated this bias by normalizing the concepts presentation during the annotation tasks, i.e., by putting them in lower case. We also removed the occasional clarifying terms in parentheses that are characteristic of Wikipedia titles for the end-to-end evaluation. By doing so, different articles can result in the same concept. During the annotation task and when reporting results, we considered these articles as a single, polysemous concept. We erred on the side of inclusion when judging the relatedness of such polysemous concepts, as trying to find an unrelated sense for each concept would be counter-productive. We did not remove parentheses from Wikipedia titles for the cohesiveness evaluation as the bias favored the LDA baseline.

Finally, our sampling strategies limit the generalizability of our results. For the topic cohesiveness evaluation, we sampled open source Android projects. This sampling frame biases the output of the compared techniques towards Java- and Android-related topics. For the end-to-end evaluation, we used classes instead of larger project components as input units to ensure that it was possible for the annotators to reliably understand the context of each unit when judging the concepts. We also only sampled classes from a single source,

i.e., the standard Java Class Library. However, given that this library is intended to provide fundamental classes supporting a wide variety of applications, it covered concepts from a large variety of computing domains.

Chapter 4

Variations in Software Tutorial Design

In the previous chapter, we reported on our investigation of conceptual dependencies of software documents. This research project applied to readers before they start reading documentation: Our findings can help capture the background that readers must have to understand a document, and orient them to further resources to address knowledge gaps prior reading the document. However, identifying the conceptual dependencies of existing documents is insufficient for a writer who wishes to improve the quality of the documents. The writer must judge which concepts they should describe and which ones they can leave as part of the expected background of readers. Beyond the selection of concepts to cover in a document, it is challenging to organize the content of a document in a way that allows readers to easily find the information they are looking for.

To address these challenges, we focus on the design of tutorial documentation. For software development organizations, both large and small, software tutorials serve as a façade for a technology and can help better market it [57]. Many developers refer to "Getting Started" documentation or introductory tutorials when learning new APIs [141]. Thus, high-quality tutorials can help organizations increase the usage share of their flagship products as well as upstart projects to acquire critical adoption levels.

What constitutes a high-quality tutorial, however, remains an open question. A glance at the offerings for popular technologies shows a diversity of styles related to content selection and presentation. In this chapter, we report on a systematic data-centric analysis of popular tutorials to better understand the design dimensions for software tutorials.

We looked at three tutorials aimed at beginners to Android programming: The App Basics tutorial from the official Android Developer Documentation (AndroidOfficial), Google's Android Developer Fundamentals tutorial (GoogleCourse), and Vogella's Android Development

tutorial (VogellaTraining). We selected these tutorials because of their quality, inferred from their cohesiveness, up-to-dateness, and authoritativeness. The selection process consisted of searching for Android tutorials on three web search engines and manually discarding low-quality results (see our online appendix for the exact selection procedure [14]). We avoided the many tutorials that consist of loosely structured collections of documents or poorly edited blog posts. We used this selection process to elicit a variety of design decisions made by professional writers, rather than to gather a representative sample of tutorials.

Despite their similar audience and purpose, the tutorials differ in essential ways. We observed radical variations in their division into sections and their use of code fragments and links to other documentation. The overlap of topic coverage between tutorials was also surprisingly low, hinting at different content selection strategies.

Publication This study of design variations in tutorials was published in an article titled *"A Data-Centric Study of Software Tutorial Design"*, which appeared in *IEEE Software* [15].

Study Data The complete details of the study procedure, including the criteria to select tutorials, as well as the data collected during the study, are available in a data artifact [14]. Appendix B details the content of this artifact.

Note This chapter presents the results from a collaboration with another PhD student, Deeksha Arya. The author and D. Arya contributed equally to this research project. We present all the results together to keep this thesis more coherent and self-contained.

Studied Tutorials

AndroidOfficial is the set of introduction documents under the "App Basics" header from the Android platform [8]. It is the official set of short references to get programmers started quickly on Android development. GoogleCourse is the "Android Developer Fundamentals" tutorial created by the Google Developers Training Team to accompany an in-person course leading to an entry-level Android certification [80]. VogellaTraining is the set of training resources under the "development starter" and "fundamental" sections of the Android Development tutorials of the Vogella training platform [225].

4.1 Design Decisions

Consistent with van der Meij et al.'s discussion of the evolution of content and presentation of software tutorials between 1980 and 2009 [138], we chose to compare the three tutorials based on

Table 4.1: Organization Design Dimensions with Sample Decisions and Their Impact on the Readers

Decision	When/Why	Trade-Off
Structure of the tutorial components		
Sequential sections forming a single narrative	Provides beginners with an explicit entry point and a measurable progression	Requires readers to go through sections that may be less relevant to them
Modular set of independent, focused sections	Allows more advanced developers to only read sections that address their information needs	Creates more complex dependencies between sections that can be challenging to navigate
Context included in a code fragment		
Complete **compilable code** (e.g., entire files)	Encourages readers to clone the examples for a more participatory tutorial	Can distract from or hide the code elements of interest
Short **focused fragments** (e.g., few statements)	Focuses the discussion on relevant code, e.g., when comparing different approaches	Can become challenging for readers to integrate many focused fragments together
Links to external resources		
Integral component of the tutorial	Reduces tutorial creation and maintenance effort, and provides a broad overview of the topic	*All links:* Each additional link can distract the reader, who needs to jump back and forth between the tutorial and external resources
Optional supplement for specialized topics	Allows readers to further their expertise and interest on a topic	

two aspects: the *content* they present, and the *organization* of this content. We captured details about the organization of a tutorial by extracting structural properties such as the number and length of each document and section of a tutorial and of its code examples. We operationalized the content of a tutorial using Stack Overflow tags as a closed set of topics that a tutorial may or may not cover. This data allowed us to reflect on the design of the tutorials based on quantitative evidence.

We observed several impactful differences between tutorials, which we articulate as eleven design decisions along five dimensions. Tables 4.1 and 4.2 summarizes these decisions, with their rationales and trade-offs, related to the organization and content of a tutorial, respectively. Importantly, different parts of the same tutorial can realize different decisions, even within the same dimension.

Table 4.2: Content Design Dimensions with Sample Decisions and Their Impact on the Readers

Decision	When/Why	Trade-Off
Main topic selection strategy		
Selection based on **external factors**	Tailors the tutorial content to demonstrated information needs	Reduces the cohesiveness of topics, compared to a baseline reflecting the author's perspective
Selection based on **interactions** between topics	Describes how topics work together to build more complex applications	Requires a lot of effort: the number of interactions grows exponentially with topics
Selection of additional topics		
Implementations of broad core topics	Provides concrete details to understand a core topic in a specific context	*All topics:* Each additional topic lengthens the tutorial, making it more expensive to create and maintain, and more daunting to readers who may want to get coding quickly
Non-functional topics (e.g., development tools)	Introduces beginners to good development practices early	
Peripheral topics (e.g., third-party libraries, deprecated APIs)	Addresses varying needs of readers working on specialized applications or legacy systems	

For example, a tutorial creator hired to document new features of an API may choose to design the tutorial as a *modular* set of independent sections for an audience already familiar with the technology. They may use *focused code fragments* throughout each section to keep the focus on the new features, and base the content of the tutorial on *external factors* such as popular feature requests to showcase the value of the updated API.

In the remainder of this chapter, we discuss each dimension in depth, including examples of design variations from the three Android tutorials, and details of the process we followed to analyze the tutorials.

4.2 Tutorial Organization

Table 4.3 reports structural properties of the tutorials under study. Although the three tutorials are not intended to be representative of all tutorials, we compared them to eleven other introductory

Table 4.3: Properties of the Three Studied Tutorials Compared with Eleven Other Android Tutorials for Context

Property	Android	Google	Vogella	Others (11)		
				Min.	Mean	Max.
Documents*	56	33	10	4	38	109
Sections*	292	697	333	16	252	712
Words	83 351	103 431	21 890	1559	30 925	104 441
... per document	1488	3134	2189	212	948	2984
... per section	285	148	66	32	135	240
Code fragments	338	430	174	2	222	711
Visible code fragment characters	67 810	90 107	102 141	172	132 528	454 856
... per code fragment	201	210	587	86	607	905
Hyperlinks (non-self referencing)	1447	1800	64	3	237	847
... to other pages of the tutorial	529	2	4	0	42	124
... to advanced tutorial pages	110	0	2	1	92	340
... to API reference documentation	602	1018	0	0	4	29
... to other resources	206	780	58	1	98	606

* A document refers to a web page, delimited into sections by a header (HTML `h1`-`h3` tags).

Android tutorials to assess how they fit among the range of available tutorials (see the online appendix for the details of our sampling strategy). We found that, except for an uncharacteristically high number of hyperlinks, they do not exhibit unusual properties.

Intended Reading Order

All three tutorials are designed to be read in different manners. GoogleCourse's content is organized in a single sequence to read in a prescribed order. This single sequence is easy to follow for beginners who may not know in advance what information is the most relevant.

In contrast, AndroidOfficial does not have a clear reading order. Each document contains only information related to a focused subject, and delegates related information to other documents. Thus, AndroidOfficial consists of a complex network of 56 short documents (with 5.2 sections on average, compared to 21.1 and 33.3 sections for GoogleCourse and VogellaTraining) linked by 529 references to each other. Decoupled documents can improve their reusability in other learning frameworks, according to Boyle [29]. They are also useful to readers with specific information needs: Readers can access the desired information without wasting time on context built in prior documents. Links to other documents provide learning resources for related concepts if necessary.

VogellaTraining, with its ten documents, lies between AndroidOfficial and GoogleCourse with regard to its reading sequence. The largest document (103 sections and 8437 words, more than twice the size of the second largest) covers a list of concepts that beginners should read in sequence, similarly to GoogleCourse. The other nine documents explore in more details different fundamental aspects of Android. These shorter documents share similarities with AndroidOfficial's documents, as they can be read in any order and focus on a specific concept. The combination of both organization styles is a compromise that grants readers the flexibility to explore different aspects of the framework as they please, after having been introduced to fundamental notions relevant to all of these aspects.

Use of Code Fragments

The tutorials exemplify two approaches for presenting code fragments to the reader. AndroidOfficial and GoogleCourse mostly contain short code fragments that focus on the statements of interest. In contrast, VogellaTraining's code fragments often display an entire file, including more trivial information such as the package and import declarations. Despite these general tendencies, all three tutorials use both short and long code fragments at some point, with the largest code fragment having 2110, 1525, and 2368 characters respectively for AndroidOfficial, GoogleCourse, and VogellaTraining.

Code fragments that focus on a single method do not overwhelm the reader with unnecessary information. They convey a clearer purpose. In contrast, showing the complete content of a file provides context for the relevant code. It thereby allows readers to follow the evolution of code through several manipulations and understand how different concepts interact in a complete application.

Another interesting design decision, although only observed in AndroidOfficial, is to present equivalent code fragments in both languages officially supported for Android development, Java and Kotlin. The tutorial uses a tabbing mechanism to show fragments in the language the reader prefers. This design can increase the audience of a tutorial, but requires the additional cost of creating and maintaining pairs of equivalent code fragments.

Links to External Resources

Resources found outside a tutorial can complement the content of tutorials. These external resources can include pages of advanced tutorials from the same website, official API reference documentation, and other resources, for example, blogs and third party libraries.

AndroidOfficial uses external resources to lighten its content, allowing readers to go through each document more quickly. It contains 110 links to advanced tutorial pages hosted on the same website. There are 602 links to API reference documentation in AndroidOfficial, to avoid

redundant descriptions of API types. It also contains 206 links to other resources, among which 163 links refer to official documentation hosted on the Android Developer website, such as graphical design guides. This large number of links is representative of the Android documentation website, which can be viewed as a large network of learning resources, of which AndroidOfficial is a subset. These external resources conveniently refer readers to additional concepts, but can break the flow of the tutorial if readers navigate back and forth between the tutorial and external resources.

GoogleCourse also contains many links (1798) to external resources. The 1018 links to API reference documentation are due to mentions of API types being systematically linked to their reference documentation. The remaining 780 links point to a variety of resources, including websites of different technologies (e.g., the Mockito framework), Stack Overflow posts, and coding exercises, often under a "Learn more" header. This use of links contrasts with AndroidOfficial, as external resources are explicitly marked as supplemental material. Hence, in GoogleCourse, the external resources complement, rather than directly support, the content of the tutorial.

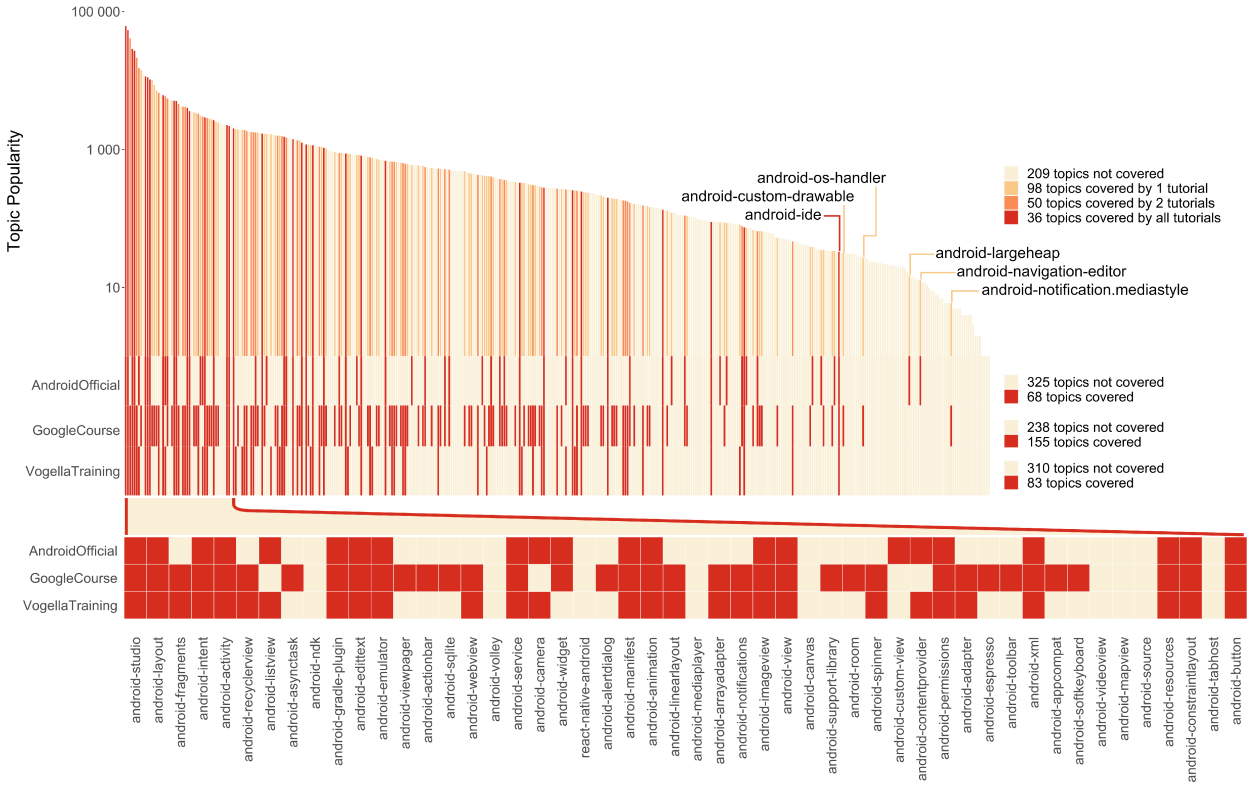
VogellaTraining contains far fewer links to external resources, amounting to 60 in total. It contains zero references to the API documentation, and so often repeats text that is already present in the official documentation. Without a method to synchronize updates to the official documentation with the content in the tutorial, VogellaTraining faces the risk of containing inconsistent information and becoming outdated [10]. The hyperlinks that it does contain usually point to the root page of documentation-hosting websites rather than specific documents. Similarly to GoogleCourse, these few links in the main text of the tutorial encourage readers to remain within the tutorial until its completion to limit potential distractions. VogellaTraining even includes programming exercises between the more conceptual sections, limiting the need for readers to refer to an external resource for practice material.

4.3 Tutorial Content

The lack of content presentation standard for tutorials makes their investigation a difficult problem [74]. To provide an objective definition of a *tutorial content topic*, we used Stack Overflow tags as proxies for topics.

We retrieved tags that contain the substring "android". After removing irrelevant tags, such as specific Android versions, we obtained 393 topics, related to API types (e.g., android-intent), libraries (e.g., android-glide), and generic concepts (e.g., android-camera).

For each of the $3 \times 393 = 1179$ tutorial–topic pairs, we manually identified whether the tutorial covers the topic, reporting the section where the topic is covered (when applicable) as evidence. We did not discriminate between degrees of coverage, but considered passing mentions as insufficient to cover a topic. All authors independently annotated a distinct set of pairs which included a common subset of 20 tutorial–topic pairs from each tutorial. On this common subset, the annotators achieved



The top section shows the popularity of each topic, measured by the number of Stack Overflow posts tagged by the topic. The middle section indicates, for each 393 topics, which tutorial covers it. The bottom section zooms in on the 50 most popular topics.

Figure 4.1: Popularity and Coverage of Android Topics by Each Tutorial

a pairwise agreement of 95% to 100% (Cohen's κ between 0.83 and 1.00) for AndroidOfficial and GoogleCourse, and of 80% to 100% (Cohen's κ between 0.49 and 1.00) for VogellaTraining, which demonstrates the reasonably low subjectivity of the task. The resulting mapping between topics and tutorials, with further methodological details, are available as part of our online appendix.

This procedure produced a coverage incidence matrix with three rows (tutorials) and 393 columns (topics), which provides valuable insights into different strategies to select the content of a tutorial. Figure 4.1 shows this incidence matrix (middle section), in relation with the popularity of each topic (top section).

Core and Additional Topics

We expected tutorials to cover a common core of essential Android topics. However, among the 393 topics, only 36 are covered by all three tutorials. This intersection of topics among tutorials is small in comparison with the 184 topics covered by at least one tutorial (20%). The 148 topics

covered by one or two tutorials are not equally distributed: GoogleCourse covers 119 of them, while AndroidOfficial covers only 32 of these additional topics.

The topics common to all three tutorials include the most basic aspects of Android development. For example, it includes android-layout, android-view, android-activity, and android-intent—the four pillars of any Android app. It also includes android-ide, android-gradle-plugin, and android-emulator to teach beginners to implement, build, and run an application. Thus, all three tutorials cover at least the essential topics to develop a basic application.

Topics covered only by AndroidOfficial relate to features of Android devices, even if they may not be used by a majority of beginners, such as android-gps, and android-strictmode. In contrast, the many topics solely covered by GoogleCourse include convenient API classes (e.g., android-pendingintent) and third-party libraries (e.g., pocketsphinx-android), helping beginners explore topics beyond those strictly necessary to start a project.

VogellaTraining covers some deprecated APIs, such as the popular tag android-listfragment. The coverage of deprecated APIs can be helpful for developers joining older projects or if the classes are used in Android projects despite the deprecation notice. A survey conducted by Lethbridge et al. revealed that 81% of the 45 respondents agreed that even though it may not be up to date, software documentation can still be useful [115].

The inclusion in a tutorial of topics beyond the essential core to use the framework constitutes a design trade-off. Comprehensive tutorials that cover a broader range of topics require additional effort from the authors and may discourage readers by their increased length. However, omitted topics can prevent beginners from exploiting useful features.

Popularity of Topics

We use the number of Stack Overflow questions associated with each topic as a measure of how prevalent its related information needs are among Stack Overflow users. The incidence matrix in Figure 4.1 shows that tutorials tend to cover popular topics. AndroidOfficial, GoogleCourse, and VogellaTraining cover respectively 22, 37, and 28 of the 50 most popular topics, but only zero, one, and two of the 50 least popular ones. However, all three tutorials cover topics across the whole range of popularity, and leave out some of the most popular topics that are demonstrably challenging for developers.

AndroidOfficial covers fewer of the popular topics than the other two tutorials. For example, it covers android-view, but not its more popular subclass, android-recyclerview. It excludes other popular topics that are not essential for building applications, such as android-fragments (third most popular). Thus, AndroidOfficial remains strict in its goal to provide the minimal information to build simple applications, as opposed to GoogleCourse and VogellaTraining, which are broader.

Tutorial authors cannot only rely on a topic's popularity to decide whether to cover it, as popularity is not a perfect assessment of relevance. Popularity depends also on the complexity of the topic: prevalent but trivial topics are less likely to generate questions from developers. For example, retrieving metadata about an Android application using the ApplicationInfo class is not very complex, but is an important task for beginners to learn. Introductory tutorials still need to cover these prevalent topics, so a lack of popularity does not indicate that the topic is irrelevant. Some of the least popular topics are covered by all three tutorials, including for example android-applicationinfo, which is among the 150 least popular topics.

Categories of Topics

Many of the covered topics are associated with an API type from the Android Framework (e.g., android-asynctask corresponds to the type android.os.AsyncTask). Although such type-related topics amount to only 34% of all 393 topics, they constitute the majority of topics covered by GoogleCourse and VogellaTraining (50% and 52%, respectively), and 38% of the topics covered by AndroidOfficial. The tutorials also cover many topics related to the architecture and components of Android, e.g., android-styles and android-manifest. Similarly to type-related topics, these topics reveal that GoogleCourse and VogellaTraining largely cover the functionality and behavior of the API, as opposed to other relevant concerns such as third-party libraries and development tools.

Another common kind of covered topics are Android features visible to end-users, such as android-sharing, and android-orientation. For example, AndroidOfficial, GoogleCourse, and VogellaTraining uniquely cover android-keypad, android-launcher, and android-button respectively. We observed that this kind of topic was especially prevalent in topics uniquely covered by AndroidOfficial.

Contrary to AndroidOfficial and VogellaTraining, GoogleCourse covers external libraries such as android-espresso and android-glide. Although it defers to their documentation for a more extensive description, a short introduction increases awareness of the Android development ecosystem.

Finally, all three tutorials cover few topics related to development methods and tools, such as android-lint and android-monkey, ignoring even official development tools such as android-ndk and android-jetpack. This general bias against development and maintenance concerns is surprising, as beginners would benefit from learning good development practices.

Correlated Topics

As software technology is comprised of multiple interacting components, cohesive tutorials cannot cover topics in isolation. For example, Android *activities* and *intents* should be discussed together,

as intents are used to switch between activities. To assess the cohesiveness of the tutorial contents, we looked at pairs of topics that frequently co-occur, i.e., are tagged on the same Stack Overflow question. We considered the pairs of topics that co-occur in at least 40 questions, which cumulatively constitute over 75% of all co-occurrences.

Figure 4.2 shows how many topics of each pair are covered by each tutorial. Both AndroidOfficial and VogellaTraining cover only one topic in almost half of the correlated pairs. Covering the other topic in these pairs would convey a more cohesive and complete perspective of Android programming, but each additional topic can in turn create a new correlated pair with only one covered topic. Tutorial authors must carefully choose when to stop adding related topics to avoid overwhelming a reader. Consistently with the previous observation of a small set of topics covered by all tutorials, only 42 topic pairs (6%) are covered by all three tutorials.

GoogleCourse and VogellaTraining both cover the two topics of *all* but three of the top 20 most frequently co-occurring pairs. In contrast, AndroidOfficial covers only half of the 20 most common pairs. This observation is consistent with the organization of AndroidOfficial as short, decoupled sections. In many cases, when a pair consists of a broad topic and a more specific one, e.g., android-layout with android-linearlayout, AndroidOfficial typically only discusses the broader topic. So AndroidOfficial provides an overall introduction to the framework, but leaves out specific instances of the different concepts.

4.4 Towards a Systematic Approach to Tutorial Design

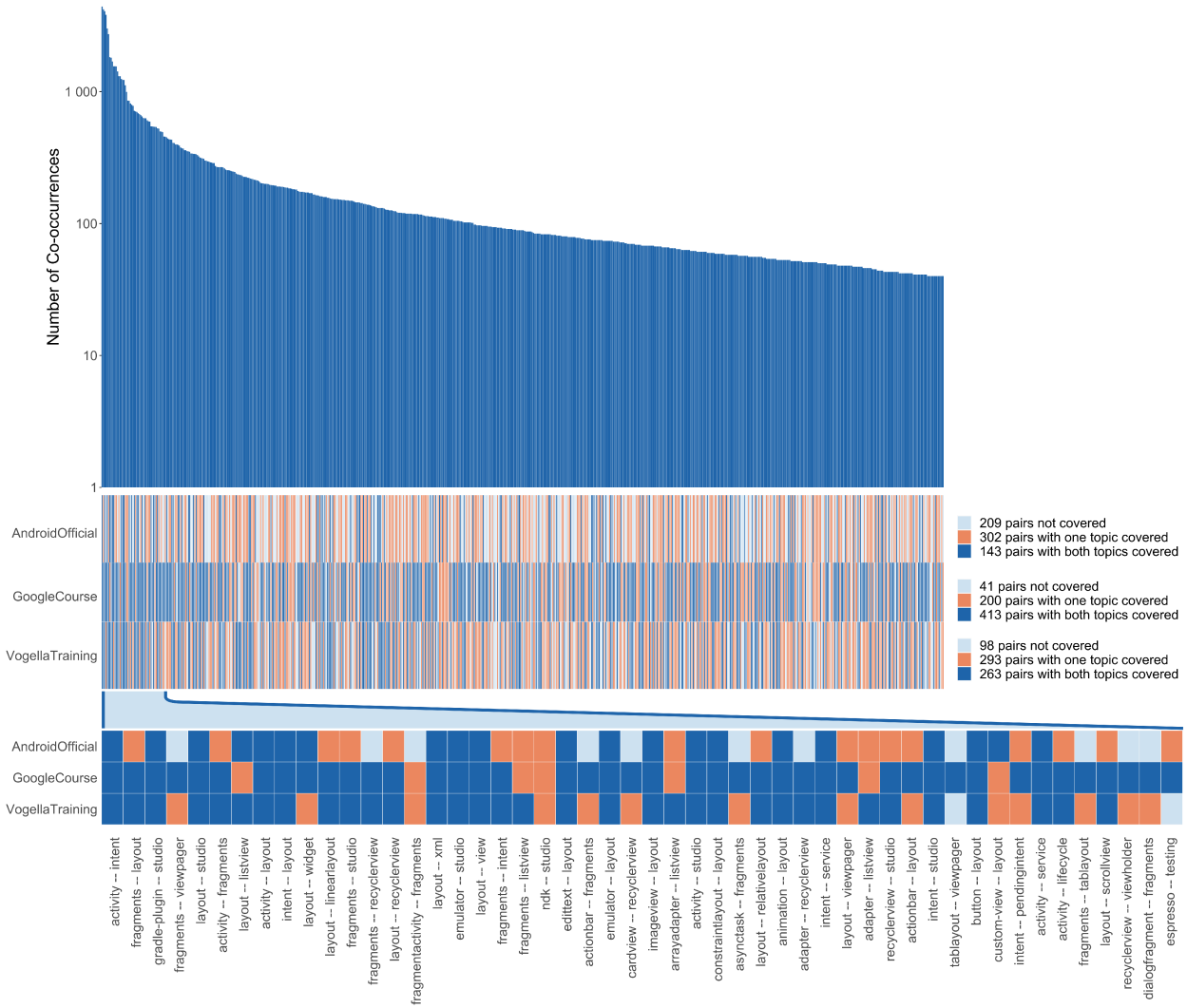
The design of tutorials has evolved over time, for example, by introducing minimalist documentation in response to work on improved usability and readability of shorter, focused texts [138]. This evolution is similar to the evolution of software design, which is supported by conceptual frameworks and tools to systematically assess and document design decisions. However, contrary to software design, no such system exists for tracking tutorial design rationale.

Our investigation of the organization and content of three introductory Android tutorials has revealed many variation points in tutorial design. The lack of a standardized format for tutorials is not inherently problematic, and in fact design variations can be beneficial in tailoring documentation to a specific audience. However, we find that tutorial creators must be careful in the design decision they make. Creators should consider alternative decisions based on evidence of their impact on how different audience will receive the tutorial. Even carefully composed tutorials can be poorly received if its content does not match the information needs of its target audience. Thus, we propose a framework, captured in Tables 4.1 and 4.2, that authors can use to systematically review the design of their tutorials.

To illustrate how these guidelines can be useful, we consider the case of a tutorial creator who is tasked to write a "Getting Started" tutorial to present a novel unit testing library. Their initial draft consists of a series of sections that cover all features of the library, with one feature per section.

Upon reviewing our guidelines, the tutorial creator may realize that their long draft can discourage readers. Thus, they keep only the description of the *core features* in a *sequential* reading order for a quicker introduction. To cater to advanced users, they can refactor sections about *specialized features* into optional *independent* sections, excluded from the main tutorial, and add "Additional Reading" boxes at relevant places in the tutorial to *link* to those sections. They can also decide to cover additional topics based on *external factors*, such as common testing patterns and features of competing libraries. However, to keep the main tutorial more cohesive, they can choose to only place those topics in the optional sections.

While incorporating our guidelines in Tables 4.1 and 4.2, authors must balance trade-offs based on the context and expectations of each tutorial. In the previous example, the tutorial writer may have instead opted to avoid links to optional sections altogether, to avoid possible distractions for beginners and reduce the tutorial creation effort. Both strategies have their merits. Although our study focused on tutorials targeted at an audience of Android programming beginners, our guidelines are applicable to different technologies and audience expertise levels. The guidelines are thus offered as a tool to stimulate additional reflection and encourage a systematic and informed approach to tutorial design.



The 654 pairs shown account for over 75% of all occurrences. The top section shows the popularity of each pair, the middle section indicates how many topics of each pair is covered by each tutorial, and the bottom section zooms in on the 50 most frequent pairs (stripping the android- prefix of each topic).

Figure 4.2: Coverage of Each Tutorial for the 654 Most Common Topic Pairs

Chapter 5

Casdoc: Code Examples with Interactive Annotations

The previous chapter presented several decisions used to select and organize the content of professionally created tutorials. Although we highlighted different strategies to create tutorials, we did not gather empirical evidence of the behavior of readers when faced with each design variation. As the next and final part of this thesis, we delve deeper into the design of a documentation *format* (i.e., the *organization* aspects of a document) for learning resources for the usage of unfamiliar application programming interfaces (APIs).

Documentation is a crucial asset to understand an unfamiliar software system [77, 189]. However, creating high-quality documents is effort-intensive, as documentation quality is a multi-faceted challenge [98]. Documents must not only contain enough information to address the needs of its audience [51], but the information must also be readable, navigable, and understandable [5, 232]. These aspects, which relate to *how* the content is organized and presented, are necessary to ensure readers can locate the information they seek. Without a carefully designed format, documents that contain a lot of information—which is typically a desirable quality—may fail to emphasize the most useful fragments (e.g., [244]).

The evolution of web technologies has motivated a migration from printed documents to web pages. This migration is an opportunity to revisit documentation formats [93, 221]. Programmers can now find a large amount of learning resources online, including official documentation from software organizations, public forums such as Stack Overflow, community-maintained knowledge bases such as Wikipedia, and a plethora of tutorials on virtually any development technology. New formats have emerged to change how users interact with information systems, such as video tutorials [63, 219]. Researchers have designed new approaches to create [49] and consume [105] video-based content. Others have also investigated augmented reality (AR) and virtual reality (VR)

environments to contextualize information and encourage content exploration (e.g., [111, 117, 123, 237]). More recently, research has shown the potential and challenges of conversational agents as an intermediate layer between users and documents [209, 214]. Software engineering researchers have also proposed different techniques to alleviate the effort and expertise required to create good software documentation by generating content (e.g., [35, 121, 153]) or retrieving it from knowledge bases (e.g., [56, 176, 233]).

Yet, despite the explosion of digital documentation, the structure of information within *text-based* documents has not evolved considerably since the early days of the Internet. Most documents present information as a fixed linear sequence of headers, paragraphs, and supporting elements such as code examples, figures, and tables.

The static format of a traditional web page has several benefits. It is familiar to most writers and readers, and it can be viewed in any web browser, printed, or read by a screen reader. Due to its lack of interactivity, a traditional format also does not require writers to design a user interface, and readers to learn how to use this interface. Nevertheless, an important limitation is that the structure of information does not match how many readers consume the information in a document. Studies have found that programmers often jump between sections of a document, focus on elements such as code examples, and try to only read the content relevant to a specific task [30, 95]. Such behavior was necessary with printed resources, but web technologies create opportunities for interactive documents that better support a variety of reading behavior. Thus, new approaches are needed to improve the way documents present information to programmers.

We present Casdoc, a novel technology for improving the presentation of online learning resources for programmers. Casdoc, which stands for Cascading documentation, presents the content of an HTML document as a graph of concise and interactive annotations rooted in a code example. A transformation tool simplifies the authoring process of these documents by generating them from annotated code files. Casdoc is a solution to improve the navigability of content in code-oriented documents. Specifically, we designed Casdoc for documents that most readers will not read in the same order. For example, readers will read different sections of how-to guides based on the task they want to perform with a given API. In contrast, Casdoc is not optimal for documents where readers are not expected to look for targeted information, such as a first programming tutorial for novices, where a static format with a single linear narrative is preferable (possibly complemented by annotated Casdoc code examples).

In a Casdoc document, readers interact with code elements to reveal further explanations of those elements. Information about elements that are irrelevant to a reader remains hidden to avoid unnecessary distractions. Casdoc relies on popovers and dialogs to achieve this objective. Hence, it recasts two graphical elements which are typically used for secondary navigation aid as the primary structure to organize the content of a document. As a result, this strategy splits the content of a document into concise annotations. Annotations are created by the document's author, who inserts

them directly into working code files as code comments. This authoring process is similar to the generation of API reference documentation from header comments, but supports a different type of documentation (i.e., tutorials and other learning-oriented documents). By writing each explanation in isolation, authors do not need to concern themselves with the narrative flow of the document. The Casdoc transformation tool then converts the annotated code files into dynamic web documents.

Researchers have proposed other techniques to add interactive elements to existing static documents, such as data visualizations [17, 133, 134], custom annotations [96], and automatically generated links to external resources [16]. Others have suggested to make document visualization software more interactive with navigation features inspired by paper-based formats [202, 207]. Specifically for software documentation, explaining why specific code examples are matched can help programmers decide on the pertinence of the code search results [228]. Digital documents can also contain dynamic elements, such as runnable code examples [144, 222], explorable statistical analyses [61], or modifiable machine learning models [22]. All of these techniques, however, do not change the linear organization of information in existing documents.

With Casdoc, we challenge this traditional structure. Through the design and implementation of the format, which does not require specialized software, as well as the transformation tool, we explored alternative solutions for interacting with documentation. The design of Casdoc is guided by principles elicited from prior work on document usage behavior.

The prototype implementation served as an instrument to study how programmers react to a non-linear format. First, we conducted a seven-month field study with 326 participants and 126 documents. Participants were undergraduate students enrolled in a programming-intensive software design course, who used the documents to learn professional software design know-how. We collected and analyzed over 18 000 participant actions on the documents to assess the strengths and limitations of Casdoc in an ecologically valid environment.

We complemented these *quantitative* results with *qualitative* navigation patterns observed during a laboratory study. The laboratory study consisted of asking participants to solve a series of implementation tasks using an unfamiliar API during a one-hour session. Participants had only access to a limited set of documents that we authored specifically for the tasks. Each document showed information both in the Casdoc format and in a non-interactive textual representation. We analyzed the audio and video recordings of the participants' use of the documents during the sessions to assess how Casdoc supported or hampered recurrent navigation patterns.

Based on our results, we propose guidelines for designing new code-oriented documentation formats and improving existing ones. We also leveraged these findings and feedback received on preliminary versions of this work [154, 158] to design an improved version of Casdoc. Finally, we released a set of public learning resources for software design that use the Casdoc format. Hence, we make the following contributions:

1. the description of an interactive and non-linear format for software documents, which addresses common documentation issues (Sections 5.1 and 5.7);
2. an analysis of five relevant document design factors, based on prior work (Section 5.2);
3. a complete methodology for the design of a field study that maximizes the ecological validity and reliability of the results in a context where the investigators have authority over participants (Section 5.3);
4. the complete material to perform controlled experiments with programming tasks that elicit meaningful information needs within a short time period (Section 5.5);
5. the results of our studies, synthesized into guidelines for designing new software-oriented documentation formats and insights into the combination of multiple formats (Sections 5.4 and 5.6).

Publications Preliminary stages of the Casdoc project were presented at two conferences. The initial implementation of Casdoc was published in an article titled *"Casdoc: Unobtrusive Explanations in Code Examples"*, presented at the *International Conference on Program Comprehension* in 2022 [154]. Preliminary results of the field study were published in an article titled *"A Field Study of Developer Documentation Format"*, presented at the *CHI Conference on Human Factors in Computing Systems* in 2023 [158].

A comprehensive description of Casdoc, including the field study design and results, is presented in an article titled *"Non Linear Software Documentation with Interactive Code Examples"*, currently under review for the journal *ACM Transactions on Software Engineering and Methodology* [160]. The details of the laboratory study are presented in an article titled *"Evaluating Interactive Documentation for Programmers"*, currently under review for presentation at the conference *ACM Designing Interactive Systems* in 2024 [162].

Study Data The material necessary to replicate the field study is publicly available from the Casdoc project's web page, at https://www.cs.mcgill.ca/~martin/casdoc/. The page includes a link to a free online service to convert annotated code files into the preliminary version of Casdoc used during the field study, with a detailed description of the annotation syntax. The page also includes links to the course textbook, which has a public companion website, and to the annotated code examples used in the study. The annotated code examples have been updated to the newest version of Casdoc, but their content is similar to what was available to participants during the field study. We do not publicly release the database of interaction events collected during the field study to protect the participants' information.

The material necessary to replicate the laboratory study is publicly available as an online artifact [161]. The artifact contains the set of documents, task instructions, and programming environments. It also includes the qualitative annotation guidelines and results of each phase of our analysis.

Appendix C presents the content of the Casdoc project's web page and of the data artifact in detail.

5.1 The Casdoc Documentation Format

Casdoc is a *presentation format* for online programming learning resources. Casdoc documents present a central code example, with additional explanations as interactive annotations. Authors create documents by writing regular source code files and inserting explanations in-place as code comments. The Casdoc *transformation tool* then converts the annotated code files into interactive web documents. Our implementation currently supports code examples written in the Java programming language.

Casdoc is designed for learning resources that focus on the implementation of programming concepts, such as programming forum posts and tutorials. It can demonstrate how to use a programming technology or the realization of programming concepts such as design patterns. In contrast, Casdoc is not intended for internal developer documentation and documents that focus on theoretical concepts.

5.1.1 Presentation Format

Figures 5.1 to 5.5 present five views of a Casdoc document. The initial view of the document shows only a central code example, which acts as the *root* of the document. For example, Figure 5.1 shows a code example that illustrates how to use Java's cryptography application programming interface (API) to encrypt a message.

Additional explanations of the root code example are placed in *annotations*. Annotations are interactive elements that are overlaid on top of the code example. They are hidden in the initial view of the document. Readers can selectively reveal the annotations that contain information relevant to them, then hide them again once they no longer need the information. Annotations can include any kind of content, such as textual explanations or definitions of relevant concepts, alternative code examples, sample input and output data, and figures.

Each annotation is associated with a specific code element, called its *anchor*. Anchors have visual *markers*, which indicate the presence of additional explanations to the reader. The anchor of an annotation can be any string of text on a single line (*inline anchor*) or any continuous set of lines (*block anchor*) in the code example. Figure 5.2 shows an annotation, anchored on the keyword

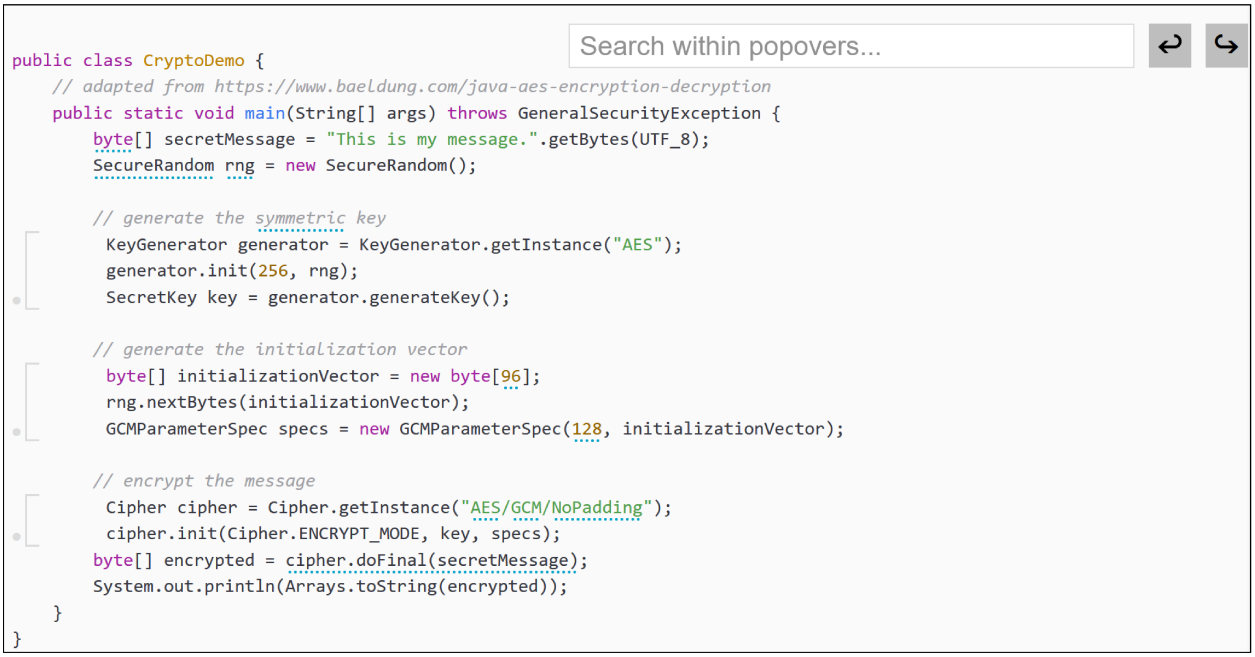


Figure 5.1: Initial View of a Casdoc Document. A reader sees only a code example with subtle markers when opening a new Casdoc document, in this case about Java's cryptography API.

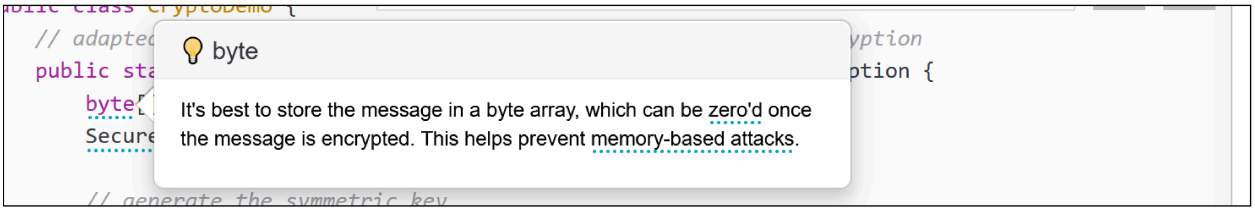


Figure 5.2: Revealing a Floating Casdoc Annotation. When inspecting the code example, a reader may wonder why the input is provided as a byte array instead of a character string. The reader can hover over the keyword byte to reveal an explanation.

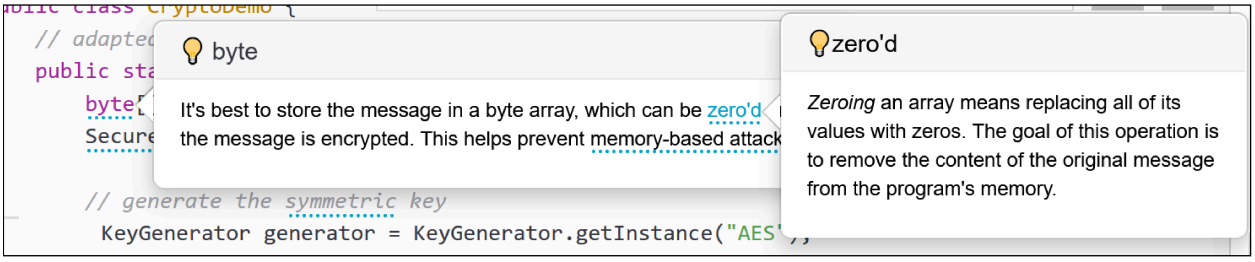


Figure 5.3: Revealing a Nested Floating Casdoc Annotation. Annotations can contain markers themselves that point to further nested annotations. Here, a reader who is confused by the phrase "zeroing an array" may reveal a definition of this concept.

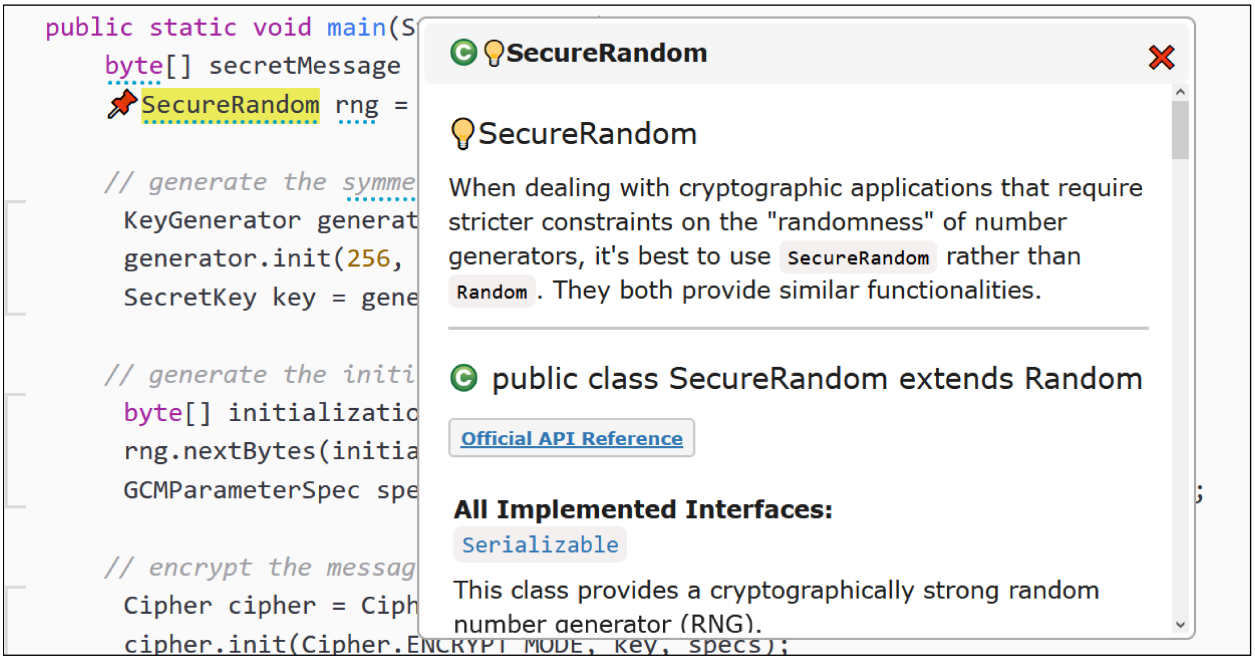


Figure 5.4: Revealing a Pinned Casdoc Annotation With Original Content and API Reference Documentation. Pinned annotations remain visible until a reader closes them. Here, a reader may want to pin the annotation about the SecureRandom class, which contains both original information and its API reference documentation, to refer to it later.

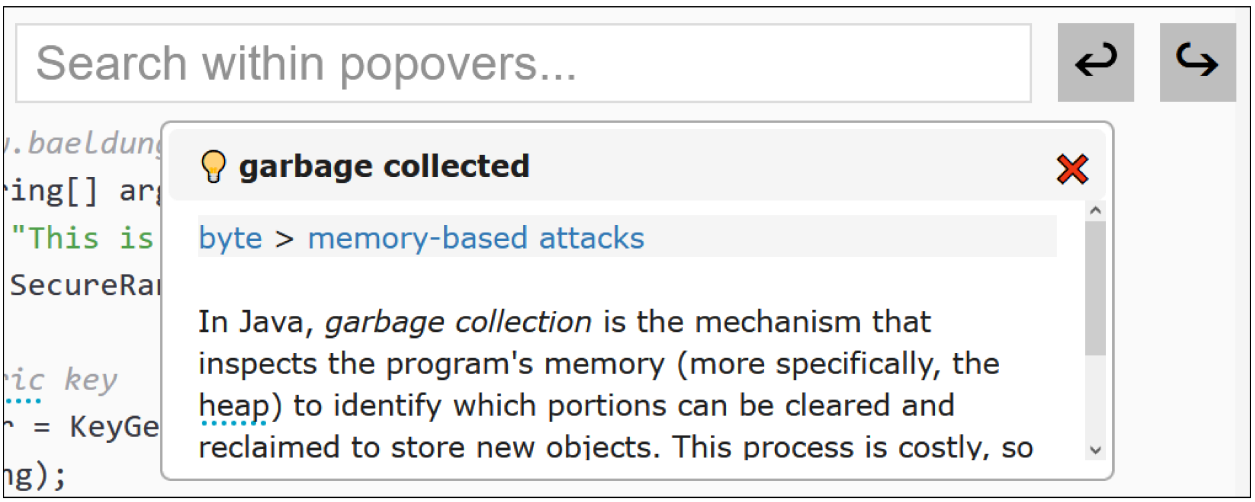


Figure 5.5: Secondary Navigation Tools Included with Casdoc. Readers have access to a search bar to find content within hidden annotations, breadcrumb trail to identify the parents of nested annotations, and buttons to undo and redo annotation pinning actions.

byte, that explains why the original message is stored in a byte array as opposed to a String object. Some annotations can be associated with multiple anchors, for example when an important code elements appears multiple times.

The anchor of an annotation can also be a string of text *inside* another annotation, such as the mention of an important concept. In this case, the annotation that contains the anchor is the *parent* annotation, and the annotation that the anchor links to is a *nested* annotation. Figure 5.3 shows a nested annotation that defines the expression "zeroing an array", which appears in the annotation about the byte array. Nested annotations can themselves contain other nested annotations.

Readers can view annotations in two forms. Hovering over an anchor reveals a *floating* annotation, which disappears when the reader leaves the area of the anchor and its annotation (Figure 5.2). Floating annotations allow readers to read, then discard, many explanations only by moving their mouse, without cluttering the document. Clicking on an anchor *pins* the annotation, keeping it visible until the reader clicks again on the anchor (Figure 5.4). Readers can move and resize pinned annotations. Pinned annotations allow readers to revisit later information that they find useful. Thus, by moving and resizing pinned annotations, readers can organize the content of a document in a way that fits their preferences and needs.

Typical annotations come from the comments inserted by the document's author in the annotated code file. However, the Casdoc transformation tool automatically creates additional annotations with the official API reference documentation for standard Java types and their members (*Javadoc* annotations), anchored on the type or member's name in the code.[1] By contrast, annotations created by the document's author are referred to as *original* annotations. If the anchor of a Javadoc annotation overlaps with the anchor of an original annotation, the two annotations are combined into a single one, with the two fragments clearly separated. Figure 5.4 shows an annotation that contains both the rationale for using the SecureRandom class instead of the more usual Random to generate numbers, and the reference documentation of that class.

To help readers orient themselves across the graph of annotations, Casdoc includes several visual aids and navigation tools. When an annotation is pinned, a pin icon appears beside its anchor, and the anchor is highlighted when the reader hovers over the annotation (Figure 5.4). Pinned nested annotations show a *breadcrumb trail* to indicate their parents and allow readers to open them (Figure 5.5). Readers can also use a custom *search bar* to search among the content of all annotations.[2] Finally, readers can *undo* and *redo* pinning and unpinning actions, e.g., in case they accidentally close a nested annotation and forgot where the anchor was.

[1]The anchor of Javadoc annotations are not indicated by markers, to avoid too many anchors and because the presence of the annotation is predictable.

[2]The native search feature of web browsers cannot reveal or find text in hidden annotations.

```
// adapted from https://www.baeldung.com/java-aes-encryption-decryption
public static void main(String[] args) throws GeneralSecurityException {
    /*?
     * Type: Keyword
     * Anchor: byte
     * ---
     * It's best to store the message in a byte array, which can be zero'd
     * once the message is encrypted. This helps prevent memory-based attacks.
     *
     * +++
     *
     * Type: Internal
     * Anchor: zero'd
     * Parent: byte
     * ---
     * *Zeroing* an array means replacing all of its values with zeros.
     * The goal of this operation is to remove the content of the original
     * message from the program's memory.
     */
    byte[] secretMessage = "This is my message.".getBytes(UTF_8);
    /*?
     * Type: Keyword
     * Anchor: SecureRandom
     * ---
     * When dealing with cryptographic applications that require
```

Figure 5.6: Example of the Annotation Language to Create Casdoc Documents. The code comments will generate the two annotations shown in Figure 5.3.

5.1.2 Authoring Process

To create a Casdoc document, an author starts by providing the root code example in a new Java file. The author then inserts annotations in code comments next to their anchors. Embedding annotations in a code file provides a format familiar to programmers. It ensures that the root document can be created and maintained using common development tools, such as integrated development environments (IDEs) and version control systems (e.g., Git). Each Java file will be converted into a separate Casdoc document.

Authors use a special syntax to distinguish code comments that contain annotations from regular comments to keep in the final document. Figure 5.6 shows an example of these special comments to generate Casdoc annotations. Each annotation comment is enclosed between the sequences /*? and */, and may contain multiple annotations. Within a comment, each annotation starts with the declaration of its anchor, followed by its content. Authors can use the Markdown syntax to create visually rich annotations [82]. After inserting the desired annotations, the author uses the transformation tool to convert the annotated Java file into the final Casdoc web document.

The special syntax for declaring annotations does not interfere with the original Java code. Therefore, the annotated files can be validated for syntax and symbol resolution by any compatible compiler

5.1.3 Implementation

Casdoc documents are self-contained. The HTML file generated by the transformation tool contains all declared annotations, using dedicated HTML elements to identify them and their anchors. The visual elements and interactive aspects of the format are implemented with client-side CSS and JavaScript assets, which themselves only rely on mature libraries.[3] Hence, Casdoc documents can be viewed in any software that supports standard web technologies and can be deployed easily without requiring a complex server infrastructure.

The transformation tool is implemented as a Java program. It relies on a Java-specific parsing and symbol resolution library to extract custom annotations from code comments and Javadoc annotations related to standard types and members. A preliminary version of the tool is available as a free online service from the project web page.[4]

5.2 Key Properties of Casdoc

Documentation formats can vary across numerous dimensions, such as the length of code examples, the interplay between text, figures, and code, or the use of external resources as integral or peripheral information sources [87] (and as shown in Chapter 4). The creation of Casdoc involved many decisions, from core design principles to technical implementation details. Not all of those decisions, however, have the same impact on the ways readers find information in documents.

We identified five properties of Casdoc that are key components of the format. Casdoc documents

1. focus on code examples,
2. gradually reveal information,
3. split information into small fragments,
4. use explicit hints about the information structure, and
5. integrate content from external sources.

[3] The web assets can optionally be embedded in the Casdoc document, to make it a truly independent HTML file.

[4] https://www.cs.mcgill.ca/~martin/casdoc/

The properties are informed by prior work on programmer information needs and reading behavior. Each property thus corresponds to a hypothesis, namely that the property will help readers locate the information they need within a document. We used these properties to scope our evaluation of Casdoc. This scope focuses our findings on aspects of documentation that can be found in other existing formats or that can be used to design new ones.

We present each property with the prior work that supports it, its realization in Casdoc, and other examples of the property found in existing documentation. We also discuss potential limitations of the property, or contexts in which it may be detrimental to a document's quality. Table 5.1 shows an overview of the properties across a sample of documentation sources and formats. Those examples demonstrate alternative implementations of the properties. We note that the presence or absence of a property does not correlate with the overall quality or usefulness of the documents. In particular, the official Java tutorials (second row), despite being high-quality documents, do not exhibit any of the studied properties. We selected all resources as examples of good documentation. The objective of this work is to explore various strategies for presenting information, rather than try to rank different formats.

We discuss these properties as they apply within the context of a single document, i.e., a single web page. The organization of documents within a set is outside the scope of this work.

5.2.1 Focus on Code

> *The document format emphasizes high-quality code examples that readers can use to understand a concrete application of software development technologies.*

Tutorial authors recognized the importance of good code examples [87] and many of the documentation retrieval and synthesis techniques focus on code examples as the main source of information(e.g., [56, 76, 233]). They capture concrete solutions that anchor the discussion of more abstract concepts or guidelines related to the code. A code example is also a useful starting point for programmers to copy and adapt to accomplish the task they want. For those reasons, programmers commonly choose to first read the code examples of a tutorial, and only refer to surrounding text if they need more information [30]. As a result, documents without code examples, or with code examples that are too simple, are viewed as less helpful by programmers [152, 189]. The benefits of code examples in learning resources are similar, to an extent, to the benefits of video tutorials: they allow the audience to follow along a concrete application of the abstract concepts being discussed [130].

Implementation Casdoc focuses on code by anchoring the hierarchy of annotations in a complete and compilable code example. Other learning resources emphasize code differently. Some tutorials are accompanied by curated sets of standalone examples, intended to integrate the notions described

Table 5.1: Presence of the Five Properties in Documents from Various Sources

Document Source **Link**	**FC**	**GR**	**SF**	**EH**	**EC**
Casdoc https://www.cs.mcgill.ca/~martin/casdoc/	✓	✓	✓	✓	✓
Oracle's Java Tutorials https://docs.oracle.com/javase/tutorial/java/index.html	-	-	-	-	-
Java API documentation (Javadoc) https://docs.oracle.com/en/java/javase/17/docs/api/	-	-	✓	✓	-
Stack Overflow https://stackoverflow.com/questions	-	-	✓	-	-
Android Developer Guides https://developer.android.com/topic/architecture	-	✓	-	✓	-
Amazon API Gateway's FAQs https://aws.amazon.com/api-gateway/faqs/	-	✓	✓	-	-
R Cookbook https://rc2e.com/	✓	-	✓	-	-
Codelets [166] https://dl.acm.org/doi/10.1145/2207676.2208664	✓	✓	✓	✓	-
Adamite [96] https://adamite.netlify.app/	-	✓	✓	✓	-
SISE [215] https://dl.acm.org/doi/10.1145/2884781.2884800	-	✓	✓	-	✓

FC: Focus on Code; **GR**: Gradual Reveal; **SF**: Small Fragments;
EH: Explicit Hints; **EC**: External Content

in the tutorial into solutions for more complex scenarios.[5] Other online resources, such as GitHub Gist,[6] are themselves databases of code examples, often with a minimal description of the example's purpose, which can be used on their own or in combination with other documents. Programming "cookbooks" are a more structured version of code example databases.[7] They typically focus on implementation solutions that the reader can adapt to perform common tasks with a technology. Some online learning platforms also guide readers through the implementation of a small program as the main learning activity.[8] However, although such platforms place a higher importance on code, some readers may prefer to see the complete program first, instead of going through each step of the guide. Finally, Oney and Brandt proposed to embed documentation in shareable code fragments, called Codelets [166]. Programmers create Codelets as an HTML document using specific tags to

[5]E.g., https://www.tutorialspoint.com/javaexamples/index.htm and https://www.w3schools.com/java/java_examples.asp

[6]https://gist.github.com/

[7]E.g., the Python [23] or R [125] cookbooks

[8]E.g., Google Codelabs, https://codelabs.developers.google.com/

identify links between the code example and its related documentation. Although their idea aims at helping programmers integrate code examples found on the web, rather than as a learning resource directly, Oney and Brandt acknowledge the pedagogical potential of Codelets.

Limitations Code examples alone are often insufficient to describe complex programming tasks. Documents that focus on code should not entirely omit accompanying explanations, which can help the reader adapt the code example to their situation, distinguish important parts of the code from peripheral elements, or learn related concepts. For example, Stack Overflow answers that contain only code often receive downvotes or edit requests to add some explanation of the code [152].

5.2.2 Gradual Reveal

> *The document format reveals only a small part of the content at a time, letting the reader understand one fragment before showing the next.*

Being overly verbose and containing insufficient information are two common, yet conflicting issues of documentation [5]. Including more information in a document is necessary when the audience is large and varied, as it is often the case for software documents addressed to third-party programmers. However, too much content can have a detrimental effect if it increases the time and effort each reader takes to find the information they need. When readers spend, or estimate that they would spend, too much effort to find the parts of a document they need, they are likely to look for another document [126, 234]. In their idea of the "Minimal Manual", Carroll et al. suggest to "slash the verbiage" [40]: technical writers should reduce the length of manuals by removing redundant and superfluous parts, to avoid readers misusing the documents or missing crucial information. Crowd-sourced documentation platforms in particular, such as Stack Overflow, can accumulate overwhelming content on popular topics. They must find effective ways to emphasize the most important information to readers [244]. Exposing readers to only parts of a document at a time is an alternative solution to this conflict between completeness and verbosity.

Implementation Casdoc gradually reveals its content through annotations in floating popovers or pinned dialogs. The initial view of a Casdoc document shows only the code example, and the reader chooses which annotations to reveal by interacting with their anchors. Collapsible HTML components can also be used for that purpose, allowing readers to choose which information to reveal.[9] Tabbed containers can be useful to include multiple variants of the same content fragment in a document without increasing its bulk. They can show, for example, information to accomplish

[9] E.g., the FAQs document of Amazon API Gateway uses a collapsible element for the answer of each question: https://aws.amazon.com/api-gateway/faqs/

a task with alternative technologies, allowing readers to select the technology that is relevant to them.[10]

Limitations Revealing information gradually inherently relies on a format that can be modified over time, based on the reader's or timed triggers. Thus, this property may be harder to consistently implement or adapt across software applications and viewing devices. For example, the Casdoc format is designed for mouse interaction in desktop web browsers, and does not support well mobile devices, touch-based interactions, or printing. This single supported context limits the usability of Casdoc documents. Furthermore, dynamic documents are not well suited for long-term archival purposes, unless the viewing technology is archived with them. Thus, it may be useful to produce a static version of dynamic documents as a replacement in situations that do not support user interactions.

5.2.3 Small Fragments

> *The document format presents its content as a series of concise fragments that each convey a single self-contained idea.*

It is common for programmers to read a document out of sequence [30]: they may look for a specific section related to their needs, skip information that they already know, or go back to an earlier point in the document to find background information about a concept. A set of concise and decoupled fragments supports such reading behaviors. In contrast, documents composed of vaguely bounded and highly dependent fragments force their readers to read larger sections to contextualize and understand the information they seek, which can create a feeling of verbosity. Additionally, identifying clear fragments in a document can facilitate the reuse of the content into other documentation systems (e.g., [56, 99, 233]) or in integrated development environments (e.g., [166, 176]). This reuse scenario expands the value of the document's information beyond its original purpose.

Implementation Casdoc's annotations encourage authors to partition the information into concise fragments that will be presented in small popovers and dialogs. Annotations can link to further supporting explanations in nested annotations, but they should present a complete idea by themselves. Non-interactive formats can also be organized as small fragments. For example, API reference documentation typically contains one fragment per API type or member.[11] Readers are

[10] E.g., Android Developer Guides uses this strategy to show equivalent code examples either in Kotlin or Java: https://developer.android.com/guide

[11] E.g., the Java API reference documentation (Javadoc), https://docs.oracle.com/en/java/javase/17/docs/api/index.html

not expected to read the entire API documentation to understand the fragment about a particular element. Question and Answer (Q&A) forums often exhibit this property, as answers are typically created as independent fragments.[12]

Limitations Documents that appear too fragmented can irritate readers [216]. Fragmentation can lead to frustration when readers do not know how to find a fragment of interest, or when they need to gather multiple fragments scattered across a document to answer a query. To prevent this problem, authors should organize the fragments carefully to support an intuitive navigation and place fragments that relate to the same task close to each others in the document's structure. Dividing the content of a document into small fragments can also break its narrative flow or disorient readers that do not know what information they must look for. Thus, this property may be detrimental in some contexts, such as online courses for a homogeneous novice audience.

5.2.4 Explicit Hints

> *The documentation format includes explicit hints, distinct from textual cues, to help readers understand and navigate the structure of a document.*

Navigating within the content of a document is an important aspect of information search [174]. Given the amount of web resources readily available and indexed by search engines, readers have a strong incentive to look for other documents if they do not find the information they seek quickly in the current one. Visual hints of the structure and content of a document reduce the cost of within-document navigation and provide a sense of location and control [208]. They can be an especially important tool to mitigate limitations of other properties, such as fragmentation [216]. This property is more incremental than the previous ones. Multiple types of visual hints can be incrementally added to a format to reveal complementary aspects of the structure.

Implementation Casdoc uses markers to indicate the presence of annotations related to a code element or to a concept. Annotations containing only Javadoc information, however, do not have markers as their presence is predictable. The Adamite annotation tool uses a similar strategy to mark parts of a document that have been annotated by readers [96]. Casdoc also uses indicators such as "pin" icons, breadcrumbs, and highlighting to identify the anchor of a pinned dialog. The XCoS code search approach proposed by Wang et al. presents information related to different aspects of the query (e.g., non functional requirements) to help users navigate the list of results [228]. Although those hints are text-based, they constitute a navigation structure distinct from the main list of results. Existing documents also include recurrent types of alternative hints. For example, a

[12]For example, the popular Stack Overflow programming forum, https://stackoverflow.com/questions/

table of contents that remains visible and indicates the current position of a reader as they scroll within a page is useful to convey a sense of location within an overview of the document.[13] API reference documentation uses hyperlinks to relate relevant fragments, such as a function to its parameter and return types.[14] Although hyperlinks are a common feature of many websites, the extremely predictable nature of a link's target in reference documentation makes them an effective mechanism to navigate its structure, as opposed to the arbitrary links in typical documents.

Limitations Structural cues integrated in the main text of a document must be used sparingly. They can bloat the content and dilute its relevant information. Ideally, explicit hints should be clearly separated from the document's content, so that the reader can ignore them once they reached the information they sought. Alternatively, hints that rely only on non-textual elements, such as Casdoc's markers, can easily be distinguished from the content. However, the hints' purpose must be intuitive, or they risk confusing the readers. They also limit the accessibility of a document for some readers. For example, readers using a screen reader would be oblivious to Casdoc's markers, and therefore to all of its annotations. Thus, purely visual hints should be complemented with other navigation cues, possibly embedded in the HTML tag attributes.

5.2.5 External Content

> *The document format provides a systematic way to integrate information from external sources within its original content without corrupting or misappropriating either source of information.*

The extensive prior work on documentation generation and information retrieval (e.g., [121, 176, 215, 233]) constitutes a valuable opportunity to increase the coverage of a document. Formats should be designed to leverage these approaches to reduce the effort of multiple authors documenting similar technologies, similarly to how software development evolved to promote the reuse of software packages (especially when well documented).

Implementation Casdoc automatically integrates API reference documentation as additional annotations. These third-party annotations are identified by special icons and contain a link to the information's source. When the anchor of a third-party annotation overlaps with the anchor of an original annotation, the two are concatenated into a single annotation that clearly distinguishes its two parts. As an alternative, the SISE tool designed by Treude and Robillard integrates information fragments from the Stack Overflow forum at the top of API reference documentation pages [215].

[13] The Android Developer Guides use such interactive tables of contents, https://developer.android.com/guide/components/fundamentals

[14] For example, Java's API reference, https://docs.oracle.com/en/java/javase/17/docs/api/

The imported information is presented in a rectangle overlay, and contains links to the original Stack Overflow posts.

Limitations The trustworthiness, authoritativeness, and tone of imported content can differ from the document's original content and vary between external sources. Thus, including content from various sources can create jarring changes for the reader, which can affect the perceived qualities of the original content. Clearly identifying the provenance of external content can mitigate these issues. Attributing proper credit is also an ethical and sometimes legal requirement.

5.3 Field Study Design

We evaluated Casdoc in a field study with undergraduate students enrolled in a software design course. Throughout the course, participants had access to a suite of Casdoc documents that complemented the course material. We analyzed how they navigated within the content of each document to assess the strengths and limitations of Casdoc.

We sought to answer the following research questions:

RQ 5.1 Is Casdoc a suitable format for creating learning resources for programmers?

RQ 5.2 What is the impact of Casdoc's key properties on the navigation behavior of the readers of a document?

(a) Focus on Code

(b) Gradual Reveal

(c) Small Fragments

(d) Explicit Hints

(e) External Content

RQ 5.1 assessed whether Casdoc is a valuable addition to the documentation landscape. We answered it by offering participants an alternative baseline format, code examples with static code comments, and comparing the adoption of the two formats. RQ 5.2 assessed whether the five key properties described in Section 5.2 help readers find information within a document. To answer these questions, we instrumented the generated documents to log events when participants interacted with Casdoc's features.

5.3.1 Research Method

Our study falls within the *field experiment* category of Stol and Fitzgerald's framework of research methods [203]. We designed the study to prioritize the ecological validity of the results. We

conducted our investigation within a natural setting, i.e., a university course, but manipulated the environment to introduce Casdoc documents. This strategy favors the realism of the setting, while allowing the introduction of new elements, such as the Casdoc format, that do not exist in a purely natural setting.

As participants could freely choose and change the documentation format they used, the study is similar to that of a quasi-experiment with a within-subject design. Beyond the field study, our investigation also integrates some aspects of action research, as we continued to improve the Casdoc format based on our findings, to integrate the new documents as a permanent part of the course material for future students (see Section 5.7).

5.3.2 Participants

The field study took place during two consecutive sections of a third-year undergraduate course on software design with an important programming component. All students enrolled in the course could choose to participate in the study by agreeing to a consent form for the collection of their interaction data.[15]

Students are a subgroup of the target audience for Casdoc, i.e., programmers who are learning software development concepts and the usage of some libraries. Although we do not claim that this sample is representative of all programmers, the participants did not act as proxy for a different population.

Both authors had a teaching role in the first section. This familiarity with the course was crucial to create relevant Casdoc documents. The authors were not involved with the second section.

Given that the investigators had authority over participants of the first section, participants remained completely anonymous throughout both sections of the study. This anonymity was important to avoid an unintentional pressure on students to participate in the study or use a format if they did not feel comfortable. Consequently, we did not collect demographic information to measure specific properties of the sample. However, the population from which the sample is taken is well known. Senior undergraduate students in computer science consists predominantly of young adults with only a few years of programming experience, with a minority having previously done industry internships. Before registering for the software design course, students are expected to be familiar with the programming language of the course, Java, and its standard library, but not necessarily with advanced concepts.

[15]This study was approved by the Research Ethics Board Office of McGill University, file number 21-06-007.

5.3.3 Documents

We used the content of the companion website of the course's textbook [188] to create the corpus of Casdoc documents.[16] The website contains three types of documents, namely lists of exercises, descriptions of their solutions in prose, and 126 code examples: 72 of them implement code described in the textbook (i.e., *chapter code*) and the other 54 implement solutions to the exercises (i.e., *solution code*).

We converted each code example into the Casdoc format and inserted additional explanations as annotations. We did not modify the exercise or solutions, which consist mostly of text. The original code examples sometimes contained code comments. We retained those comments in the converted documents, rather than transforming them into further annotations.

We also converted the annotated code examples to a static baseline format. This format included all new annotations as code comments, but it did not include the API reference documentation for standard Java types to avoid unreasonably large comments. This information is, however, easily accessible via the students' integrated development environments (IDEs) and via the official Java documentation website.

The code examples were available on a public website dedicated to the study. The website showed the first document in the Casdoc format to all participants to encourage them to try the new format and provide a consistent user experience. Once on a document, participants could change between the two formats whenever they wanted to. The website stored the last format used in a browser cookie to open the next document in the same format.

The study website initially contained only the annotated code examples. After observing a low study participation rate during the first section, we added the exercises and solutions, unmodified, to the website. Following this change, the retention rate of participants increased for the second section.

5.3.4 Data Collection Infrastructure

We instrumented the documents to record traces of the participants' activity. Client-side JavaScript functions created the interaction events and sent them to an HTTP POST endpoint of a dedicated data collection server. Only Casdoc documents had interactive features, and therefore generated interaction events. However, the server also recorded document requests in either format and requests to change the format. These events did not rely on client-side functions. For the second section, we also modified the server to record the total number of requests it received, both to

[16] https://github.com/prmr/DesignBook

Table 5.2: Events Collected During the Field Study

Event	Origin	IDs	Details
Visit any page *	server	D	
Consent to study	server	P/S	
Withdraw consent	server	P	
Start new session	server	P/S	
Open code example	server	P/S/D	format
Change format	server	P/S/D	new format
Open/Close annotation	client	P/S/D	annotation ID
Interact with marker	client	P/S/D	marker ID
Use search widget	client	P/S/D	query; selection(s)
Use navigation widgets	client	P/S/D	result

* This type of events was only collected during the second section of the course.

monitor the website's status and to estimate the sampling bias.[17] We report on an analysis of the sampling bias in Section 5.4.3

This logging mechanism was asynchronous and did not affect the user experience. The usage of Casdoc's features did not require a successful exchange with the website, other than fetching the document with an initial GET request. Thus, problems with the data collection infrastructure would not affect students, unless the entire study website was down.

The study website did not require any form of authentication, or ask for personally identifiable information, to preserve the anonymity of participants. To follow events performed by different participants, we stored two HTTP cookies in each participant's browser, in addition to the format-related cookie. Upon consent, participants received a randomly generated 64-bit integer in a persistent cookie (i.e., the *participant ID*). The website also sent a second random integer in a session cookie (i.e., the *session ID*), which was reset every time the browser was reopened.

Table 5.2 summarizes the types of events we collected. The first six types of events are generated by the website, whereas the last four types are generated by JavaScript functions and sent through the HTTP POST endpoint. For each event, the website stored the type of event and a timestamp, as well as the IDs of the participant (P), session (S), or document (D) and the additional details described in the last column.

This procedure was minimally intrusive to participants. After providing their initial consent, the participants did not see the data collection mechanisms while using the documents. This was a deliberate choice to avoid constant reminders that participants were observed, which could affect their behavior. As a consequence, we did not rely on tools such as pop-up dialogs or surveys to

[17] We did not observe any unusual access patterns, except for a large number of requests to the website's home page: there were almost six times as many requests to the home page as the number of requests to all code examples combined. These requests could be due to web crawling or server maintenance bots.

Table 5.3: Summary Statistics of the Collected Data

Property	**Section 1**	**Section 2**	**Total**
Study length (days)	104	102	206
All document requests *	–	19 594	–
Code example requests *	–	14 644	–
Enrolled students	165	321	486
Participants	124	202	326
Sessions	176	541	717
Opened code examples	827	6511	7338
Logged interactions	2795	15 570	18 365

* including from non-participating students

gather the feedback of participants. Thus, although we did not collect insights about the subjective perceptions of participants on Casdoc, the collected data is more likely to represent the true behavior and preferences of participants.

5.3.5 Data Preparation

Table 5.3 gives an overview of the data we collected. In total, 326 participants generated over 18 000 interaction events. They consulted the 126 code examples a total of 7338 times. We collected more data during the second section of the course, partly due to the larger enrollment and to the changes to the study website made between the two sections. However, both data sets show the same trends (see Section 5.4.3).

We reassembled the flat list of events into a meaningful structure to analyze our results. The actions of each participant are split into *sessions*, i.e., a period of continuous usage of the website. During a session, a participant *viewed code example documents*. Participants performed different actions on code examples, such as *viewing an annotation* and *using the search widget*. Based on a preliminary inspection of the data, we considered all events performed within 15 minutes of consenting to the study as part of a *learning period*. We excluded the data of all participants who did not interact with the website beyond their learning period.

We initially split sessions using the session ID cookie. However, we found that the cookie was unreliable to track continuous usage. Some participants rarely closed their web browser, creating sessions that spanned many days or weeks. We split such long sessions whenever a participant did not generate any event for two consecutive hours.[18] Within each session, opening a document initiates a new code example view, and all subsequent actions performed on this document are associated with

[18]We chose the threshold of two hours based on the distribution of time between two consecutive events with the same session cookie. Nevertheless, as we did not observe a significant drop in the distribution, this threshold is only approximative. We avoid relying too much on sessions in our analysis.

this view. After artificially splitting long sessions, any document that remained opened initiates a new code example view in the second part of the session if the participant performed any action on the document. A single session can contain multiple views of the same document, if it is closed and reopened, and multiple views of different documents can overlap.

We grouped successive events associated with the same Casdoc annotation as a single annotation view action. Each annotation view starts with zero or more hovering events, optionally followed by a pin event, and a final optional unpin event. We grouped together multiple hovering events if they were less than five seconds apart, to account for participants accidentally moving outside the marker and immediately going back to it. To avoid spurious events, a hovering event was generated only if the participant hovered for at least one second over a marker.

As each keystroke in the search widget generated a new event, we grouped all events that incrementally built towards a single search query, as well as subsequent interactions with the search results, as a single search action. Each use of the breadcrumbs and the undo and redo buttons constitutes a separate action.

5.3.6 Study Design Trade-Offs

There is an inherent trade-off between the realism of, and control over, the study setting. As the field study favors realism, we could not control when or for how long participants used the documents. Field studies also lack the control of confounding factors that the sterile environment of laboratory experiments provides. Thus, the decision to conduct a field experiment limited the precision of our measurements and generalizability of our results [203], but produced concrete insights that are directly applicable to an existing context. These insights led to the improvement of Casdoc for future students enrolled in the course.

Another early decision point was the choice of the environment in which to conduct the study. We chose to study students enrolled in a university course. Alternative options included looking for programmers outside our organization, such as professional developers, or using remote experimentation platforms such as Prolific.[19] The effort involved in recruiting participants for a long-term study (several months) and creating a realistic environment in which participants would need learning resources was a deciding factor for choosing the university course. A consequence of this decision was the need to mitigate potential pressures on students. We thus designed the data collection to be anonymous and minimally intrusive, which was consistent with the choice of our research method. Recruiting students as participants also narrowed down the sampling frame of our study. Thus, our results are specific to a well-defined subset of Casdoc's target audience, and more experimentation is required to generalize them to more experienced programmers.

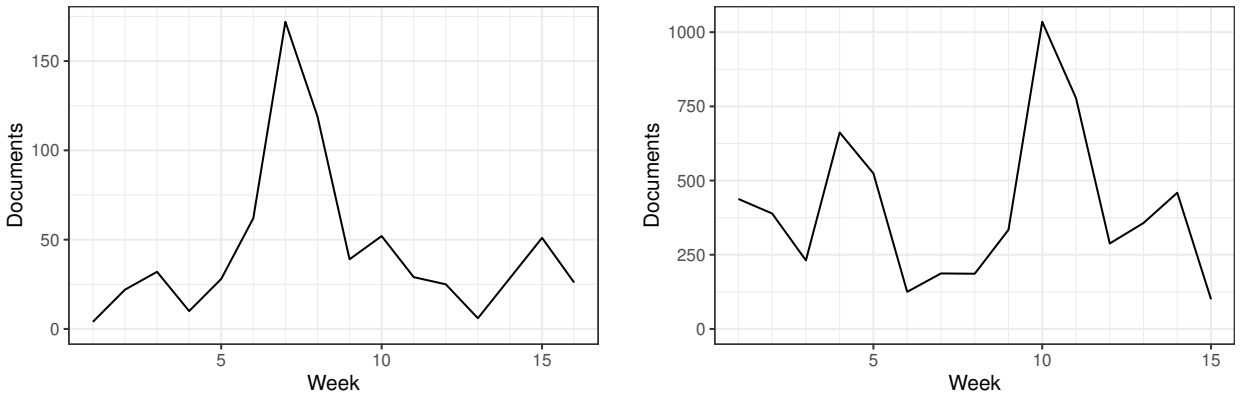
[19] https://www.prolific.co/

The number and choice of document formats to compare was also an important decision. Alternative formats include presenting the additional explanations in a narrative text above, below, or interleaved with the code example, as well as presenting a varying number of explanations in static documents. Offering more formats to participants can help contextualize our observations. However, each format requires a considerable effort to produce, and too many formats can overwhelm participants. We chose to offer one baseline format to have at least one point of comparison for Casdoc. We selected static, commented code examples for the baseline as it is conceptually the closest to Casdoc. We did not vary the content of documents to avoid students missing some relevant information due to their choice of format.

Regarding the collection of events, there is a trade-off between the reliability of the events and the quality of the user experience. Asynchronous client-side functions increase the risk of losing some data, e.g., due to students with an unreliable Internet connection. HTTP cookies can generate inconsistencies in the data, e.g., due to students clearing their cookies during the course. The website's HTTP POST endpoint was also vulnerable to potential attacks from malicious actors, who may try to send fake events. We accepted these risks to improve the user experience, to honor our responsibility to create a suitable learning environment for students in the course. The features of Casdoc were not affected by corrupted cookies or the latency of the study website, as they would have if the content of annotations was retrieved using synchronous requests, for example. To limit and detect the generation of corrupted events, we ensured that key events were generated by the website, such as new document requests. For example, events describing interactions with a document that was never opened would reveal some inconsistencies. We also devised a strategy in which the type of event in the client-side scripts would be encrypted based on the session and participant IDs to make it harder to send undetected fake data to the POST endpoint. We found no inconsistency in the collected data.

5.4 Field Study Results

Figure 5.7 shows a timeline of the participants' activity through the study, and Table 5.4 presents an overview of the main study artifacts and observations. Although we excluded 122 short-term participants (37.4%, see Section 5.3.5), we retained interaction data related to 6770 document views by the others. As participants viewed the large majority of documents (96.1%) in the Casdoc format, the data shows clear usage patterns for the different features of Casdoc, but only limited insights into situations that Casdoc does not support well. We present these patterns in Section 5.4.1, and discuss their implications on documentation formats in Section 5.4.2. We discuss the aggregated data from both sections together, as participants from the two groups exhibited the same patterns. In Section 5.4.3, we report on an analysis of the differences between the two sections and of the sampling bias.



(a) Section 1

(b) Section 2

Figure 5.7: Number of Code Examples (Documents) Accessed by Participants During Each Section of the Course

Table 5.4: Summary Statistics of the Data After Preprocessing

Property	**Section 1**	**Section 2**	**Total**
Participants	54	150	204
Sessions	155	1060	1215
Unique code examples	123 *	126	126
Code example views by all participants	677	6093	6770
Code example views in Casdoc format †	670	5836	6506
Unique original annotations	417	417	417

* For technical reasons, three code examples were not available during the first course section. We fixed this issue for the second section.

† Excluding views during which the participant changed format

5.4.1 Casdoc Usage Patterns

Tables 5.5 and 5.6 present the detailed findings of the field study, grouped according to our research questions.

RQ 5.1. Viability of Casdoc The majority of participants (178 of 204, or 87.3%) used only the Casdoc format throughout the course. Considering that participants had no explicit incentive to use Casdoc rather than the baseline format, this observation strongly suggests that Casdoc is suitable for presenting annotated code examples. Furthermore, among the 26 participants who tried both formats, only six (23%) retained the baseline until the end of the course. Most of the participants who reverted to Casdoc did so within the same session.

Table 5.5: Metrics and Results by Research Question

RQ	Question/Metric	Sec. 1	Sec. 2	Total
5.1	*Viability of Casdoc*			
	Participants who used only Casdoc	49	129	178
	Participants who tried the baseline format	5	21	26
	... only during the learning phase	1	5	6
	... for only one document after the learning phase	2	8	10
	... for only one session (2+ documents) after the learning phase	1	3	4
	... for multiple sessions	1	5	6
	Participants who changed to the baseline format more than once	0	3	3
	Participants who kept the baseline format until the end	1	5	6
Findings: Most participants only used Casdoc. Most of those who tried both formats changed back to Casdoc within the same session.				
5.2a	*Focus on Code Examples*			
	Server-side code example requests	—	14 644	—
	... chapter code	—	8857	—
	... solution code	—	5787	—
	Server-side exercise statement requests	—	2539	—
	Server-side solution description requests	—	2411	—
	Solution code to description requests ratio	—	2.4	—
	Average number of links to solution code per solution description	—	3.8	—
Findings: Participants found value in documents centered around code examples, to support the rest of the course material.				
5.2b	*Reveal Information Gradually*			
	Annotation views	356	1889	2245
	Participants who used annotations	35	115	150
	% annotated document views with 1+ annotation view(s)	18.8%	15.6%	15.9%
	% markers in the code example interacted with (average, by participant)	9.3%	9.1%	9.1%
	% unique original annotations viewed by at least one participant	23.0%	60.9%	—*
	* We did not aggregate the coverage of unique annotation over the two sections as the documents changed slightly between the sections.			
Findings: Most participants used annotations to find further information about elements of the code examples, but only for a minority of the documents they looked at.				
5.2c	*Split Information into Small Fragments*			
	Annotation views	356	1889	2245
	... viewed by only hovering on the anchor	311	1632	1943
	... viewed by clicking on the anchor	43	227	270
	... viewed without interacting with the anchor	2	30	32
	Original annotation viewed from the anchor	228	1385	1613
	... with a nested anchor	41	131	172
	... with an anchor in the code example	187	1254	1441
Findings: Participants mostly viewed annotations in floating boxes, suggesting that they can grasp the information quickly. Participants viewed nested annotations at a relative rate similar to top level annotations.				

Table 5.6: Metrics and Results by Research Question (Continued)

RQ	Question/Metric	Sec. 1	Sec. 2	Total
5.2d	*Structure Information with Explicit Hints*			
	Breadcrumbs used	0	1	1
	Undo/redo buttons used	0	1	1
	Search queries	4	208	212
	... where the participant hovered over the results without selecting one	0	32	32
	... where the participant selected at least one result	2	20	22
	Participants who used the search bar at least once	3	43	46
	Document views with at least one search query	3	159	162
	% inline (vs block) markers seen by participants (average)	57.8%	61.1%	60.3%
	% inline (vs block) markers interacted with by participants (average)	85.6%	86.9%	86.6%
Findings: Participants did not rely often on secondary navigation aids, suggesting that the markers are effective navigation hints. However, small differences in the visual appearance of markers impacted their effectiveness.				
5.2e	*Support the Integration of External Content*			
	Unique annotations in all documents	1565	1529	—
	... with only Javadoc content	1148	1112	—
	Annotation views	356	1889	2245
	... with only Javadoc content	128	485	613
Findings: API reference documentation augmented code examples with a considerable number of annotations without additional effort. These imported annotations were used by participants, representing a quarter to over a third of all annotation views.				

We investigated the documents that triggered a format change and the participants that chose the baseline format over Casdoc to identify scenarios that Casdoc does not support well. However, we found no clear trend in the type of documents (i.e., chapter code or solution code), number of annotations in the documents, or whether the document was among the first ones read by the participants. For example, some participants switched to the baseline on their first document after the learning period, whereas others only tried the baseline after having read over 80 documents already.

RQ 5.2a. Focus on Code Example As all documents presented a central code example in both Casdoc and the baseline format, we could not measure the impact of this property based on the interaction with the code example. Instead, we compared the number of requests for code examples to requests for other documents to assess the value of code-oriented documents. As the study website did not log requests during the first section, this analysis relies only on the second section of the course.

Figure 5.8 shows the number of weekly requests for each document type. Students looked at code examples, in particular chapter code, almost three times as often as exercise and solution descriptions. The number of requests fluctuated over time, but the usage of code examples was strongly correlated to the exercise and solution descriptions (Pearson's $r = 0.93$ between the distributions of weekly

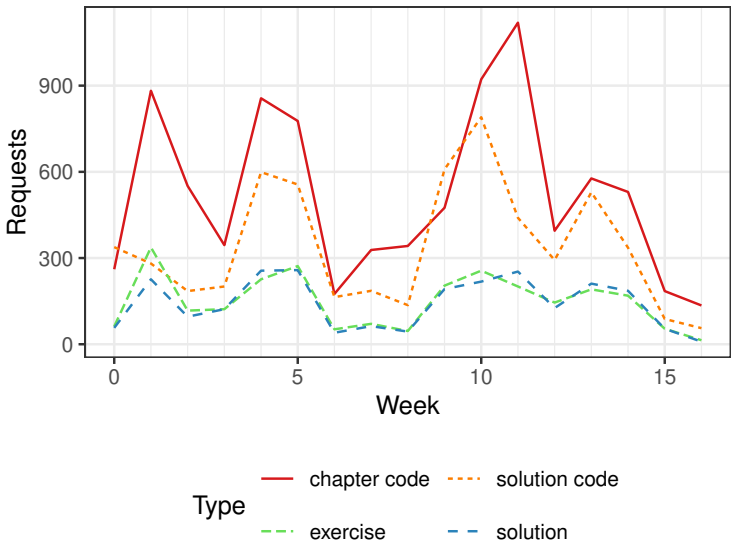


Figure 5.8: Requests to Each Type of Document Received by the Study Website During Each Week of Section 2

requests), suggesting a consistent usage of all types of documents.[20] Comparing the solution code and description requests can provide more detailed insights into the usage of text-oriented and code-oriented documents. There were nine solution descriptions (i.e., one document per chapter), each linking to an average of 3.8 solution code examples. Each solution description document presents the complete solution to all exercises, with smaller code fragments to illustrate the main part of a solution. However, we still observed that, for each chapter, participants requested solution code examples 2.4 times more often than solution descriptions on average. This ratio did not vary considerably per chapter, even for the two chapters where the solution descriptions did not include any link to solution code (ratios of 2.1 and 1.7). For each chapter, the ratio of solution code to solution description requests ranged from 0.9 and 1.5 (first two chapters) to 4.1 and 3.0 (chapters four and five, for which the description had the most links). The consistent popularity of code examples demonstrates their value as part of the learning material.

RQ 5.2b. Reveal Information Gradually Participants did not often look at the content of annotations in a code example. Although 150 participants (73.5%) used annotations at least once, they looked at any annotation in only 15.9% of the code examples they consulted (excluding code examples that did not contain any annotation). Furthermore, considering only annotations with a visible marker in the code example, i.e., annotations that are not nested inside others or

[20] Additionally, we did not find evidence that usage of code examples decreased significantly over time (Kendall's $\tau = -0.242$, $p = 0.175$ [104]).

Javadoc annotations, participants opened only 9.1% of the annotations they saw. This suggests that annotations, a key feature of Casdoc, was not systematically used by participants.

However, we did not expect, or intend, participants to open most of the annotations they saw. One objective of Casdoc is to be able to contain a large amount of information, including fragments that may only be relevant to a small fraction of the audience, without distracting most readers. Hence, although the low usage of annotations may suggest that readers are more likely to miss information in annotations, it also suggests that the gradual reveal of information is effective to avoid overwhelming readers with information that is less pertinent to them. The annotations viewed by participants varied, as they collectively viewed 23.0% and 60.9% of all original annotations (including nested annotations) during the first and second section, respectively.[21] This further suggests that participants effectively adapted the information they saw to their needs, and that the authoring effort of annotation was not wasted.

Finally, looking at each participant individually, we also observed some differences in their behavior. In particular, some participants used annotations much more than the average. For example, ten participants consulted five or more annotations on at least 10% of the code examples they looked at (excluding code examples with no annotation at all), and four participants interacted with more than half of all the visible markers anchored in the code examples. This observation reinforces the hypothesis that Casdoc can adapt the content of a document to individual readers.

RQ 5.2c. Split Information into Small Fragments When viewing an annotation using its anchor (instead of using a navigation tool), most of the time (87.8%), participants only viewed the annotation in its floating form, i.e., by hovering over the anchor, rather than in its pinned form. Floating annotations are intended for faster interactions with the content of the document, thus suggesting that concise fragments allow readers to quickly grasp the key information in an annotation or identify that it is not relevant to them.

Fragmentation can also make it hard for readers to collect all the information they need. The ratio of nested to top level annotations views was 0.119, which is comparable, and even slightly higher, than the ratio of code example markers participants interacted with (see RQ 5.2b).[22] Thus, we reach a similar conclusion as for RQ 5.2b, that the fragmentation can mitigate distractions to readers, but also increases the probably that a reader will miss useful information.

RQ 5.2d. Structure Information with Explicit Hints Participants rarely used the secondary navigation aids. We observed only one instance of a participant using the breadcrumbs and the undo

[21]We did not aggregate the coverage of unique annotation seen by all participants over the two sections, as the documents changed slightly during the modifications of the study website.

[22]We consider only original annotations in this comparison, as Javadoc annotations cannot be nested. We also exclude annotation views opened using navigation tools, as they do not discriminate between nested and top level annotations.

and redo buttons. The search bar was used more often, slightly over 200 times, by 46 participants (22.5%) in 162 unique document views (2.5%). In 22 cases (10.4%), the participant pinned at least one of the search results. In an additional 32 cases (15.1%), the participant hovered over the search results to reveal the content of the retrieved annotation, similarly to hovering over an annotation anchor. Hence, the search bar was the most useful of the navigation aids, helping participants find information in 10.4% to 25.5% of cases.[23] This is consistent with Feng et al.'s observation that search widgets help users interact more with interactive visualizations [70]: In a study with 830 participants asked to explore data visualizations, they found that most of the participants who had access to a search bar used it and looked for more diverse information than the control group. However, in our field study, relatively to all annotation views, the search bar was not often used to find information, suggesting that the explicit hints from markers were effective to help participants find relevant annotations.

We observed an interesting difference, however, in the type of markers that participants interacted with the most. Original annotations with a block anchor in the code example were marked by a gray bracket in the left margin of the code example, whereas those with an inline anchor were marked by a blue underline. Of all markers seen by participants (excluding nested anchors, which are always inlined), 60.3% were blue underlines. Yet, the annotations that participants interacted with disproportionately had inline markers (86.6%, sign test comparing the inline anchors viewed to those interacted with, per participant: $p < 10^{-15}$). We suspect that this difference may be due to the visual aspect of the two markers. Gray brackets have a lower contrast with the document's background than blue underlines, and they are often physically farther from the code element of interest due to the code indentation.

RQ 5.2e. Support the Integration of External Content Importing the API reference documentation of standard Java types and members more than tripled the number of annotations available to participants, without requiring any effort from the documents' authors. Javadoc annotations also contributed to a notable fraction (27.3%) of the annotation views. The absence of authoring cost and the concrete benefits strongly encourage the development of techniques to properly import external content in documents. We nevertheless observe that participants viewed original annotations more often than Javadoc annotations, despite their lower number. Thus, imported external content should not dilute the quality and relevance of the original content specially authored for a document.

[23]The modest success rate can be explained partly because we counted successive queries separately. Therefore, if a participant uses N queries before finding the information they need, this will be measured as a success rate of $1/N$. This was necessary as it is impossible to reliably infer whether the information sought by a participant changes between successive queries.

5.4.2 Implications

The field study showed that programmers are willing to try different types of documents, which encourages the design of new, interactive formats. From our observations, we derived several guidelines that documentation creators can consider when formatting documents. These guidelines informed the design of a new version of Casdoc, which we present in Section 5.7.

We formulate the guidelines as lessons we learned from implementing Casdoc and evaluating it in a field study. Although they are derived from empirical evidence, more work is needed to reliably generalize each guideline to various contexts and types of documentation.

Guideline 1: Involve readers to mitigate verbosity. A format that reveals information selectively based on reader interaction mitigates the cost of including more content into a document. During the study, readers typically did not inspect the entire content of documents, indicating that extraneous content had little impact on their experience. Thus, it allows document creators to aim for a wider audience while keeping documents approachable for individual readers, instead of investing effort to optimize the balance between coverage and verbosity.

Guideline 2: Do not require user actions to reveal important information. A format that requires user interaction for a reader to reach specific information increases the probability that the reader will miss that information. During the study, readers never viewed a majority of the annotations, including annotations that were clearly indicated by a marker on the code example. Readers also viewed nested annotations, which required more complex interactions, considerably less often than top level annotations. Thus, even simple interaction patterns, such as hovering over a specific part of the document, may prevent a reader from finding some key information. This risk is especially important to mitigate for novice readers who may not correctly identify their needs and the important information in a document. To avoid this threat, the document's creator should place important information in a prominent place that does not require extensive interaction to discover.

Guideline 3: Carefully assess the impact of aesthetic decisions. Small differences in the visual presentation of a component of the format can have a high impact on the readers' behaviors. During the study, participants seemed to notice blue underline markers (for inline anchors) significantly more than gray brackets (for block anchors). Although we designed both types of markers to be similarly subtle in a document, this difference possibly caused a bias towards revealing annotations with inline anchors more often than annotations with block anchors. Document creators should carefully review multiple visual options, and choose the option that optimizes the discoverability of the document's content over stylistic preferences.

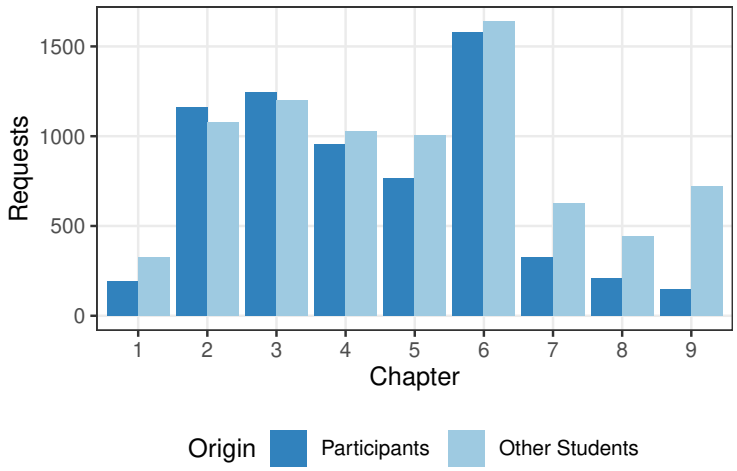


Figure 5.9: Code Example Requests by Chapter from Participants and Non-Participants During Section 2

Guideline 4: Favor an intuitive structure over navigation aids. A format with an explicit and intuitive structure with predictable hints, that readers can navigate to find specific information fragments, mitigates the need for secondary navigation aids. During the study, common navigation aids such as undo and redo buttons and breadcrumbs trails were almost never used to open annotations. Even the search bar was relatively rarely used compared to direct navigation features, despite a large fraction of each document's content being initially hidden from the readers. Thus, format designers should prioritize the improvement of a document's structure, to make it more explicit and intuitive, over the addition or improvement of more navigation aids.

Guideline 5: Consider external content to augment the content of a document. When done well, supporting the integration of external content can greatly improve the coverage and quality of a document at a minimal cost. During the study, readers benefited from over three times more annotations thanks to the integration of API reference documentation. However, this must be done carefully and with the proper attribution, as the quality, style, and authoritativeness of the imported content can vary, and the external content was not originally designed for the target document. Nevertheless, authors and format designers should consider techniques to share information across documents, given the rich documentation landscape.

5.4.3 Sampling Bias and Differences Between Sections

When recording all document requests received by the study website during the second section of the course, we observed that the majority of requests (55.1%) were not made by participants. As participation in the study was voluntary, there is the risk of a sampling bias in our results. For

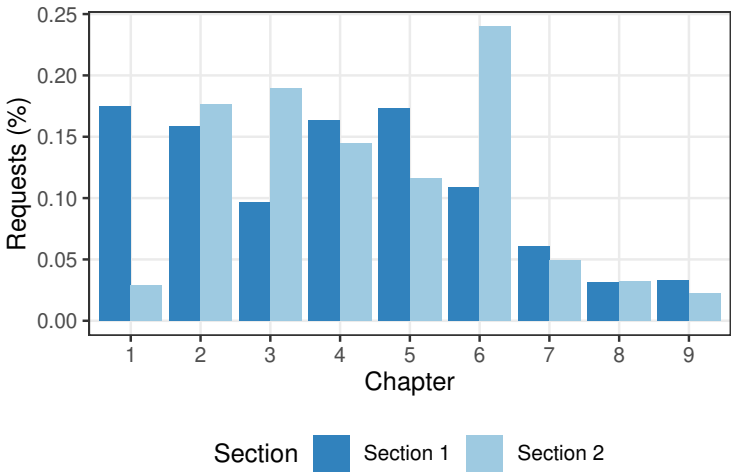


Figure 5.10: Code Example Requests by Chapter from Participants of Both Sections

example, students who are less favorable to trying new technologies may decide to only use the more familiar baseline format and forget to consent to participate in the study.[24] To assess the magnitude of the sampling bias, we investigated whether there was a notable difference between document requests from participants and other students. Figure 5.9 compares the requests for code examples for each chapter. We observe that the differences are relatively small. A Pearson's χ^2 test confirms that the differences are statistically significant ($p < 10^{-15}$), but the effect size is small (Cramer's $V = 0.18$). The same analysis, but comparing requests by week rather than by chapter, return similar results (Pearson's χ^2 test: $p < 10^{-15}$; Cramer's $V = 0.19$). Thus, although we cannot exclude the effect of a sampling bias on our results, there is no evidence of considerable differences between the sample and the target population.

We also compared the requests made by participants from the two sections of the course during which the study took place, to assess how they may differ. We expected some differences, as the two sections were independent of each other. They involved different instructors and students and used a different evaluation scheme and schedule relative to the start of the course. We also released the code examples chapter by chapter during the first section, but all at once during the second section. Figure 5.10 shows the relative number of requests per chapter for each section.[25] We can indeed observe some differences: The number of requests decreases more consistently during the first section as chapters progress—likely a symptom of the lower retention rate we observed. However, the effect size remains moderate (Pearson's χ^2 test: $p < 10^{-15}$; Cramer's $V = 0.25$), which suggests that the sample of participants did not have a considerable impact on our results.

[24]Students had to consent to the study to use the Casdoc format, as the instrumented client-side functions would generate interaction events.

[25]We use relative rather than absolute frequencies in this graph for better readability, as the total number of requests was considerably higher during the second section.

5.5 Laboratory Study Design

The field study generated precise usage metrics of the different features of Casdoc from a large population in a realistic context. The results provide evidence of the viability of Casdoc as a format for API learning resources. However, the asynchronous data collection procedure prevented us from understanding *why* readers may choose to use Casdoc. To fill this gap, we conducted a laboratory study in which we directly observed how programmers use documentation while working on implementation tasks. Specifically, we sought to answer the following research questions:

RQ 5.3 In which situations does a programmer choose to use the interactive format vs. the expanded format?

RQ 5.4 How does the interactive format support or interfere with a programmer's actions with the documentation?

We asked participants to perform several programming tasks that involved an unfamiliar API using a set of documents we provided. The documents used a combination of Casdoc and non-interactive text to present information. While participants were working on the tasks, we recorded their screen and audio to analyze how they sought information and navigated the content of the documents.

The goal of our study was to explore how well an interactive format supports programmers looking for information, rather than to demonstrate the superiority of any single format. This goal motivated an in-depth analysis of fewer participants over a statistical analysis of performance measures, such as task completion times or error rates, with a large sample from a target population. Additionally, we focused on navigation within a single document, as opposed to the synthesis of information from multiple sources.

5.5.1 Study Environment

We recruited 13 computer science students (seven undergraduates, five masters, and one PhD) from our university to participate in the study. Participants had from one to ten years of experience programming in the Java language (average: 3.5 years). Five participants identified as male, seven as female, and one preferred not to disclose their gender.

In this study, students were not a proxy for the target population. All participants were programmers who used online documentation to learn programming knowledge. Thus, we recruited students as a subgroup of the population of programmers who need to learn about unfamiliar APIs.

Table 5.7: Eight Programming Tasks Used During the Laboratory Study

#	Summary	Challenging Element
1	Return the sum of all values in a column	Getting familiar with the key elements of the JDBC API
2	Print the content of a table	Distinguishing 0 from NULL with the wasNull() method
3	Insert a new row in a table	Letting the database generate the primary key
4	Insert untrustworthy strings in a table	Using PreparedStatement to prevent SQL injection attacks
5	Create a new table and insert two rows	Creating an auto-incrementing column
6	Insert multiple rows as a single transaction	Enabling transactions and rollbacks
7	Create a connection to another database	Understanding the connection URL syntax
8	Update cells in a table based on a condition	Understanding the syntax of UPDATE statements

Each participant session took place during a videoconference with the author. Participants were compensated at the end of their session.[26]

Each session started with a pre-recorded video of the study instructions and the format of the documents. The investigator then asked the participants to locate a specific piece of information within a training document. The objective of this training activity was to help participants familiarize themselves with the documentation format. The training session involved a topic unrelated to the programming tasks.

The participants then worked on the programming tasks for 40 minutes. We asked participants to comment aloud on their information search processes, such as the information they looked for, as they worked on the tasks. The investigator would occasionally prompt participants who remained silent for too long. After 40 minutes, participants had the option to continue working on the tasks for up to an additional 20 minutes. We made it clear that this was optional and would not affect the study's compensation. Three participants chose to use this grace period to finish the task they were working on.

At the end of the session, we asked the participants to answer five questions about their experience with the documents. In total, each session lasted approximately one hour. Participants did not need to do anything for the study before or after their session.

Participants could use their preferred integrated development environment (IDE) to solve the programming tasks locally on their computer. Alternatively, we also offered the option to work on a cloud development environment that was set up by the investigator. Two participants used this option. Throughout the session, we recorded the screen and audio of the participant using the videoconferencing software.

5.5.2 Programming Tasks

We designed eight tasks that required using Java's standard JDBC API to interact with a SQLite database (see Table 5.7). We chose this domain for the tasks as it requires understanding specialized concepts, such as the relational model of databases and transactions, and supported designing tasks of increasing complexity, from simple queries to table creation operations. Another consideration was to select a domain that participants would likely have some familiarity with to be able to solve a few of the tasks, but participants should not be able to solve all tasks easily without consulting the documentation. Most participants (ten out of thirteen) had prior experience with SQL, as our institution offers an undergraduate course on databases, but not with SQLite or JDBC specifically.[27]

All tasks were presented in the same Java file, and required only Java code. After cloning a Git project, and potentially fixing configuration issues with the investigator, the participants only had to complete the body of one method per task. Peripheral issues, such as file paths, error handling, and code quality, were outside the scope of the tasks. The instructions of each task were presented as code comments in the empty method to complete.

Participants attempted the tasks in the order they were presented and had to successfully complete a task before moving on to the next one. The investigator told participants that they should not try to finish as fast as possible and that we did not expect them to finish all eight tasks during the session. Rather, the tasks served as a context to trigger information searches.

To verify whether they completed a task, participants ran the `main` method of the Java program. They could run the program as much as desired to test partial solutions, verify a solution, and debug issues. The start of the `main` method reset the database, so participants did not have to handle unintentional consequences of successive runs.

5.5.3 Documents Provided

We authored six documents for the study. Their content was based on popular tutorials, including Oracle's official "JDBC Basics" tutorial [167]. We reused the sample application scenarios from the tutorials, but adapted the content to ensure that the documents covered all the information needed to solve the tasks. As it was common in tutorials to address several variations of SQL, we included information for four popular databases: SQLite, the database used in the tasks, in addition to MySQL, PostgreSQL, and Oracle Database. The documents did not emphasize SQLite-specific content over the rest, so participants had to mentally filter out irrelevant content. However, only later tasks required vendor-specific adaptations.

[26] Participants provided informed consent before the session. This study was approved by the Research Ethics Board Office of McGill University, file number 21-06-007.

[27] We did not advertise the domain of the tasks during the recruitment phase, or used it as a selection criterion, as we did not want participants to try to learn about it prior to their session.

Table 5.8: Topic of the Six Documents Provided to the Laboratory Study Participants

Name	Topic	Task(s)
Connection	Connect to the SQL database from a Java program	7
Create-Table	Create SQL tables and specify constraints on the columns	5
Read-Values	Read the content of a database	1, 2
Write-Values	Append rows to a table of the database	3, 8
Safe-SQL	Avoid SQL injection attacks when handling user-provided values	4
Transactions	Execute several SQL statements within a transaction	6

* The study's landing page showed the topic of each document, but not the tasks they were related to.

Each document described one type of operation to perform with the JDBC API, as shown in Table 5.8. Participants were free to consult any of the documents at any time, but all the information for each task was contained in a single document. There was no indication of which document was relevant for a task.

All documents used the same format, shown in Figure 5.11. Apart from their title and a one-line description, each document started with a complete code example (highlighted in red and marked "A" in Figure 5.11). The code example contained interactive Casdoc annotations (highlighted in blue and marked "B"). Casdoc's search bar, which allowed participants to search within hidden annotations, was present in the top right corner. Below the code example, the content of all annotations was replicated in a traditional, non-interactive format (highlighted in green and marked "C"). Henceforth, we refer to the code annotations (B) as the *interactive format*, and to the non-interactive text (C) as the *expanded format*, as it expands the content of all annotations in a single (scrolling) view. We did not consider the top code example as part of either format.[28] However, we refer to the format of the entire document (A+B+C) as a *hybrid* format.

At the start of each session, the investigator told participants that both the interactive and expanded formats had exactly the same content, apart from headers and annotation titles. The only exception was for API reference documentation. Casdoc automatically generated annotations for API elements mentioned in the code (e.g., the class String). However, the expanded format did not contain that information to avoid documents that are too long, and because participants could access reference documentation from their IDE or on Oracle's website.

The hybrid format was a consequence of our study design with a single set of tasks. Given the small sample size and confounding factors such as the subjective preferences and backgrounds of participants, we opted for a within-subjects design in which each participant is exposed to both the interactive and expanded formats. In a conventional design, we would have asked all participants to work on two sets of tasks, using only one format for each set. However, this procedure would

[28] More precisely, the code example is an essential component of both formats, but it is not a *distinguishing* feature of the formats in this study.

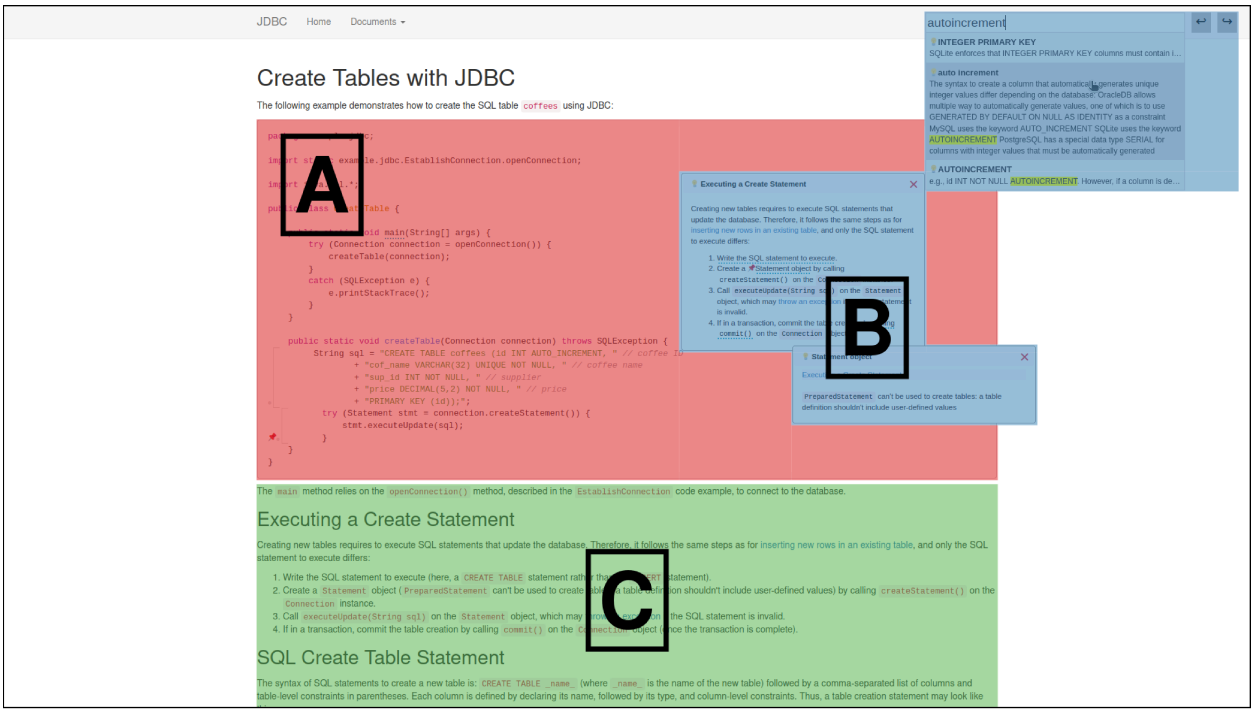


Figure 5.11: Format of the documents provided to participants. Each document uses a hybrid format composed of a code example (A), which integrates Casdoc annotations and the Casdoc search bar (B). The code example is followed by the expanded, non-interactive form of all annotations (C).

have required to create tasks that involved two distinct APIs to avoid carryover effects, introducing confounding variables due to the inevitable differences in the complexity of the tasks and quality of the documents. The conventional design would also have required long sessions, introducing the threat of participant fatigue during the second 40-minute set of tasks.[29] Thus, we employed a hybrid presentation that allowed participants to freely alternate between the interactive and the expanded formats.

5.5.4 Data Collection and Analysis

We collected data from the screen and audio recordings of participants during the programming tasks, in addition to the answers to the short questionnaire at the end of the sessions. Figure 5.12 shows the five questions in the questionnaire. The first three questions aimed at identifying aspects of the format that encourage or conflict with navigation actions (RQ 5.4). Although Q2 and Q3 required only a short answer, most participants explained their rating, providing further information

[29]We rejected the option to shorten the time to 20 minutes per set of task: When piloting the study, participants required several minutes to get familiar with the API domain before progressing on the tasks.

Q1. Did any aspect of the documentation format helped or hampered your search for information within a document?

Q2. On a scale from 1 to 6, how challenging was it to find the information you were looking for in code annotations?
[scale shown below the question: 1 = extremely easy; 6 = extremely hard]

Q3. On a scale from 1 to 6, how challenging was it to find the information you were looking for in paragraphs below the code example?
[scale shown below the question: 1 = extremely easy; 6 = extremely hard]

Q4. Did the different formats affect your strategies for finding the information you wanted? If so, how?

Q5. Did the kind of information you looked for affect which format you used?

Figure 5.12: Post-Study Questionnaire

about the relative importance of different aspects of the two formats. The last two questions aimed at eliciting types of situations that influenced the choice of format (RQ 5.3).

To answer our research questions, we started by summarizing the elements of the questionnaire responses that related to the document formats. This provided us with a list of types of situations that influenced the choice of format (RQ 5.3), and a list of navigation actions that conflicted with or were supported by the formats (RQ 5.4). We then operationalized each of these initial findings and investigated their occurrence during the programming tasks. The results consist of validated situations and navigation behavior that affect the ability of the interactive or expanded format to satisfy an information need. They contribute to the knowledge of documentation format design by providing a set of hypotheses that link the presentation format with a reader's interaction behavior.

The analysis required three phases of data annotation to transcribe the video recordings. The annotation guidelines for each phase are reproduced in Appendix C.5.

First phase: Session events The first phase took place concurrently with the analysis of the questionnaire answers. We transcribed each session as a series of events related to interactions with the documents or to the tasks. Table 5.9 shows an excerpt of this transcription. Each event has a timestamp, and belongs to one of five categories: *Task* events relate to the progress on tasks; *Window* events indicate which document the participant was looking at; *Search* events denote observable milestones of searches (e.g., starting a new search); *Doc. Component* events indicate which part of each document the participant interacted with; and finally *Intervention* events denote interactions

Table 5.9: Sample Session Events Transcribed During the First Analysis Phase

Timestamp	Task	Window	Search	Doc. Component	Intervention
...					
35:10	Start 3				
36:00		Write-Values	Start	Text	
36:16				Code	
36:37				Popover “NULL”	
36:39				Dialog “NULL”	
37:16			Found		
37:27					Question
37:33					End
...					
38:14	Coding	IDE			
39:58		Write-Values	Quick	Code	
40:02				Dialog “NULL”	
40:04		IDE			
...					

* The online appendix describes in detail the meaning of the column values.

between the participant and the investigator, such as the participant asking about the version of SQLite used. This phase provided a quantitative overview of the sessions, presented in Figure 5.13.

The bar chart excludes the time participants spent reading general instructions that applied to all tasks, e.g., instructing them to write all code in the same Java file. This exclusion is the reason why not all bars reach a total of 40 minutes. Participant 7 lost more time than others at the start because they wrongly assumed that they had to read all the documentation before starting the tasks. Participants 11 and 12, who progressed the slowest, were also the least experienced, both having only one year of prior Java experience and no prior exposure to SQL. All other participants had at least two years of Java experience and all but one of them (P8) had prior exposure to SQL.

Second phase: Search fragments For each participant, we split the study sessions into *search fragments*, which form our units of analysis. A search fragment consists of a period where the participant consults the documentation to search for some target information without making progress towards their solution. Two search fragments must be separated by time spent in the IDE working on the solution. The information sought during a search fragment may change, for example, if a participant finds some unexpected information that makes them reconsider their strategy.

In some cases, especially when trying to resolve issues, a participant would rapidly alternate between looking into the documentation and trying various ways to progress. We considered such situations as a single search fragment, as the participant stops and resumes the same search activity,

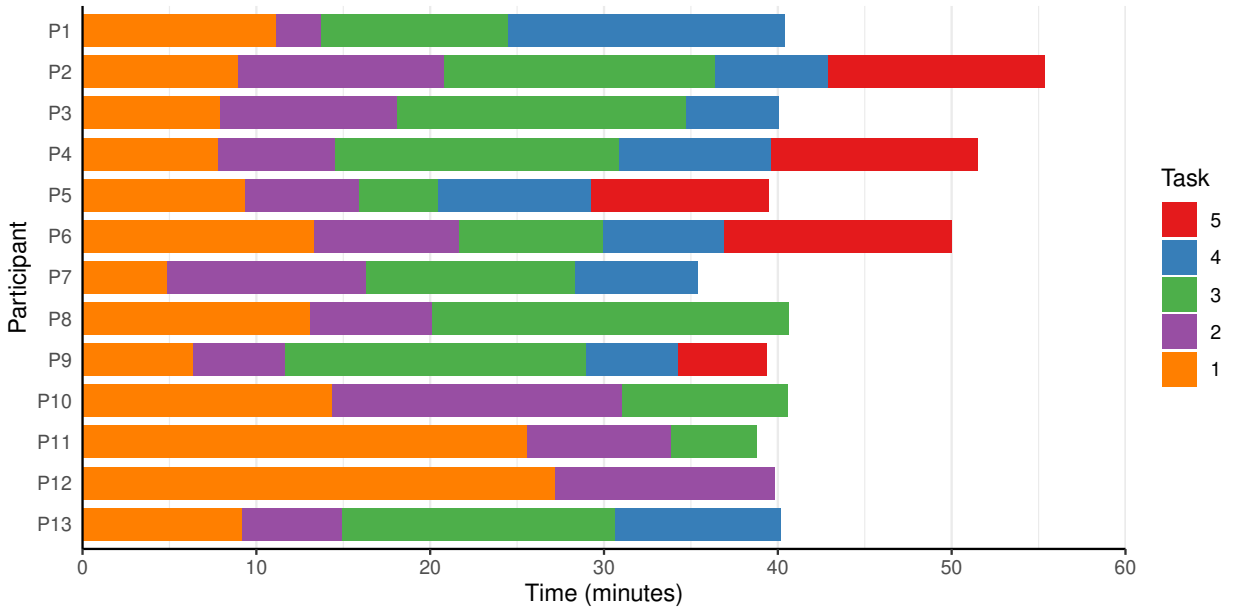


Figure 5.13: Time spent per task for each participant. The chart excludes the time spent reading general instructions about the tasks' context, which explains why not all bars reach a total of 40 minutes.

until they either find a solution or abandon trying to solve the issue (e.g., by changing their approach to the task). In both cases, we interpreted this resolution as progress in the task.

In contrast, participants searching for information only within their IDE did not constitute search fragments, as they do not involve online documentation, the focus of this study. We also excluded short interactions with the documents, such as locating known information, as they do not require the participant to *search* for information. Finally, we excluded search fragments for which the target information was unrelated to SQL or the JDBC API (e.g., information about the Java syntax), as the documents were not designed to contain such information.[30]

In total, we identified 122 search fragments, summarized in Table 5.10. For each search fragment, we extracted properties about the context in which the search took place and properties related to the navigation within the documents. The context of a search includes a broad categorization of the *intention* of the search, such as looking for an initial code to adapt as a solution or for a way to fix an issue. The navigation properties contain the list of document *components* that the participant interacted with, the *sequence* in which each component was used, and the type of information sought when using either format (*Information Type (Interactive/Expanded)*). Tables 5.11 and 5.12 show the context and navigation properties, respectively, of the search fragments of P13 as an example of the data generated during the second phase.

[30]In such cases, the investigator would provide the missing information or mention that it was irrelevant to the task, as participants could not look for other documents.

Table 5.10: Overview of the Search Fragments (Frag.) From Each Participant (Part.)

Part.	**# Frag.**	**Time/Frag.** (seconds)	**Total Time** (minutes)	**# Frag. that Use**		
				Code only	Interactive	Expanded
P1	10	50	8.3	3	3	5
P2	13	90	19.5	4	7	6
P3	9	69	10.4	2	2	7
P4	14	78	18.2	1	8	9
P5	11	89	16.3	1	10	6
P6	8	42	5.6	0	1	8
P7	7	123	14.4	1	5	2
P8	10	72	12.0	2	7	2
P9	13	71	15.4	3	9	5
P10	10	97	16.2	1	6	8
P11	5	259	21.6	0	5	4
P12	4	318	21.2	0	2	3
P13	8	61	8.1	1	6	5
Total	122	92	187.1	19	71	70

Third phase: Navigation patterns The third phase consisted of synthesizing recurring navigation patterns (presented in Section 5.6) and marking, for each search fragment, whether the fragment contains an occurrence of the pattern.

5.6 Laboratory Study Results

Figure 5.14 shows the subjective ratings of the interactive and expanded formats. Seven participants rated the information is easier to find in the interactive format than in the expanded one, two rated the expanded format above the interactive one, and one rated both equally. Three participants preferred not to rate one of the formats because they only remembered using annotations (P7 and P8) or the expanded text (P1).

Despite their preferences, all participants used both formats during their session to find information, including those who rated only one format. In answer to RQ 5.3, the analysis of the programming sessions and of the post-study questionnaire elicited **factors related to four aspects of an information search that may affect the choice of a format** (Section 5.6.1). In answer to RQ 5.4, the study also revealed **how Casdoc supported or interfered with three recurrent groups of navigation patterns, which can affect a programmer's preference for a format** (Section 5.6.2).

Table 5.11: Excerpt from the Second Analysis Phase: Context of the Search Fragments Performed by Participant P13

ID	Start	End	Task	Intention	Document(s)
115	21:16	21:48	1	Initial solution	Read-Values
116	32:55	34:03	2	Specific aspect	Read-Values, Javadoc ResultSet
117	35:59	37:16	3	Specific aspect	Write-Values, create-table
118	41:21	42:11	3	Specific aspect	Write-Values
119	42:41	43:44	3	Specific aspect	Write-Values
120	45:54	46:20	3	Specific aspect	Read-Values
121	48:57	49:49	3	Fix issue	Write-Values
122	53:26	55:27	4	Get familiar	Safe-SQL

* The online appendix describes in detail the meaning of the column values.

Table 5.12: Excerpt from the Second Analysis Phase: Navigation Properties of the Search Fragments Performed by Participant P13

ID	Components	Sequence	Information Type (Interactive)	Information Type (Expanded)
115	code	*new* → *code*		
116	code, text, popover	*known* → *code* → *text* → *code* → *popover* → *link ; javadoc* → *text*	link	content overview, element detail
117	code, text, popover, dialog	*new* → *code* → *text* → *code* → *popover* → *dialog* → *link ; new* → *code* → *text* → *code* → *back ; known* → *dialog*	continue "how to"	how to, content overview
118	dialog, popover, text	*known* → *dialog [*↔ *popover]* → *text*	element detail	confirm "content overview"
119	code, text, popover, dialog, unint. popover	*known* → *code* → *text* → *code [*↔ *popover]* → *dialog* → *link ; javadoc* → *text*	continue "how to"	content overview, how to
120	code, popover	*known* → *code* ↔ *popover*	element detail	
121	code, popover, dialog	*known* → *code* ↔ *popover* → *dialog [*↔ *popover]* ↔ *ide*	error cause	
122	code, text, unint. popover	*new* → *code* → *text* → *code [*↔ *ide] [*↔ *text]*		concept

* The online appendix describes in detail the meaning of the column values.

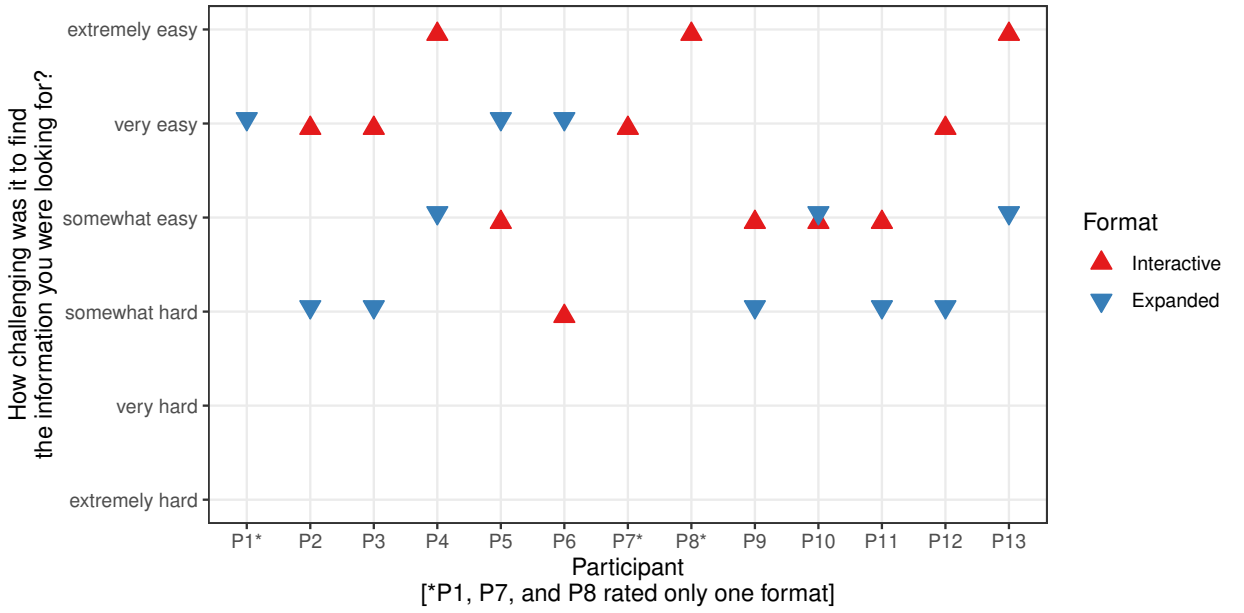


Figure 5.14: Participants' Ratings of the Interactive and Expanded Formats

Overall, the results show that the two formats can coexist in the same document and provide complementary benefits. Despite the duplication of all the content, participants appreciated having the possibility to switch between the two formats according to their evolving needs. We discuss the implications of our results in Section 5.6.3.

5.6.1 Choice of Documentation Format (RQ 5.3)

Participants related their use of the interactive and expanded formats to the need to find different types of information. For example, P5 expressed *"The general 'how to do a thing' was definitely more for the code highlights. [...] If I wanted to fix a bug or something, I looked through the paragraphs."* Similarly, P10 mentioned *"When I just want to search a keyword, I prefer just using the popup over."* We investigated this behavior at two levels. First, for each search fragment, we attributed one of four broad categories of **search intention**. The intention captured the purpose of the sought information in relation to the task. Second, we identified the **type of information** that participants looked for every time they switched to a new format during an information search fragment.[31]

Another recurring strategy from participants was to use the code and the interactive format first when looking for information, then move to the expanded format if they could not find what they were looking for easily. For example, P9 said *"I would first look at the code annotations. Then, if there's any detail that's missing, I don't really bother to go over all those levels of popovers to look*

[31]While using a format, the information type could change as the search progressed. We only considered the information type that triggered the initial usage of a format.

Table 5.13: Search Intentions and Formats Used During Search Fragments

Stage	Code only	Interactive	Expanded	Both
Get familiar	0	1	3	3
Initial solution	13	12	7	6
Specific aspect	3	16	17	25
Fix issue	3	4	5	4
Total	19	33	32	38

for it. I would just go through the paragraphs." P3 expressed more succinctly *"I always go to the code annotations first, and then try reading the paragraphs."* These responses hinted at a consistent **sequence of formats** when participants did not know in advance where the information would be located (e.g., from a prior search fragment).

Finally, multiple participants indicated that they perceived the expanded format as more **reliable and complete** than the interactive format, despite being shown that both formats contain the same content. P6 stated this bias clearly: *"I don't know if that's true, but I assumed that there's going to be more information in the paragraphs."* P10 also said *"I thought the documentation below has more thorough explanation."*

Search intention We distinguished four broad categories of search intentions: *getting familiar* with the API or domain concepts, looking for an *initial solution* in the document that can be adapted, finding information about a *specific aspect* of the task, or *fixing an issue* with an unknown cause. We used the progression on the current task and the information found as a result of the search to infer the intention when participants did not mention it explicitly.

Table 5.13 shows the number of search fragments that used either format for each intention. When looking for a specific aspect of the task or a solution to an issue, participants used each format almost as often. However, when looking for an initial solution to the task, participants used interactive annotations more often than the expanded text to support their initial understanding of the code example. In contrast, although it did not happen too often, reading the expanded format was preferred when a participant wanted to get familiar with the domain of a task without a specific query in mind.

Type of Information Participants often looked for the same recurring information as they proceeded through the tasks. We identified six categories of information needs specific to our experiment based on how participants expressed what they looked for: instructions on *how to* perform an operation, *error causes* and potential solutions, API *element details*, clarifications about the *code example implementation*, details about the *SQL syntax*, and *conceptual* information about

Table 5.14: Usage of Each Format to Look for Information of Each Type

Information Type	**Format**		
	Interactive	Expanded	Total
how to	10	18	28
error cause	2	7	9
element detail	27	6	33
example implementation	13	2	15
SQL syntax	13	17	30
concept	3	5	8

the task or API domain. When a participant used both formats to search for the same information, only the first format is considered, as it was the participant's initial instinct.

Table 5.14 shows the usage frequency of both formats for each information type. Participants predominantly used the interactive format for details about an API element and clarifications of the main code example's implementation. In contrast, they sought "how to" instructions and error causes more often in the expanded format. Information about SQL syntax and concepts were more balanced between both formats, with a preference for the expanded format.

Sequence of formats Many participants mentioned looking for information first in the interactive annotations, and then in the expanded text if it took too long to find the information. However, when identifying instances where a participant changed format when looking for the same information, we observed that they went from the interactive format to the expanded format 15 times, and the other way around 14 times. This suggests that regardless of the initial format, when participants could not find some information, changing format was a viable strategy to continue the search.

However, a closer look at the instances revealed that participants mostly (11 out of 14 times) relied on the search feature of Casdoc after failing to find information in the expanded text. This suggests a more nuanced sequence of formats: first interacting with the annotations from the code example, then looking into the expanded text, and finally using a text search feature to locate hard-to-find information.

Perceived reliability and completeness Despite an explicit mention that both formats had the exact same content at the start of each session, participants showed a bias to trust more the expanded format. We did not expect this bias, and thus did not measure this subjective aspect of the format, e.g., with questions in the post-study questionnaire.

We nevertheless gathered evidence of this bias by identifying instances where participants used one format to confirm information already found. Five search fragments, from four distinct

participants, showed such behavior, all using the expanded format. In these instances, participants expressed the desire to scan the text in case they missed relevant information.

5.6.2 Support of Navigation Actions (RQ 5.4)

Participants referred to three recurrent groups of navigation actions during the study. One of the most common actions was to first **look for code examples** when reading a new document. For example, P1 stated *"I prefer to look directly at some code which is simple but really good. [...] Even if the code is at the end of a document, I always try to look at it."* This sentiment was echoed by P2, among others: *"I always default to code first."* Even after reading the main code example, some participants looked through the document for further examples, such as P12, who indicated while scanning the expanded text *"I am looking for examples, because that's how I learn best."*

Another common action was to **scan a document's content** without a single specific information need. Participants performed this action to locate where a given topic was discussed, to assess the extent of the coverage of a topic, or to get an overview of the set of topics discussed in a document. For example, P6 discussed having prior strategies for reading expanded text efficiently, and said *"I guess my strategy is always to look for the keyword and then, once I see the keyword, I will read stuff around it."* P9 and P12 also mentioned their appreciation for features that help scan documents by stating, respectively, *"One thing that I found really helpful was the subtitles."* and *"I'm also someone who really enjoys indentation in things, to make it easier to scan through."*

Finally, participants pointed out limitations related to popovers appearing unintentionally when they **read with the mouse**. P7 mentioned this issue directly: *"There were so many pop-ups. I've a tendency to move my mouse while I read the code. [...] That kind of interrupts the reading flow."* Similarly, popovers interfered with participants trying to copy and paste code, as P6 said: *"When I try to do copy-pasting, the pop-up window [...] actually interferes with the actual copy-pasting of the code."*

Looking for code examples We measured the reliance on code examples by identifying when, if at all, participants used the main code example during search fragments. In 19 search fragments (16%), participants used only the code example. In 57 fragments, (47%), the code was the first element participants looked at, and in 16 fragments (13%), they looked at other components of the documents first. In the remaining 30 fragments (25%), participants did not use the code example. This prominent use of the code example is consistent with the participants' comments. Although it was not a distinguishing feature of the two formats in this study, the focus on a complete code example was one of the key design properties of Casdoc (see Section 5.2). Our results confirm the validity of this design aspect for documentation.

Scanning a document's content Casdoc does not support well scanning the content of a document, as it cannot provide an overview of all annotations at once. We measured how often participants did scan the expanded format to assess the impact of this limitation of the interactive format. We observed this action in 30 search fragments, performed by all but one participant. This represents 25% of all search fragments, or 43% of the fragments where the expanded format was used. The popularity of this action suggests that it should be a consideration for documentation designers, especially for interactive formats that do not show all of their content at once.

Reading with the mouse Although many participants appreciated the swift interaction mechanism to reveal and hide popovers, some noted their distracting nature when performing other mouse actions. To measure this negative impact, we measured how often popovers opened by unrelated mouse movements distracted participants.[32] A popover was *distracting* when it interfered with a participant trying to select some code to copy, when it hid the code that the participant was trying to read, or when it triggered an abrupt mouse motion to leave the anchor. We found instances of distracting popovers in 36 search fragments (30%), affecting all but one participant. This result highlights a trade-off when designing the interaction mechanism of a format: simplifying the actions to access content can generate accidental events.

5.6.3 Discussion

Our results generally show that the interactive format allowed participant to directly get concise information to answer specific queries. However, queries that required more elaborate answers were better answered in the expanded format. The expanded format also made it easier for participants to scan information related to their original query. P10 gave the syntax of SQL conditions as an example: *"Sometimes I want to see other content related to 'WHERE' as well, [...] to see the overall picture."* Reading this related content typically required participants to spend more time than with the interactive format, but it could help them feel more confident that they correctly understood the information they found.

The personality and prior navigation behavior of a reader may also affect their preference for a format. How much a reader uses their mouse while reading may affect their opinion on whether the usefulness of pointer-triggered interactions to reveal contextual information outweigh their distracting potential. Some readers may also be more enthusiastic about developing new navigation techniques for the interactive format, whereas others may be reticent to abandon the more familiar expanded format. Thus, the design of documentation should not only accommodate differences in the search contexts, but also in personal preferences.

[32]These unintentional popovers did not count as a usage of the interactive format in the other metrics, e.g., when comparing how often participants used each format.

An overarching theme among the results is that the combination of the interactive and expanded formats accommodated a variety of navigation techniques. Each format mitigated some limitations of the other, which allowed participants to use the most appropriate format depending on their needs. Participants noted this synergy: *"If I wanted very quick information, I would just look at the highlights that were directly in the code. [...] If I got stuck on something, or I needed a bit more information, then I went into the paragraphs to look for it. [...] The two levels were quite nice, with respect to how much information I needed"* (P5). Even the redundancy of the content, which is typically regarded as a negative aspect of documentation, had some advantages: *"It was kind of reassuring to know that there are different strategies for finding what I wanted. [...] It [duplicated content] made me more comfortable looking anywhere"* (P12). We did not expect such positive feedback on the dual nature of the hybrid format, which we created to avoid long programming sessions. Future work could fully integrate the two formats, e.g., by allowing readers to jump between annotations and their expanded representation, to further improve documentation design.

5.6.4 Limitations

Studying a novel documentation format required to impose some constraints that limited the realism of the experimental conditions. First, forcing participants to use only the provided documents meant that they could not use familiar resources such as their favorite search engine or Stack Overflow. Some participants expressed that this constraint affected their typical programming habits, particularly when trying to find the cause of an error in their code. Many participants (P4, P6, P7, and P13) also indicated that redirecting the keyboard shortcut Ctrl+F to Casdoc's search bar, instead of the browser native search tool, negatively affected their experience.[33] These constraints had an impact on the navigation behavior of participants, and therefore the study findings. They were necessary to encourage participants to try the interactive format during the sessions, but further investigation is needed to carefully assess their impact.

The study also required participants to get familiar with a new documentation format. Participants were used to searching within the expanded format, and had developed techniques to do so more efficiently. P6 expressed this bias directly: *"Reading the paragraphs is more intuitive, because I'm already used to reading that kind of documentation."* In contrast, participants had to learn about Casdoc's features and develop an intuition about how and when to use them. Participants reacted differently to this change. Some were more enthusiastic and tried more advanced features of Casdoc, whereas others preferred to rely mostly on their existing strategies. It is possible that programmers who have used Casdoc for more than the study's 40 minutes would elaborate new navigation approaches that the findings do not capture.

[33]The browser search tool remained available without the keyboard shortcut, but participants had to open it from the browser menu.

A threat to the validity of our observations is that the investigators designed both the tasks and the documents. We ensured that both formats in each document contained exactly the same content, which was based on existing online tutorials. It is possible, however, that other design aspects of the documentation, such as the order of paragraphs and sections in the static text, would affect their navigability. Regarding the tasks, we designed them to represent common database operations, while also involving more challenging parts of the JDBC API so that participants would need to consult the documentation. To mitigate the potential of an investigator bias on the results, the analysis focused on qualitative similarities and differences between the formats, rather than on quantitative measures of properties (e.g., time) used as a proxy for document quality.

The selection of the participant sample also limits the generalizability of the results. A smaller sample size permits a more detailed analysis of the programming sessions within a practical time budget. This approach was consistent with the exploratory nature of the study, and is a first step to generate hypotheses that can be tested in future studies using statistical methods over a large sample.

5.7 Improving the Casdoc Format

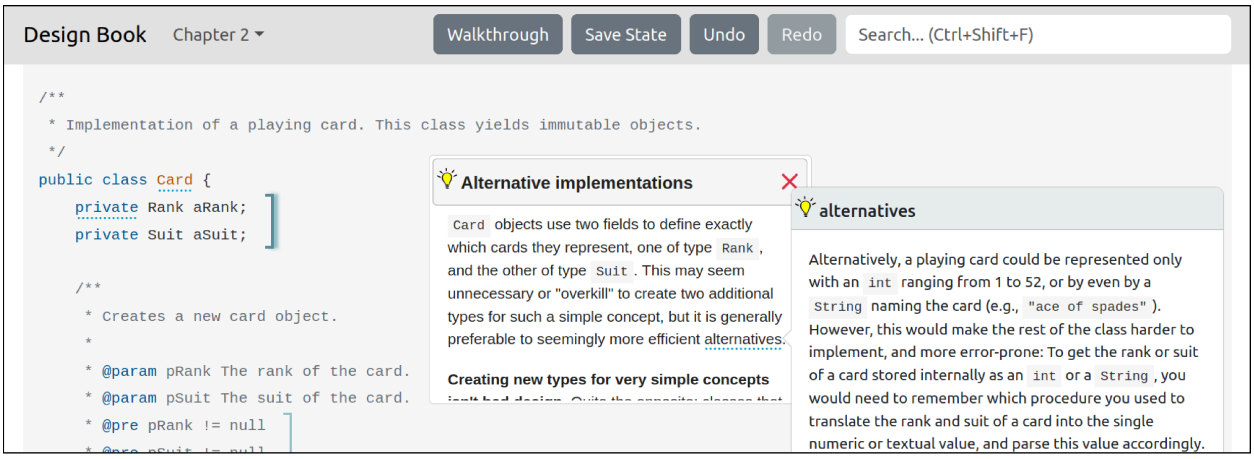
Following the field study, we released the annotated code examples, without the JavaScript data collection functions, on a permanent website for students enrolled in future sections of the undergraduate course.[34] We leveraged the study findings to implement a new version of Casdoc that address some of its original shortcomings. Figure 5.15 shows one of the documents in the revised format.

According to Guideline 3, we revised the visual elements of Casdoc, taking into consideration their impact on readers. Instead of its original arbitrary color scheme, Casdoc now uses a color scheme for code blocks that is similar to the one from Stack Overflow, a popular forum among programmers (see Figure 5.15a). This color scheme should be more familiar to programmers, mitigating the adoption cost of a new format. To make block anchors stand out more, their markers appear at the right of the code and are blue instead of light gray. This increases the contrast of block markers with the background and reduces the distance between the marker and indented code. Finally, anchors of pinned annotations no longer show a "pin" icon next to its marker, as it was disrupting the layout of the code.[35] Instead, inline anchors are shown in bold and italics font and block anchors have their marker in darker blue and with a drop shadow to indicate that the associated annotation is pinned.

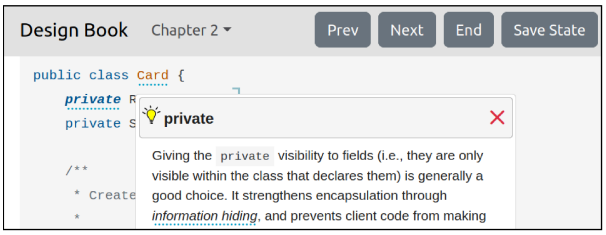
We also added three new features to Casdoc. First, authors can now identify a sequence of important annotations when creating a Casdoc document. In addition to their normal behavior,

[34] https://www.cs.mcgill.ca/~martin/designbook/

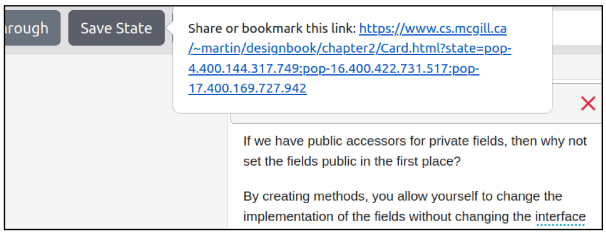
[35] The icon was also interfering with copy-and-paste behavior, as it would be included in the copied code.



(a) General view of the document with one pinned and one floating annotation



(b) Navigation using the Walkthrough feature



(c) Creation of a link to save and share pinned annotations

Figure 5.15: Example of a Document Using the Revised Version of Casdoc

readers can reveal the annotations in such a sequence by clicking on a "Walkthrough" button (see the top of Figure 5.15a). The walkthrough reveals one annotation at a time, controlled by a pair of "Prev" and "Next" buttons (see Figure 5.15b). As they move through the sequence, readers can interact with any other annotations. The walkthrough feature was informed by Guideline 2: Although walkthroughs still require some user actions, they allow readers to access important annotations in a standard way across documents, thus reducing the cognitive effort associated to these actions. Consistently with Guideline 4, this feature also provides an additional linear structure to help readers who do not know what information to look for navigate the non-linear graph of annotations.

Second, after pinning annotations and laying them out on the page, readers can save the state of the document by clicking on a "Save State" button. Casdoc will generate a custom URL that will reopen the document with all pinned annotations already visible and in the same position (see Figure 5.15c). Readers can bookmark this link to keep track of annotations they find relevant. The author of a document can also use this feature to manipulate the annotations initially visible to

readers. Thus, it can help make important annotations more prominent without requiring any user action (Guideline 2).

Finally, authors can now store reusable annotations in a database, and insert them in multiple Casdoc documents. In addition to mitigating the need to copy the content of recurrent annotations (e.g., an annotation that describes a common theoretical concept), the database provides a flexible interface to integrate external content into Casdoc documents (Guideline 5). The database stores the exact content of each annotation in an HTML file, associated with another file that contains properties of the annotation, such as its title. Thus, the database can be populated with external content using simple scripts, but the author of a document remains in control of which annotations are added to the document.

Chapter 6

Discussion

Our investigation of software documentation design revealed exciting opportunities to improve current practices. Overall, this thesis demonstrates that the design of API documentation goes well beyond writing technical information. The identification of related concepts, the selection of which concepts to include or not in a document, and the organization of information fragments into an effective presentation structure all play an important role in the quality of a document. Yet, this thesis only focused on user-facing characteristics of existing documents. The documentation authoring process was outside the scope of our research. Although we considered the authoring process, e.g., when designing the Casdoc transformation tool to ease the creation of Casdoc document, future empirical studies should directly target the characteristics, motivations, and behavior of documentation creators. Such studies would complement this thesis by revealing the impact, and viability, of novel documentation practices.

In the following sections, we discuss recurring themes in documentation design, with implications for researchers and practitioners. We then present ideas for future work. We end this chapter with a reflection of the impact language models and conversational agents (e.g., ChatGPT) on documentation practices.

6.1 Recurring Themes in Documentation Design

Our investigation of documentation highlighted several design aspects that researchers and practitioners should consider when assessing or improving the quality of documents.

6.1.1 Adaptability of Documentation Guidelines to Various Contexts

Our comparison of tutorials and the design and evaluation of Casdoc elicited several guidelines and design dimensions to consider when designing documentation (see Table 4.1 and Sections 5.2 and 5.4.2). In addition to our findings, researchers have collected empirical evidence about the impact of other design aspects, such as the use of UML diagrams [3, 148] or showing API usage statistics [204]. Nevertheless, the laboratory study reaffirms that those design considerations are not universal. The same feature of a document's format can be both useful and detrimental, depending on the reader's personality and on the information search context. The subjectivity and contextual aspects of documentation quality highlights the importance of considering various groups of developers and various development tasks when designing documentation.

To refine design guidelines for different contexts, prior work on different types of user personalities, such as GenderMag [34], can provide theoretical tools to correlate documentation usage patterns to different groups of users. Future work can develop and validate psychometric tests to better capture the subjective aspects of various software engineering activities, including documentation [81]. Categorizations of the types of documentation (e.g., API reference, tutorials, how-to guides, and explanations [180]) can provide a framework to support the analysis of the contextual aspect of design guidelines.

Comparing the documentation approach of different communities of programmers can also help reflect on the applicability of design guidelines for various scenarios. For example, R is a language used primarily for statistical analysis and visualizations [211]. It is designed to be accessible to scientists and statisticians who do not otherwise have programming experience. Therefore, studies of the API reference documentation of R packages (e.g., [223]) can reveal differences from the reference documentation of typical development languages, such as Java. Understanding these differences is important not only to adapt guidelines for different technologies, but also to design truly adaptable documentation for different users of the same technology [190].

Despite the potential for improvement, the risk of decreasing document quality with ineffective documentation designs can incite writers to reuse conservative designs. However, the results of our studies with Casdoc showed that readers are willing to try new types of documentation. Writers should leverage this enthusiasm to explore new approaches to documentation. A novel format that helps readers locate the information they need effectively can constitute a competitive advantage over other documentation resources in the current documentation landscape. Advanced formats that effectively adapt to multiple information search contexts can be especially useful to attract large audiences. With Casdoc, we contribute to this exploration of novel formats, but many design aspects remain unexplored. For example, a document format could offer readers the ability to modify the length of code examples or the proportion of images and tables in a document to match their

preference [11, 12]. Researchers should leverage the enthusiasm for innovative formats to further study documentation design aspects, to eventually synthesize different aspects into a comprehensive theory.

6.1.2 Selection of Topics to Include in Documentation

Our comparison of three Android tutorials showed a surprising variety of topics covered in each tutorial. Prior studies that elicited questions from developers about unfamiliar APIs (e.g., [62, 197]) offer valuable insights to improve the description of *individual* topics. For example, they reveal what information developers need about selected API elements (e.g., *"Which keywords best describe a functionality provided by the API?"* [62]). However, they do not offer concrete guidelines to prioritize some topics over others within a domain. For example, when creating a tutorial, writers must decide which API elements to describe, which ones to mention, and which ones to ignore or relegate to a "Further reading" section.

To help with the selection of content for some documentation, writers can rely on information needs expressed on public developer forums (e.g., Stack Overflow, as we used in our comparison of tutorials), usage information from open source projects (e.g., hosted on GitHub), or other sources of data (e.g., telemetry). These sources of information can help identify more challenging or popular aspects of the technology to document, but they do not capture the inherent dependencies between different components of the technology. Therefore, the topics identified by these sources can be disconnected, leading to disjointed documents. Our investigation of conceptual dependencies can support an analysis of the relevant topics to cover in documentation, which takes into consideration the structure of the technology.

With Scode, writers can identify the components of a technology that relate to the same topic. A writer who wishes to create a tutorial about a popular library could start by identifying key concepts commonly discussed in Stack Overflow. The writer could then use Scode to identify the conceptual dependencies of each component of the library. Based on this information, the writer would then be able to systematically select cohesive groups of API elements to cover that relate to topics often discussed by developers.

A similar approach can also be applied to other knowledge graphs. Identifying communities among API elements, based on mentions in the reference documentation or on advanced API knowledge graphs (e.g., [121]), could reveal cohesive sets of elements to describe in relation to a task. Paired with data about the popularity or complexity of different components, this information can help writers create cohesive documents that comprehensively cover the most useful topics.

Document formats that help readers navigate efficiently to the information they seek can mitigate this problem by allowing writers to select more topics to cover without creating overwhelming documents. For example, Casdoc documents can include a lot of content in nested annotations, but

this content will only be showed to readers who seek it. However, even an optimal format would not completely solve the problem of topic selection, as writers would still have to prioritize topics to emphasize within a document.

We note that practical constraints can also affect the content of a tutorial. The motivation for creating documentation can impact the content that writers choose to cover. For example, writers who create documentation as a hobby or as a personal learning exercise may select topics based on their personal interests rather than on the needs of their expected audience [13, 170]. Other factors, such as a company's interest in advertising their products, a limited time or effort budget, or known characteristics of a small audience (e.g., when creating internal documentation for colleagues within an organization) can also affect the selection of content. In our investigation of Android tutorials, we attempted to minimize the impact of these considerations when selecting the tutorials to study. However, future work should study how practical constraints affect the decisions taken by practitioners.

6.1.3 Explicit Representations of Knowledge about Software Systems and Development

Measuring or manipulating the information encapsulated in documents is challenging due to the intangible nature of human knowledge. Yet, many documentation tasks, such as the selection of topics to cover in a document, require to explicitly and systematically reflect on knowledge. Knowledge bases can support these tasks by providing explicit representations of knowledge.[1] They can be valuable sources of information to support approaches for generating dynamic API class documentation [121] and for comparing similar API elements [122].

Although many knowledge bases are available, finding one suitable for a specific application requires considering many trade-offs. We used two knowledge bases in our studies: one (Wikipedia) in the development of Scode to identify conceptual dependencies, and another one (API reference documentation) in the development of Casdoc to generate Javadoc annotations. In this section, we discuss some of the trade-offs we considered when selecting appropriate knowledge bases for Scode and Casdoc. Table 6.1 complements this discussion with some examples of knowledge bases with different properties, and a summary of the benefits and limitations of each.

[1] In the context of this discussion, we consider as a knowledge base any structured collection of information organized into meaningful *entries* and that can be queried in a systematic way. This definition includes knowledge graphs, ontologies, and glossaries. For example, we consider the complete reference documentation of an API as a knowledge base, as it is composed of separate entries, i.e., the documentation of each API element.

Table 6.1: Types of Knowledge Base to Support Software Documentation Tasks

Properties	**Example**	**Benefits**	**Limitations**
Expert-curated, software-specific	SWEBOK [28], ISO 24765 [101]	Authoritative and traceable knowledge.	Require a large effort investment; less frequent updates and smaller coverage.
Generated, software-specific	HDSKG [245]	High coverage, especially for technical information.	Varying reliability of information and consistency of abstract concepts.
Crowdsourced, software-specific	Stack Overflow	High coverage; creation cost distributed among community members.	Quality of information correlates to the popularity of a topic; consistency depends on moderators rather than strict structures.
Crowdsourced, general-domain	Wikipedia	High coverage; large community can attract experts that contribute high-quality content.	Lack of specialized information; task-specific considerations may not align with general-domain guidelines.
Derived	SOTorrent [19], Wikidata [226]	Similar coverage and quality than the original knowledge base, with more consistent structures.	Transformation process introduces an additional source of errors; excludes information about natural community interactions.
Combined	API Caveats Knowledge Graph [116]	Mitigate content limitations of multiple knowledge bases.	Heterogeneous entries can create challenges when processing the information.

Origin of Knowledge

Gathering and organizing the content of a knowledge base requires a lot of effort. Different strategies are possible to alleviate or distribute this effort.

Expert-curated knowledge bases are manually created by a group of domain experts, ensuring the high quality of their content. They include the Software Engineering Body of Knowledge (SWEBOK) [28] or an ISO/IEC/IEEE standard [101]. The quality of their content comes at a cost, as the effort required to create and evolve these knowledge bases prevents frequent updates. In particular, curated project-agnostic knowledge bases typically focus on stable software development concepts, rather than covering the most recent technologies. API reference documentation can also be considered a curated, project-specific knowledge base, as its content is created and maintained by the project's developers. In this case, the content focuses on technical information about a single API.

Tool-generated knowledge bases, such as HDSKG [245] or PengKG [121], can cover a wide range of concepts and entities, including detailed information about software APIs (e.g., list of public types and methods). However, the accuracy and coverage of a generated knowledge base depend on the quality of the corpus from which the knowledge is extracted. Thus, although the generation of the knowledge base requires little to no effort, this effort is instead spent on gathering and curating the input corpus, in addition to the initial effort to develop the generation approach. Generation approaches also have an inherent trade-off between the range and consistency of the extracted information. This trade-off was apparent in our comparison between Scode and PengKG: We observed that PengKG included a large variety of concepts, but few were consistently applied to multiple entries.

Crowdsourced knowledge bases, such as Stack Overflow tags, cover a wider range of topics than curated knowledge bases, but without the unpredictability of automated knowledge base construction methods. However, the quality and coverage of crowdsourced knowledge bases depend on the size of the community of contributors. Thus, successful crowdsourced knowledge bases typically have loose structures maintained by moderators to lower the barriers to entry for newcomers. For example, on Stack Overflow, there is no systematic protocol to detect redundant tags. This task instead relies on contributors individually identifying and voting on tag synonyms on a case-by-case basis. As a result, the quality of information is not uniform across the knowledge base, as popular entries receive more attention than specialized ones.

Domain of Knowledge

The breadth of topics covered in knowledge bases varies considerably. **Domain-specific** knowledge bases contain specialized information about a narrow domain. For example, the Common Vulnerabilities and Exposures (CVE) dataset contains structured information about cybersecurity

vulnerabilities [210]. In contrast, **general-domain** knowledge bases cover common topics related to many domains. For example, WordNet is a lexical knowledge base that contains most of the common English words [179]. The distinction between domain-specific and general-domain knowledge bases is relative to the domain of the target application. For example, Stack Overflow tags can be considered a domain-specific dataset to study the landscape of software technologies [163], but a general-domain dataset to study the content of Android tutorials (Section 4.3).

As domain-specific knowledge bases focus on fewer topics, they can contain more detailed information about them. For example, they can divide a concept into many fine-grained entries in the knowledge base (e.g., include one entry for each component of a software library, rather than a single entry for the entire library). They can also include specialized structures to relate entries that would not apply outside the given domain (e.g., software products affected by CVE entries).

As general-domain knowledge bases capture many topics, they are typically more popular. This popularity can positively affect the quality and quantity of information contained in the knowledge base. In particular, popular crowdsourced knowledge bases can benefit from larger communities of contributors, which can also attract many experts in a variety of topics. Thus, in some cases, the quality of information in a domain-specific subset of a general-domain knowledge base can rival the information in a knowledge base specific to the same domain. For example, when studying information needs about the Android framework, it may be preferable to use data from the general-domain forum Stack Overflow instead of its Android-specific equivalent, Android Enthusiasts Stack Exchange [24]: As of March 2023, Stack Overflow contains over 1.4 million questions tagged with `android`, whereas Android Enthusiasts contains a total of 59k questions.

Among general-domain knowledge bases, **Wikipedia** is a mature, extensive, and active option. Due to its popularity, researchers and practitioners have designed many tools to interact with Wikipedia, including the six wikifiers we evaluated. These tools are a practical consideration to reduce the development effort of knowledge-oriented documentation tools. Wikipedia contains an impressive amount of information, with over six million articles (as of March 2023). It uses many structures to organize this information, such as categories and information boxes. However, due to the size of Wikipedia, these structures inevitably contain inconsistencies. To illustrate this limitation, we consider Wikipedia's categories. A category groups articles about a similar topic. Categories can themselves be grouped into parent categories, creating a hierarchical structure. However, topics captured by categories can considerably change when navigating the hierarchy. For example, Java enterprise platform is a descendant of Economy through three intermediate categories, and of Paleoanthropology through four additional intermediate categories.[2] The

[2]Java enterprise platform is a subcategory of Business software, which is a subcategory of Business computing, which is a subcategory of Business, which is a subcategory of Economy, which is a subcategory of Society, which is a subcategory of Humans, which is a subcategory of Hominina, which is a subcategory of Hominini, which is a subcategory of Paleoanthropology. This sequence of categories is not a rare oddity, but rather a representative example of topics changing when retrieving parent categories.

hierarchical structure of categories is also imperfect, as it contains cycles. Thus, although categories provide useful information, this information can be unreliable on a large scale.

Knowledge Base Transformations

A key attribute of knowledge bases is to provide the ability to manipulate knowledge using systematic procedures. As a consequence, knowledge bases can be created by transforming other knowledge bases.

Some knowledge bases are **derived** from other knowledge bases. For example, DBpedia [114] and Wikidata [226] are derived from Wikipedia. They provide more rigid structures by curating or automatically extracting structured information. Thus, they address one of the main limitations of Wikipedia, but they lose information that has not yet been formalized into a rigid structure. For example, the context in which internal Wikipedia links appear (e.g., its surrounding words or the section within the article) can provide information about the nature of the relationship between the two linked articles. This information is lost in derived knowledge bases. Therefore, derived knowledge bases can provide more structured information with similar coverage and accuracy than the original knowledge base, but they exclude the latent information that arise from natural interactions between community members.

It is also possible to **combine** multiple knowledge bases into a single one by extracting meaningful associations between entries of each original knowledge base. For example, Li et al. created a knowledge graph of API caveats (i.e., directives) by linking the graph of API elements of a framework to a dataset of caveat sentences extracted from tutorials [116]. Similarly, Scode can help combine a graph of API elements to Wikipedia entries. Combined knowledge bases can mitigate limitations in the content of its sources. However, they introduce the challenge of data heterogeneity. Documentation approaches that use combined knowledge bases must ensure to properly handle the different types of entries, which can increase the complexity of developing accurate processing techniques. For example, WiBiTaxonomy is an approach that uses Wikipedia to identify pairs of terms in a semantic generic–specific relation (e.g., Programming language and C++) [73]. As it relies on a heterogeneous graph, where nodes can be either a Wikipedia article or a category, it requires two distinct parameters to control the number of articles and categories considered, respectively.

Creating or Selecting an Appropriate Knowledge Base

These considerations for selecting an appropriate knowledge base for a specific application demonstrate the complexity of representing knowledge explicitly. Yet, a carefully selected or constructed knowledge base constitutes an invaluable asset to developers and researchers.

When designing documentation, writers can consider adding machine-readable structures to documents, so that the documentation can also serve as a knowledge base. For example, a tutorial writer could mark usage constraints, common error symptoms, or supported tasks described in the tutorial with a special HTML syntax (e.g., using a recognizable HTML class). With this approach, the tutorial writer could increase the value of its documentation with only a small additional effort.

Researchers can also further study the opportunities provided by different knowledge bases. For example, Casdoc can be expanded by generating annotations from different knowledge bases (e.g., Wikipedia, Stack Overflow, or CVE). Each additional knowledge base would require solving challenges to ensure that only relevant annotations are added to Casdoc documents. Solutions to these challenges would constitute useful contributions to both the software documentation and traceability recovery fields.

6.2 Future Work

The themes discussed in the previous section highlighted areas of software documentation research that can be further explored. Future work on the contextual aspects of documentation design can include studies of various audiences (e.g., students, professional developers, non-developer programmers), various technological contexts (e.g., declarative, object-oriented, or functional languages), and various domains (e.g., utilities libraries, security libraries, end-user software bound by legal requirements). Eventually, secondary studies can create theoretical models to identify effective sets of guidelines for new development contexts.

Researchers can further study topic selection guidelines by conducting observation studies in which they analyze the correlation between topics that readers look at in existing documents. Alternatively, they can conduct experiments in which they modify a sample of base documents to evaluate the impact of excluding different topics on quality scores given by participants.

Future work on explicit representations of knowledge can develop techniques to generate knowledge bases with an emphasis on the consistency of abstract concepts or relations.

Beyond these research ideas, we discuss three additional opportunities for future research. First, we describe how Casdoc can serve as an instrument to gather precise information about documentation usage in realistic environments. Second, we discuss the need of more studies on the effort required to create documentation. Finally, we discuss possible improvements that can be made to the Casdoc format.

6.2.1 Casdoc as an Extensible Instrument for Studying Documentation

The development of Casdoc constitutes a concrete contribution to practitioners. We integrated Casdoc documents as part of the learning material for an undergraduate course, and plan to continue improving the format.

Beyond this contribution, an incidental benefit of the design of Casdoc is that it allowed us to collect detailed data about the information that participants looked at without disrupting their documentation usage behavior (see Section 5.3). The objective of our field study was to evaluate Casdoc itself, so we did not analyze the content of the annotations that participants looked at. However, future work can use a similar data collection methodology to study information needs.

As readers have to consciously reveal information by interacting with the document, and as information is split into concise fragments, monitoring readers' interaction can generate fine-grained evidence of specific information needs. The design of Casdoc also supports the addition of different types of annotations. The revised version of Casdoc presented in Section 5.7 includes two types of annotations (i.e., authored by the investigators and extracted from API reference documentation). Supporting more types would be a straightforward modification consistent with the design principles of Casdoc. Having multiple types, each with a unique visual appearance, would allow to collect even more precise data about information sought by readers.

Using Casdoc as a study instrument has several benefits over alternative data collection methods such as surveys and observation studies. It avoids the need for data collection forms that disrupt information searching activities, rely on self-reported data, and may require participants to recall actions or to reflect on subconscious behavior. As opposed to observation studies, Casdoc can be deployed in realistic environments. The automated and asynchronous data collection process also allows to study many participants over long periods of time. Casdoc also does not require the use of specialized tools such as eye tracking equipment and software to collect precise data.

Casdoc also has limitations. Although our studies demonstrated the viability of the format, using it in a study of information needs introduces a confounding factor, as it requires readers to adapt their strategies for finding information. Casdoc also requires investigators to create new documents and participants to use them, as opposed to using existing resources that participants are already familiar with. These limitations impact the realism of the study environment. In return, investigators have more control over the documentation, which can allow them to mitigate other confounding variables (e.g., the quality of the documents). Finally, Casdoc can only collect quantitative data, limiting its usefulness for exploratory research.

These benefits and limitations of Casdoc make it a valuable instrument for large-scale descriptive or confirmatory studies. For example, it can help confirm information needs elicited in observation studies [62, 197] or measure their relative frequency.

In addition to their research potential, instrumented Casdoc documents can also provide valuable feedback to documentation writers. A challenging aspect of creating online documentation is the lack of interaction with an actual audience [118]. Casdoc can address this problem by capturing fine-grained usage data from any reader of a document. This information can help writers fix the parts of a document that need improvement, or identify effective design decisions to replicate in other documents. To realize this scenario, researchers or practitioners can design methodologies and tools to elicit actionable recommendations from the analysis of Casdoc's interaction data.

6.2.2 Challenges of Creating Documentation

Our comparison of Android tutorials and our studies of Casdoc focused on the readers' perception of documentation. However, the perspective of writers is also important to make a documentation format successful. Creating documentation is already an effort-intensive activity, where writers must balance multiple objectives. Thus, formats that require impractical amounts of effort from writers will not be adopted, or only by the most motivated minority, even if they provide clear benefits to readers.

As an example, researchers have argued for the publication of interactive scientific articles to encourage readers to critically examine the data and their interpretation presented in articles [61]. However, creating interactive articles without efficient tools require considerably more effort than traditional articles, which can explain why this practice is not prevalent in the scientific community [93]. The same constraints apply to the context of software documentation: the additional effort required to create interactive documents without specialized tools may outweigh the improvement in quality for readers.

We were conscious of this trade-off during the development of Casdoc. Several design decisions were motivated by the impact on the authoring process. We evaluated those design decisions by using Casdoc ourselves to author over a hundred documents. However, this experience only provides anecdotal evidence of the viability of Casdoc as an authoring tool. Future work should study the challenges faced by writers, and how tools such as Casdoc can mitigate these challenges, to complement research on software documentation with the document creators' perspective [13, 130].

6.2.3 Further Improvements to the Casdoc Format

Although Casdoc is already a viable format for documentation, our studies revealed some of its limitations that can be addressed in future versions of the format. We describe some of the improvements we already implemented in Section 5.7. We plan to continue the development of Casdoc, with some of the following improvements.

One benefit of non-interactive expanded documents over Casdoc is that readers can scan their content to get an overview of the topics and amount of details covered by the document. We can

address this limitation with a summarized representation of all annotations in Casdoc. Designing this summary may involve an exploration of text summarization and information prioritization techniques, to select which annotations to show in the summary and condense their content; topic inference strategies, to organize the summary based on latent semantic relations between annotations; and visual representations of information, to present the summarized information, with its non-linear structure, in an understandable representation.

Expanded documents also allow readers to find information they did not specifically seek, for example when trying to become more familiar with a topic. As the hybrid format we used in the laboratory study proved to be a desirable option, a future version of Casdoc could duplicate the content of some annotations to place it around the code example in an expanded form. This alternative representation of the content introduces the challenge of transforming a graph-based representation of information (i.e., annotations) into a coherent linear representation. In addition to contributing to the improvement of Casdoc, solutions to this challenge could help develop techniques to generate documentation from knowledge bases by aggregating a set of related entries into a single narrative. Solutions that are reversible, i.e., that allow to transform the linear representation into a graph of annotations, could support approaches that automatically convert existing documents into the Casdoc format. The development of these solutions should also improve our understanding of equivalent representations of knowledge.

The integration of additional knowledge bases to automatically generate annotation can improve the content of Casdoc documents while reducing the authoring effort. Integrating a knowledge base requires to investigate how to precisely associate its entries to specific anchors in the Casdoc document. This is a traceability recovery problem. Thus, developing this feature will both validate traceability recovery approaches for instances of the problem already studied in prior work, and suggest new approaches for contexts that are yet unexplored.

6.3 Documentation in the Age of Language Models

The rising popularity of language models in recent years has introduced a new alternative for software documentation. Conversational agents, such as ChatGPT, can generate on-demand documentation based on user queries (i.e., prompts). In the most recent Stack Overflow developer survey, 83% of respondents indicated having used ChatGPT in the last year [201]. Whether the current excitement around language models constitutes the start of a new paradigm or a marketing success to pass remains to be determined. Nevertheless, it encourages the community to reflect on new modes of interaction for documentation. We note two prominent ways in which language models can influence documentation practices: as a design option for the interface of interactive documentation, or as a tool to support writers in the creation of non-interactive documents. We conclude with a reflection

on how documentation research can affect the development of conversational agents, as opposed to being disrupted by them.

Conversational Agents as an Interface Design Option

Using conversational agent as a form of documentation achieves several documentation quality objectives. It can capture a large range of topics and technologies (i.e., high coverage) without overwhelming readers with too much information (i.e., conciseness). They can also provide varying amounts of background information to adapt to the expertise of its readers. Finally, it mitigates navigability issues, as readers interact with only two elements: an input component to receive queries, and an output component to show responses. In particular, it achieves many of the core design objectives of Casdoc, and there are some similarities between the user interface of the two formats. Both Casdoc and conversational agents show concise information to the reader, who can then query the document to find more information about confusing aspects of the initial information.

With such benefits, one may question if conversational agents will (or should) become the ubiquitous method for accessing documentation in a near future. The usefulness of conversational agents depends on the quality of user queries [39, 84] and on the agent's ability to understand the query [230]. Both aspects can be challenging and introduce usability issues.[3] In particular, during our laboratory study, we observed several instances of participants looking for general information about a topic they were unfamiliar with. In such cases, participants appreciated the ability to discover information they did not specifically look for. In a document authored by a human expert, readers can rely on the expert's judgment to know what information is necessary to learn a topic.

Using conversational agents as a design strategy for documentation can also have a negative impact on document creators. Training a large language model requires extensive computational resources that many creators cannot afford. The development of conversational agents may also conflict with the motivation of many documentation creators [13, 130]. For example, creators who publish documentation to establish and improve their reputation within a community would not benefit from creating a conversational agent, as it demonstrates an expertise in machine learning techniques rather than in the domain actually covered by the conversational agent.

Language Models as a Content Creation Aid

Language models can help support writers in the creation of new documents. For example, writers can use generative models to create the description of a concept for a tutorial or to improve the

[3]These issues add to other problems that are commonly discussed, such as the reliability and trustworthiness of answers. Language models require users to trust both the original sources of information and the algorithms to synthesize them. Although it is a valid argument against current language models, we are more interested in language models as a tool to design documents. Thus, we assume that document writers and readers can interact with trustworthy conversational agents.

language quality of a section. These applications leave a human expert in control of design aspects of the documentation (e.g., the selection of topics to cover and the organization of sections), but help with the generation of its content. In this case, language models affect some steps in the documentation creation process, but they do not constitute a fundamental paradigm shift. Therefore, although language models show a lot of potential to reduce the burden of documentation, they do not completely eliminate the human effort needed to create high-quality documents.

Documentation Research to Support Conversational Agents

Research on documentation design can help improve the user interface of conversational agents. For example, we designed Casdoc as a format for documents authored by humans, but conversational agents could also use a similar format. The user interface would include a component for the code example, either produced by a human expert or generated based on an initial query. The user would then be able to interact with specific elements of the code example to request additional automatically generated information. This interface would contextualize the agent's responses for the user, but also allow to clarify the context of the user's queries for the agent. Presenting information in annotations instead of a chat interface would also allow the user and the agent to follow multiple threads of information.

Chapter 7

Conclusion

Software documentation plays a critical role in sustaining rich ecosystems of reusable software components. In addition to the quality of a document's content, different design aspects affect a document's ability to effectively support sharing knowledge between developers. These aspects include the selection of topics, their organization across a set of documents, and the navigation features of the user interface.

We explored the design of software documentation to help writers improve the quality of documents. Creating documents for a large audience is challenging, as individual readers have varying needs and preferences. For example, if a tutorial about a library to perform cryptographic operations includes definitions of domain concepts to be more accessible to novice developers, domain experts may find it too verbose. Conversely, if the tutorial assumes prior knowledge of domain concepts, novices may find it incomplete. In this thesis, we focused on design strategies that promote interaction between readers and documents to address this challenge. Instead of creating rigid documents tailored to a hypothetical average reader, we studied techniques to create flexible documents that adapt to the preferences of individual readers.

We investigated three aspects of documentation design. First, we studied techniques to identify concepts that developers must be familiar with when using a software library. We called these concepts *conceptual dependencies* to denote the importance of developers understanding them to correctly use the library. We designed Scode, an approach that combines wikification and community search, to automatically identify conceptual dependencies represented by Wikipedia articles. Our evaluation of Scode and of state-of-the-art wikifiers revealed insights on how to extract structured knowledge from natural language documents and knowledge graphs. In particular, using a knowledge base to identify conceptual dependencies leads to more consistent and meaningful, but less precise results than extracting free-form concepts. We also found that selecting an optimal wikifier is challenging and that they are generally less accurate on software-specific inputs, but simple heuristics

(e.g., filtering the output with an inclusion list) can mitigate this limitation. The use of community search algorithms generates more cohesive sets of concepts than alternative options, such as latent Dirichlet allocation (LDA), but can generate very large sets around popular concepts such as SQL.

Second, we studied design variations in the content and organization of existing documents by comparing three introductory tutorials about the Android framework. Although the tutorials were all comprehensive, published by reputable organizations, and aimed at a similar audience, we identified eleven design decisions related to five design aspects, such as the length of code examples and the inclusion of peripheral topics to cover.

Finally, we designed a new format of documentation, called Casdoc, to explore the potential of web technologies to support interactive documents. Casdoc documents focus on a high-quality code example, and embed additional explanations about the code in dynamic annotations. We conducted a field study and a laboratory study to understand the strengths and limitations of interactive documentation designs. Our results demonstrate that Casdoc is a viable format for API learning resources. It allows writers to include large amounts of information in a single document that readers can access on demand. However, it also increases the probability that readers will miss important information if they do not specifically search for it. We also observed that optimal format designs depend on both the preferences of readers and the context of the search. These findings informed several improvements to the Casdoc formats that we already made or plan to make in the future.

This thesis demonstrates that many aspects of documentation design are insufficiently explored, despite their potential to improve software documents. The design variations elicited from the Android tutorials illustrate that even professional writers use various strategies, with an unclear impact on the audience, to design complex tutorials. The design and evaluation of Casdoc allowed us to elaborate guidelines, supported by empirical evidence, to help writers make informed decisions about some design aspects. Our investigation of conceptual dependencies provide techniques for writers to systematically select and review topics covered in a document. It also provides detailed insights to manipulate knowledge graphs, e.g., to automatically generate annotations for Casdoc documents.

We plan to pursue the development of Casdoc and to continue investigating better knowledge representation techniques in the future. We hope our effort encourages both researchers and practitioners to consider innovative strategies to present documentation.

Bibliography

[1] Surafel Lemma Abebe and Paolo Tonella. 2015. Extraction of domain concepts from the source code. *Science of Computer Programming* 98, 4 (2015), 680–706.

[2] Nahla J. Abid, Bonita Sharif, Natalia Dragan, Hend Alrasheed, and Jonathan I. Maletic. 2019. Developer Reading Behavior While Summarizing Java Methods: Size and Context Matters. In *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering*. 384–395.

[3] Silvia Abrahão, Carmine Gravino, Emilio Insfran, Giuseppe Scanniello, and Genoveffa Tortora. 2013. Assessing the Effectiveness of Sequence Diagrams in the Comprehension of Functional Requirements: Results from a Family of Five Experiments. *IEEE Transactions on Software Engineering* 39, 3 (2013), 327–342.

[4] Emad Aghajani, Csaba Nagy, Mario Linares-Vásquez, Laura Moreno, Gabriele Bavota, Michele Lanza, and David C. Shepherd. 2020. Software Documentation: The Practitioners' Perspective. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 590–601.

[5] Emad Aghajani, Csaba Nagy, Olga Lucero Vega-Márquez, Mario Linares-Vásquez, Laura Moreno, Gabriele Bavota, and Michele Lanza. 2019. Software Documentation Issues Unveiled. In *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering*. 1199–1210.

[6] Esra Akbas and Peixiang Zhao. 2017. Truss-based Community Search: a Truss-equivalence Based Indexing Approach. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1298–1309.

[7] Miltiadis Allamanis and Charles Sutton. 2013. Why, When, and What: Analyzing Stack Overflow Questions by Topic, Type, and Code. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. 53–56.

[8] Android Developers. 2020. *Developer guides*. Retrieved 2024-03-28 from https://developer.android.com/guide

[9] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand, and Frank Zimmer. 2017. Automated Extraction and Clustering of Requirements Glossary Terms. *IEEE Transactions on Software Engineering* 43, 10 (2017), 918–945.

[10] Deeksha M. Arya, Jin L. C. Guo, and Martin P. Robillard. 2020. Information Correspondence between Types of Documentation for APIs. *Empirical Software Engineering* 25, 5 (2020), 4069–4096.

[11] Deeksha M. Arya, Jin L. C. Guo, and Martin P. Robillard. 2023. How programmers find online learning resources. *Empirical Software Engineering* 28, 2, Article 23 (2023), 30 pages.

[12] Deeksha M. Arya, Jin L. C. Guo, and Martin P. Robillard. 2024. Properties and Styles of Software Technology Tutorials. *IEEE Transactions on Software Engineering* 50, 2 (2024), 159–172.

[13] Deeksha M. Arya, Jin L. C. Guo, and Martin P. Robillard. 2024. Why People Contribute to Software Documentation. In *Proceedings of the 17th International Conference on Cooperative and Human Aspects of Software Engineering*. 6 pages. To appear.

[14] Deeksha M. Arya, Mathieu Nassif, and Martin P. Robillard. 2021. *Appendix for "A Data-Centric Study of Software Tutorial Design"*. Retrieved 2024-03-07 from https://zenodo.org/records/5075903

[15] Deeksha M. Arya, Mathieu Nassif, and Martin P. Robillard. 2022. A Data-Centric Study of Software Tutorial Design. *IEEE Software* 39, 3 (2022), 106–115.

[16] T. K. Attwood, D. B. Kell, P. McDermott, J. Marsh, S. R. Pettifer, and D. Thorne. 2010. Utopia documents: linking scholarly literature with research data. *Bioinformatics* 26, 18 (2010), i568–i574.

[17] Sriram Karthik Badam, Zhicheng Liu, and Niklas Elmqvist. 2019. Elastic Documents: Coupling Text and Tables through Contextual Visualizations for Enhanced Document Reading. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 661–671.

[18] Gavin Bailey, Deepak Ranjan Sahoo, and Matt Jones. 2020. Digital Bookmark: Seamless Switching Between Printed and Electronic Books. In *Proceedings of the ACM Designing Interactive Systems Conference*. 885–894.

[19] Sebastian Baltes, Lorik Dumani, Christoph Treude, and Stephan Diehl. 2018. SOTorrent: Reconstructing and Analyzing the Evolution of Stack Overflow Posts. In *Proceedings of the ACM/IEEE 15th International Conference on Mining Software Repositories*. 319–330.

[20] Sebastian Baltes, Christoph Treude, and Martin P. Robillard. 2022. Contextual Documentation Referencing on Stack Overflow. *IEEE Transactions on Software Engineering* 48, 1 (2022), 135–149.

[21] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. 2014. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empirical Software Engineering* 19, 3 (2014), 619–654.

[22] Alex Bäuerle, Ángel Alexander Cabrera, Fred Hohman, Megan Maher, David Koski, Xavier Suau, Titus Barik, and Dominik Moritz. 2022. Symphony: Composing Interactive Interfaces for Machine Learning. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 210, 14 pages.

[23] David Beazley and Brian K. Jones. 2013. *Python Cookbook: Recipes for Mastering Python 3* (3 ed.). O'Reilly Media.

[24] Stefanie Beyer, Christian Macho, Massimiliano Di Penta, and Martin Pinzger. 2020. What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories. *Empirical Software Engineering* 25, 3 (2020), 2258–2301.

[25] Ted J. Biggerstaff, Bharat G. Mitbander, and Dallas Webster. 1993. The Concept Assignment Problem in Program Understanding. In *Proceedings of the Working Conference on Reverse Engineering*. 27–43.

[26] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3, Jan (2003), 993–1022.

[27] Abir Bouraffa and Walid Maalej. 2020. Two Decades of Empirical Research on Developers' Information Needs: A Preliminary Analysis. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 71–77.

[28] Pierre Bourque and Richard E. Fairley. 2014. *Guide to the Software Engineering Body of Knowledge* (3 ed.).

[29] Tom Boyle. 2003. Design principles for authoring dynamic, reusable learning objects. *Australasian Journal of Educational Technology* 19, 1 (2003), 46–58.

[30] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1589–1598.

[31] Janez Brank, Gregor Leban, and Marko Grobelnik. 2017. Annotating Documents with Relevant Wikipedia Concepts. In *Proceedings of the Slovenian Conference on Data Mining and Data Warehouses*. 4 pages.

[32] Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks* 30 (1998), 107–117.

[33] Marcel Bruch, Mira Mezini, and Martin Monperrus. 2010. Mining Subclassing Directives to Improve Framework Reuse. In *Proceedings of the 7th IEEE Working Conference on Mining Software Repositories*. 141–150.

[34] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, and William Jernigan. 2016. GenderMag: A Method for Evaluating Software's Gender Inclusiveness. *Interacting with Computers* 28, 6 (2016), 760–787.

[35] Raymond P. L. Buse and Westley Weimer. 2012. Synthesizing API Usage Examples. In *Proceedings of the 34th International Conference on Software Engineering*. 782–792.

[36] Liang Cai, Haoye Wang, Bowen Xu, Huang Qiao, Xin Xia, David Lo, and Zhenchang Xing. 2019. AnswerBot: An Answer Summary Generation Tool Based on Stack Overflow. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1134–1138.

[37] Tianyuan Cai, Shaun Wallace, Tina Rezvanian, Jonathan Dobres, Bernard Kerr, Samuel Berlow, Jeff Huang, Ben D. Sawyer, and Zoya Bylinskii. 2022. Personalized Font Recommendations: Combining ML and Typographic Guidelines to Optimize Readability. In *Proceedings of the ACM Designing Interactive Systems Conference*. 1–25.

[38] Eduardo C. Campos, Lucas B. L. de Souza, and Marcelo de A. Maia. 2016. Searching crowd knowledge to recommend solutions for API usage tasks. *Journal of Software: Evolution and Process* 28, 10 (2016), 863–892.

[39] Kaibo Cao, Chunyang Chen, Sebastian Baltes, Christoph Treude, and Xiang Chen. 2021. Automated Query Reformulation for Efficient Search Based on Query Logs From Stack Overflow. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering*. 1273–1285.

[40] John M. Carroll, Penny L. Smith-Kerker, James R. Ford, and Sandra A. Mazur-Rimetz. 1987. The Minimal Manual. *Human–Computer Interaction* 3, 2 (1987), 123–153.

[41] Danilo Carvalho, Çağatay Çallı, André Freitas, and Edward Curry. 2014. EasyESA: A Low-effort Infrastructure for Explicit Semantic Analysis. In *Proceedings of the 13th International Semantic Web Conference*. 177–180.

[42] Taylor Cassidy, Heng Ji, Lev-Arie Ratinov, Arkaitz Zubiaga, and Hongzhao Huang. 2012. Analysis and Enhancement of Wikification for Microblogs with Context Expansion. In *Proceedings of the 24th International Conference on Computational Linguistics*. 441–456.

[43] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-Graber, and David M. Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. In *Advances in Neural Information Processing Systems*. 288–296.

[44] Chunyang Chen and Zhenchang Xing. 2016. Mining Technology Landscape from Stack Overflow. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 1–10.

[45] Chunyang Chen, Zhenchang Xing, and Yang Liu. 2019. What's Spain's Paris? Mining analogical libraries from Q&A discussions. *Empirical Software Engineering* 24, 3 (2019), 1155–1194.

[46] Chunyang Chen, Zhenchang Xing, and Ximing Wang. 2017. Unsupervised Software-Specific Morphological Forms Inference from Informal Discussions. In *Proceedings of the 39th International Conference on Software Engineering*. 450–461.

[47] Nicholas Chen, Francois Guimbretiere, and Abigail Sellen. 2012. Designing a Multi-Slate Reading Environment to Support Active Reading Activities. *ACM Transactions on Computer-Human Interaction* 19, 3, Article 18 (2012), 35 pages.

[48] Xiao Cheng and Dan Roth. 2013. Relational Inference for Wikification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1787–1796.

[49] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2012. MixT: Automatic Generation of Step-by-Step Mixed Media Tutorials. In *Proceedings of the 25th annual ACM symposium on User Interface Software and Technology*. 93–102.

[50] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Nord Robert, and Judith Stafford. 2010. *Documenting Software Architectures: Views and Beyond* (2 ed.).

[51] Filipe Roseiro Cogo, Xin Xia, and Ahmed E. Hassan. 2023. Assessing the Alignment between the Information Needs of Developers and the Documentation of Programming Languages: A

Case Study on Rust. *ACM Transactions on Software Engineering and Methodology* 32, 2, Article 43 (2023), 48 pages.

[52] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46.

[53] Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. 2013. A Framework for Benchmarking Entity-Annotation Systems. In *Proceedings of the 22nd International Conference on World Wide Web*. 249–260.

[54] Alex Cummaudo, Rajesh Vasa, and John Grundy. 2019. What should I document? A preliminary systematic mapping study into API documentation knowledge. In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 6 pages.

[55] Bill Curtis, Sylvia B. Sheppard, Elizabeth Kruesi-Bailey, John Bailey, and Deborah A. Boehm-Davis. 1989. Experimental Evaluation of Software Documentation Formats. *Journal of Systems and Software* 9, 2 (1989), 167–207.

[56] Rodrigo Fernandes Gomes da Silva, Chanchal K. Roy, Mohammad Masudur Rahman, Kevin A. Schneider, Klérisson Paixão, Carlos Eduardo de Carvalho Dantas, and Marcelo de Almeida Maia. 2020. CROKAGE: Effective solution recommendation for programming tasks by leveraging crowd knowledge. *Empirical Software Engineering* 25, 6 (2020), 4707–4758.

[57] Barthélémy Dagenais and Martin P. Robillard. 2010. Creating and Evolving Developer Documentation: Understanding the Decisions of Open Source Contributors. In *Proceedings of the 18th ACM SIGSOFT international symposium on Foundations of software engineering*. 127–136.

[58] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving Efficiency and Accuracy in Multilingual Entity Extraction. In *Proceedings of the 9th International Conference on Semantic Systems*. 121–124.

[59] Biniam Fisseha Demissie, Mariano Ceccato, and Lwin Khin Shar. 2020. Security analysis of permission re-delegation vulnerabilities in Android apps. *Empirical Software Engineering* 25, 6 (2020), 5084–5136.

[60] Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk. 2013. Feature location in source code: a taxonomy and survey. *Journal of Software: Evolution and Process* 25, 1 (2013), 53–95.

[61] Pierre Dragicevic, Yvonne Jansen, Abhraneel Sarma, Matthew Kay, and Fanny Chevalier. 2019. Increasing the Transparency of Research Papers with Explorable Multiverse Analyses. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 65, 15 pages.

[62] Ekwa Duala-Ekoko and Martin P. Robillard. 2012. Asking and Answering Questions about Unfamiliar APIs: An Exploratory Study. In *Proceedings of the 34th International Conference on Software Engineering*. 266–276.

[63] Shreyosi Endow and Cesar Torres. 2021. "I'm Better Off on my Own": Understanding How a Tutorial's Medium Affects Physical Skill Development. In *Proceedings of the ACM Designing Interactive Systems Conference*. 1313–1323.

[64] Neil A. Ernst and Martin P. Robillard. 2023. A study of documentation for software architecture. *Empirical Software Engineering* 28, 5, Article 122 (2023), 23 pages.

[65] Saad Ezzini, Sallam Abualhaija, Chetan Arora, Mehrdad Sabetzadeh, and Lionel C. Briand. 2021. Using Domain-Specific Corpora for Improved Handling of Ambiguity in Requirements. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering*. 1485–1497.

[66] F-Droid Limited and Contributors. 2022. *F-Droid*. Retrieved 2024-01-02 from https://www.f-droid.org/

[67] J.-R. Falleri, M. Huchard, M. Lafourcade, C. Nebut, V. Prince, and M. Dao. 2010. Automatic Extraction of a WordNet-Like Identifier Network from Software. In *Proceedings of the 18th IEEE International Conference on Program Comprehension*. 4–13.

[68] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *The VLDB Journal* 29, 1 (2020), 353–392.

[69] K. J. Kevin Feng, Maxwell James Coppock, and David W. McDonald. 2023. How Do UX Practitioners Communicate AI as a Design Material? Artifacts, Conceptions, and Propositions. In *Proceedings of the ACM Designing Interactive Systems Conference*. 2263–2280.

[70] Mi Feng, Cheng Deng, Evan M. Peck, and Lane Harrison. 2018. The Effects of Adding Search Functionality to Interactive Visualizations on the Web. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 137, 13 pages.

[71] Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. 1625–1628.

[72] Alessio Ferrari and Andrea Esuli. 2019. An NLP approach for cross-domain ambiguity detection in requirements engineering. *Automated Software Engineering* 26, 3 (2019), 559–598.

[73] Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2014. Two Is Bigger (and Better) Than One: the Wikipedia Bitaxonomy Project. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 945–955.

[74] Adam Fourney and Michael Terry. 2014. Mining Online Software Tutorials: Challenges and Open Problems. In *Extended Abstracts on Human Factors in Computing Systems*. 653–664.

[75] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. 1606–1611.

[76] Zhipeng Gao, Xin Xia, David Lo, John Grundy, Xindong Zhang, and Zhenchang Xing. 2023. I Know What You Are Searching for: Code Snippet Recommendation from Stack Overflow Posts. *ACM Transactions on Software Engineering and Methodology* 32, 3, Article 80 (2023), 42 pages.

[77] R. Stuart Geiger, Nelle Varoquaux, Charlotte Mazel-Cabasse, and Chris Holdgraf. 2018. The Types, Roles, and Practices of Documentation in Data Analytics Open Source Software Libraries. *Computer Supported Cooperative Work* 27, 3-6 (2018), 767–802.

[78] Katy Ilonka Gero, Lydia Chilton, Chris Melancon, and Mike Cleron. 2022. Eliciting Gestures for Novel Note-taking Interactions. In *Proceedings of the ACM Designing Interactive Systems Conference*. 966–975.

[79] Elena L. Glassman, Tianyi Zhang, Björn Hartmann, and Miryung Kim. 2018. Visualizing API Usage Examples at Scale. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 580, 12 pages.

[80] Google. 2020. *Android Developer Fundamentals*. Retrieved 2024-03-07 from https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/index.html

[81] Daniel Graziotin, Per Lenberg, Robert Feldt, and Stefan Wagner. 2022. Psychometrics in Behavioral Software Engineering: A Methodological Introduction with Guidelines. *ACM Transactions on Software Engineering and Methodology* 31, 1, Article 7 (2022), 36 pages.

[82] John Gruber. 2004. *Daring Fireball: Markdown.* Retrieved 2024-03-07 from https://daringfireball.net/projects/markdown/

[83] Philip J. Guo. 2013. Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. In *Proceedings of the 44th ACM technical symposium on Computer science education*. 579–584.

[84] Sonia Haiduc, Gabriele Bavota, Andrian Marcus, Rocco Oliveto, Andrea De Lucia, and Tim Menzies. 2013. Automatic Query Reformulations for Text Retrieval in Software Engineering. In *Proceedings of the 35th International Conference on Software Engineering*. 842–851.

[85] Feng Han, Yifei Cheng, Megan Strachan, and Xiaojuan Ma. 2021. Hybrid Paper-Digital Interfaces: A Systematic Literature Review. In *Proceedings of the ACM Designing Interactive Systems Conference*. 1087–1100.

[86] Hideaki Hata, Christoph Treude, Raula Gaikovina Kula, and Takashi Ishio. 2019. 9.6 Million Links in Source Code Comments: Purpose, Evolution, and Decay. In *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering*. 1211–1221.

[87] Andrew Head, Jason Jiang, James Smith, Marti A. Hearst, and Björn Hartmann. 2020. Composing Flexibly-Organized Step-by-Step Tutorials from Linked Source Code, Snippets, and Outputs. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 669, 12 pages.

[88] Andrew Head, Caitlin Sadowski, Emerson Murphy-Hill, and Andrea Knight. 2018. When Not to Comment: Questions and Tradeoffs with API Documentation for C++ Projects. In *Proceedings of the 40th International Conference on Software Engineering*. 643–653.

[89] Andrew Head, Amber Xie, and Marti A. Hearst. 2022. Math Augmentation: How Authors Enhance the Readability of Formulas using Novel Visual Design Practices. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 491, 18 pages.

[90] Jeffrey Heer, Stuart K. Card, and James A. Landay. 2005. prefuse: A Toolkit for Interactive Information Visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 421–430.

[91] Keita Higuchi, Shotaro Sano, and Takeo Igarashi. 2021. Interactive Hyperparameter Optimization with Paintable Timelines. In *Proceedings of the ACM Designing Interactive Systems Conference*. 1518–1528.

[92] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing*. 782–792.

[93] Fred Hohman, Matthew Conlen, Jeffrey Heer, and Duen Horng (Polo) Chau. 2020. Communicating with Interactive Articles. *Distill* 5, 9 (2020), e28. https://distill.pub/2020/communicating-with-interactive-articles

[94] Michihiro Horie and Shigeru Chiba. 2010. Tool support for crosscutting concerns of API documentation. In *Proceedings of the 9th International Conference on Aspect-Oriented Software Development*. 97–108.

[95] Kasper Hornbæk and Erik Frøkjær. 2003. Reading Patterns and Usability in Visualizations of Electronic Documents. *ACM Transactions on Computer-Human Interaction* 10, 2 (2003), 119–149.

[96] Amber Horvath, Michael Xieyang Liu, River Hendriksen, Connor Shannon, Emma Paterson, Kazi Jawad, Andrew Macvean, and Brad A Myers. 2022. Understanding How Programmers Can Use Annotations on Documentation. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 69, 16 pages.

[97] Jiafeng Hu, Xiaowei Wu, Reynold Cheng, Siqiang Luo, and Yixiang Fang. 2017. On Minimal Steiner Maximum-Connected Subgraph Queries. *IEEE Transactions on Knowledge and Data Engineering* 29, 11 (2017), 2455–2469.

[98] Xing Hu, Qiuyuan Chen, Haoye Wang, Xin Xia, David Lo, and Thomas Zimmermann. 2022. Correlating Automated and Human Evaluation of Code Documentation Generation Quality. *ACM Transactions on Software Engineering and Methodology* 31, 4, Article 63 (2022), 28 pages.

[99] Qiao Huang, Xin Xia, Zhenchang Xing, David Lo, and Xinyu Wang. 2018. API Method Recommendation without Worrying about the Task-API Knowledge Gap. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 293–304.

[100] Xin Huang, Laks V. S. Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. 2015. Approximate Closest Community Search in Networks. *Proceedings of the VLDB Endowment* 9, 4 (2015), 276–287.

[101] ISO/IEC/IEEE. 2017. *International Standard – Systems and software engineering – Vocabulary.* Standard 24765:2017. ISO/IEC/IEEE.

[102] Heng Ji and Ralph Grishman. 2011. Knowledge Base Population: Successful Approaches and Challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 1148–1158.

[103] Matthew B. Kelly, Jason S. Alexander, Bram Adams, and Ahmed E. Hassan. 2011. Recovering a Balanced Overview of Topics in a Software Domain. In *Proceedings of the IEEE 11th International Working Conference on Source Code Analysis and Manipulation*. 135–144.

[104] Maurice G Kendall. 1938. A New Measure of Rank Correlation. *Biometrika* 30, 1/2 (1938), 81–93.

[105] Kandarp Khandwala and Philip J. Guo. 2018. Codemotion: Expanding the Design Space of Learner Interactions with Computer Programming Tutorial Videos. In *Proceedings of the 5th Annual ACM Conference on Learning at Scale*. Article 57, 10 pages.

[106] Kimia Kiani, George Cui, Andrea Bunt, Joanna McGrenere, and Parmit K. Chilana. 2019. Beyond "One-Size-Fits-All": Understanding the Diversity in How Software Newcomers Discover and Make Use of Help Resources. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–14.

[107] Dae Hyun Kim, Enamul Hoque, Juho Kim, and Maneesh Agrawala. 2018. Facilitating Document Reading by Linking Text and Tables. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 423–434.

[108] Amy J. Ko, Brad A. Myers, Michael J. Coblenz, and Htet Htet Aung. 2006. An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks. *IEEE Transactions on Software Engineering* 32, 12 (2006), 971–987.

[109] Amy J. Ko and Yann Riche. 2011. The Role of Conceptual Knowledge in API Usability. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*. 173–176.

[110] Amy J. Ko and Bob Uttl. 2003. Individual Differences in Program Comprehension Strategies in Unfamiliar Programming Systems. In *Proceedings of the 11th IEEE International Workshop on Program Comprehension*. 175–184.

[111] Balasaravanan Thoravi Kumaravel, Cuong Nguyen, Stephen DiVerdi, and Björn Hartmann. 2019. TutoriVR: A Video-Based Tutorial System for Design Applications in Virtual Reality. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 284, 12 pages.

[112] J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (1977), 159–174.

[113] Tien-Duy B. Le, Richard J. Oentaryo, and David Lo. 2015. Information Retrieval and Spectrum Based Bug Localization: Better Together. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*. 579–590.

[114] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

[115] Timothy C. Lethbridge, Janice Singer, and Andrew Forward. 2003. How Software Engineers Use Documentation: The State of the Practice. *IEEE Software* 20, 6 (2003), 35–39.

[116] Hongwei Li, Sirui Li, Jiamou Sun, Zhenchang Xing, Xin Peng, Mingwei Liu, and Xuejiao Zhao. 2018. Improving API Caveats Accessibility by Mining API Caveats Knowledge Graph. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*. 183–193.

[117] Chen Liang, Anhong Guo, and Jeeeun Kim. 2022. CustomizAR: Facilitating Interactive Exploration and Measurement of Adaptive 3D Designs. In *Proceedings of the ACM Designing Interactive Systems Conference*. 898–912.

[118] Eden Litt. 2012. Knock, Knock. Who's There? The Imagined Audience. *Journal of Broadcasting & Electronic Media* 56, 3 (2012), 330–345.

[119] Jiakun Liu, Sebastian Baltes, Christoph Treude, David Lo, Yun Zhang, and Xin Xia. 2021. Characterizing Search Activities on Stack Overflow. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 919–931.

[120] Mingwei Liu, Xin Peng, Andrian Marcus, Shuangshuang Xing, Christoph Treude, and Chengyuan Zhao. 2022. API-Related Developer Information Needs in Stack Overflow. *IEEE Transactions on Software Engineering* 48, 11 (2022), 4485–4500.

[121] Mingwei Liu, Xin Peng, Andrian Marcus, Zhenchang Xing, Wenkai Xie, Shuangshuang Xing, and Yang Liu. 2019. Generating Query-Specific Class API Summaries. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 120–130.

[122] Yang Liu, Mingwei Liu, Xin Peng, Christoph Treude, Zhenchang Xing, and Xiaoxin Zhang. 2020. Generating Concept based API Element Comparison Using a Knowledge Graph. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 834–845.

[123] Ziyi Liu, Zhengzhe Zhu, Enze Jiang, Feichi Huang, Anna M. Villanueva, Xun Qian, Tianyi Wang, and Karthik Ramani. 2023. InstruMentAR: Auto-Generation of Augmented Reality Tutorials for Operating Digital Instruments Through Recording Embodied Demonstration. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 32, 17 pages.

[124] Dmitry Lizorkin, Olena Medelyan, and Maria Grineva. 2009. Analysis of Community Structure in Wikipedia. In *Proceedings of the 18th International Conference on World Wide Web*. 1221–1222.

[125] J. D. Long and Paul Teetor. 2019. *R Cookbook: Proven Recipes for Data Analysis, Statistics & Graphics* (2 ed.). O'Reilly Media.

[126] Lori Lorigo, Bing Pan, Helene Hembrooke, Thorsten Joachims, Laura Granka, and Geri Gay. 2006. The influence of task and gender on search and evaluation behavior using Google. *Information Processing & Management* 42, 4 (2006), 1123–1131.

[127] Suyu Ma, Zhenchang Xing, Chunyang Chen, Cheng Chen, Lizhen Qu, and Guoqiang Li. 2021. Easy-to-Deploy API Extraction by Multi-Level Feature Embedding and Transfer Learning. *IEEE Transactions on Software Engineering* 47, 10 (2021), 2296–2311.

[128] Walid Maalej and Martin P. Robillard. 2013. Patterns of Knowledge in API Reference Documentation. *IEEE Transactions on Software Engineering* 39, 9 (2013), 1264–1282.

[129] Walid Maalej, Rebecca Tiarks, Tobias Roehm, and Rainer Koschke. 2014. On the Comprehension of Program Comprehension. *ACM Transactions on Software Engineering and Methodology* 23, 4, Article 31 (2014), 37 pages.

[130] Laura MacLeod, Andreas Bergen, and Margaret-Anne Storey. 2017. Documenting and sharing software knowledge using screencasts. *Empirical Software Engineering* 22, 3 (2017), 1478–1507.

[131] Laura MacLeod, Margaret-Anne Storey, and Andreas Bergen. 2015. Code, Camera, Action: How Software Developers Document and Share Program Knowledge Using YouTube. In *Proceedings of the IEEE 23rd International Conference on Program Comprehension*. 104–114.

[132] H. B. Mann and D. R. Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18, 1 (1947), 50–60.

[133] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. 2023. Charagraph: Interactive Generation of Charts for Realtime Annotation of Data-Rich Paragraphs. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 146, 18 pages.

[134] Damien Masson, Sylvain Malacria, Edward Lank, and Géry Casiez. 2020. Chameleon: Bringing Interactivity to Static Digital Documents. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 432, 13 pages.

[135] Paul W. McBurney and Collin McMillan. 2014. Automatic Documentation Generation via Source Code Summarization of Method Context. In *Proceedings of the 22nd International Conference on Program Comprehension*. 279–290.

[136] Andrew Kachites McCallum. 2002. *MALLET: A Machine Learning for Language Toolkit*. Retrieved 2024-01-02 from http://mallet.cs.umass.edu

[137] Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. 2012. Adding Semantics to Microblog Posts. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining*. 563–572.

[138] Hans Meij, Joyce Karreman, and Michaël Steehouder. 2009. Three Decades of Research and Professional Practice on Printed Software Tutorials for Novices. *Technical Communication* 56, 3 (2009), 265–292.

[139] David A. Mellis, Ben Zhang, Audrey Leung, and Björn Hartmann. 2017. Machine Learning for Makers: Interactive Sensor Data Classification Based on Augmented Code Examples. In *Proceedings of the Conference on Designing Interactive Systems*. 1213–1225.

[140] Pablo N. Mendes, Max Jakob, Andres Garcia-Silva, and Christian Bizer. 2011. DBpedia Spotlight: Shedding Light on the Web of Documents. In *Proceedings of the 7th International Conference on Semantic Systems*. 1–8.

[141] Michael Meng, Stephanie Steinhardt, and Andreas Schubert. 2018. Application Programming Interface Documentation: What Do Software Developers Want? *Journal of Technical Writing and Communication* 48, 3 (2018), 295–330.

[142] Michael Meng, Stephanie M. Steinhardt, and Andreas Schubert. 2020. Optimizing API Documentation: Some Guidelines and Effects. In *Proceedings of the 38th ACM International Conference on Design of Communication*. Article 24, 11 pages.

[143] Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The Senseval-3 English lexical sample task. In *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*. 25–28.

[144] Bradley N. Miller and David L. Ranum. 2012. Beyond PDF and ePub: Toward an Interactive Textbook. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*. 150–155.

[145] David Milne and Ian H. Witten. 2008. Learning to link with Wikipedia. In *Proceedings of the 17th ACM conference on Information and Knowledge Management*. 509–518.

[146] David Milne and Ian H. Witten. 2013. An open-source toolkit for mining Wikipedia. *Artificial Intelligence* 194 (2013), 222–239.

[147] Aliaksei Miniukovich, Antonella De Angeli, Simone Sulpizio, and Paola Venuti. 2017. Design Guidelines for Web Readability. In *Proceedings of the Conference on Designing Interactive Systems*. 285–296.

[148] Daniel Moody and Jos van Hillegersberg. 2008. Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams. In *Proceedings of the International Conference on Software Language Engineering*. 16–34.

[149] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics* 2 (2014), 231–244.

[150] Parisa Moslehi, Juergen Rilling, and Bram Adams. 2022. A user survey on the adoption of crowd-based software engineering instructional screencasts by the new generation of software developers. *Journal of Systems and Software* 185, Article 111144 (2022), 21 pages.

[151] Sarah Nadi and Christoph Treude. 2020. Essential Sentences for Navigating Stack Overflow Answers. In *Proceedings of the IEEE 27th International Conference on Software Analysis, Evolution and Reengineering*. 229–239.

[152] Seyed Mehdi Nasehi, Jonathan Sillito, Frank Maurer, and Chris Burns. 2012. What Makes a Good Code Example? A Study of Programming Q&A in StackOverflow. In *Proceedings of the 28th IEEE International Conference on Software Maintenance*. 25–34.

[153] Mathieu Nassif, Alexa Hernandez, Ashvitha Sridharan, and Martin P. Robillard. 2022. Generating Unit Tests for Documentation. *IEEE Transactions on Software Engineering* 48, 9 (2022), 3268–3279.

[154] Mathieu Nassif, Zara Horlacher, and Martin P. Robillard. 2022. Casdoc: Unobtrusive Explanations in Code Examples. In *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*. 631–635.

[155] Mathieu Nassif and Martin P. Robillard. 2021. *Replication package for "Wikifying Software Artifacts"*. Retrieved 2024-03-07 from https://zenodo.org/records/4442458

[156] Mathieu Nassif and Martin P. Robillard. 2021. Wikifying Software Artifacts. *Empirical Software Engineering* 26, 2, Article 31 (2021), 31 pages.

[157] Mathieu Nassif and Martin P. Robillard. 2023. *Artifact for "Identifying Concepts in Software Projects"*. Retrieved 2024-03-16 from https://zenodo.org/records/7835197

[158] Mathieu Nassif and Martin P. Robillard. 2023. A Field Study of Developer Documentation Format. In *Extended Abstracts of the ACM CHI Conference on Human Factors in Computing Systems*. Article 7, 7 pages.

[159] Mathieu Nassif and Martin P. Robillard. 2023. Identifying Concepts in Software Projects. *IEEE Transactions on Software Engineering* 49, 7 (2023), 3660–3674.

[160] Mathieu Nassif and Martin P. Robillard. 2023. Non Linear Software Documentation with Interactive Code Examples. *ACM Transactions on Software Engineering and Methodologies* (2023), 29 pages. https://arxiv.org/abs/2311.18057 Accepted pending minor revision.

[161] Mathieu Nassif and Martin P. Robillard. 2024. *Appendix to "Evaluating Interactive Documentation for Programmers"*. Retrieved 2024-03-07 from https://zenodo.org/doi/10.5281/zenodo.10637078

[162] Mathieu Nassif and Martin P. Robillard. 2024. Evaluating Interactive Documentation for Programmers. *Empirical Software Engineering* (2024), 25 pages. Under review.

[163] Mathieu Nassif, Christoph Treude, and Martin P. Robillard. 2020. Automatically Categorizing Software Technologies. *IEEE Transactions on Software Engineering* 46, 1 (2020), 20–32.

[164] Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 Task 12: Multilingual Word Sense Disambiguation. In *Second Joint Conference on Lexical and Computational Semantics, Volume 2: Proceedings of the 7th International Workshop on Semantic Evaluation*. 222–231.

[165] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence* 193 (2012), 217–250.

[166] Stephen Oney and Joel Brandt. 2012. Codelets: Linking Interactive Documentation and Example Code in the Editor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2697–2706.

[167] Oracle. 2022. *Lesson: JDBC Basics*. Retrieved 2024-03-08 from https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

[168] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report 1999-66. Stanford InfoLab.

[169] Fabio Palomba, Pasquale Salza, Adelina Ciurumelea, Sebastiano Panichella, Harald Gall, Filomena Ferrucci, and Andrea De Lucia. 2017. Recommending and Localizing Change Requests for Mobile Apps Based on User Reviews. In *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering*. 106–117.

[170] Chris Parnin, Christoph Treude, and Margaret-Anne Storey. 2013. Blogging Developer Knowledge: Motivations, Challenges, and Future Directions. In *Proceedings of the 21st International Conference on Program Comprehension*. 211–214.

[171] Sangameshwar Patil. 2017. Concept-Based Classification of Software Defect Reports. In *Proceedings of the 14th International Conference on Mining Software Repositories*. 182–186.

[172] Maksym Petrenko, Václav Rajlich, and Radu Vanciu. 2008. Partial Domain Comprehension in Software Evolution and Maintenance. In *Proceedings of the 16th IEEE International Conference on Program Comprehension*. 13–22.

[173] Francesco Piccinno and Paolo Ferragina. 2014. From TagME to WAT: a new Entity Annotator. In *Proceedings of the first International Workshop on Entity Recognition & Disambiguation*. 55–62.

[174] Peter Pirolli and Stuart Card. 1999. Information Foraging. *Psychological Review* 106, 4 (1999), 643–675.

[175] Luca Ponzanelli, Alberto Bacchelli, and Michele Lanza. 2013. Seahawk: Stack Overflow in the IDE. In *Proceedings of the 35th International Conference on Software Engineering*. 1295–1298.

[176] Luca Ponzanelli, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Michele Lanza. 2014. Prompter: A Self-Confident Recommender System. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*. 577–580.

[177] Martin F. Porter. 1980. An algorithm for suffix stripping. *Program* 14, 3 (1980), 130–137.

[178] Denys Poshyvanyk, Malcom Gethers, and Andrian Marcus. 2012. Concept Location Using Formal Concept Analysis and Information Retrieval. *ACM Transactions on Software Engineering and Methodology* 21, 4, Article 23 (2012), 34 pages.

[179] Princeton University. 2010. *WordNet: A Lexical Database for English.* Retrieved 2024-03-17 from https://wordnet.princeton.edu/

[180] Daniele Procida. 2017. *Diátaxis documentation framework.* Retrieved 2024-03-18 from https://diataxis.fr/

[181] Mohammad Masudur Rahman and Chanchal K. Roy. 2018. Improving IR-Based Bug Localization with Context-Aware Query. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.* 621–632.

[182] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies – Volume 1.* 1375–1384.

[183] Daniel Ratiu, Martin Feilkas, and Jan Jurjens. 2008. Extracting Domain Ontologies from Domain Specific APIs. In *Proceedings of the 12th European Conference on Software Maintenance and Reengineering.* 203–212.

[184] Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. YAGO: A Multilingual Knowledge Base from Wikipedia, Wordnet, and Geonames. In *Proceedings of the International Semantic Web Conference.* 177–185.

[185] Xiaoxue Ren, Jiamou Sun, Zhenchang Xing, Xin Xia, and Jianling Sun. 2020. Demystify Official API Usage Directives with Crowdsourced API Misuse Scenarios, Erroneous Code Examples and Patches. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering.* 925–936.

[186] Peter C. Rigby and Martin P. Robillard. 2013. Discovering Essential Code Elements in Informal Documentation. In *Proceedings of the 35th IEEE/ACM International Conference on Software Engineering.* 832–841.

[187] Martin P. Robillard. 2009. What Makes APIs Hard to Learn? Answers from Developers. *IEEE Software* 26, 6 (2009), 27–34.

[188] Martin P. Robillard. 2022. *Introduction to Software Design with Java* (2 ed.). Springer.

[189] Martin P. Robillard and Robert DeLine. 2011. A field study of API learning obstacles. *Empirical Software Engineering* 16, 6 (2011), 703–732.

[190] Martin P. Robillard, Andrian Marcus, Christoph Treude, Gabriele Bavota, Oscar Chaparro, Neil Ernst, Marco Aurélio Gerosa, Michael Godfrey, Michele Lanza, Mario Linares-Vásquez, Gail C. Murphy, Laura Moreno, David Shepherd, and Edmund Wong. 2017. On-Demand Developer Documentation. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*. 479–483.

[191] Martin P. Robillard and Christoph Treude. 2020. Understanding Wikipedia as a Resource for Opportunistic Learning of Computing Concepts. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 72–78.

[192] Christoffer Rosen and Emad Shihab. 2016. What are mobile developers asking about? A large scale study using Stack Overflow. *Empirical Software Engineering* 21, 3 (2016), 1192–1223.

[193] Peter J. Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* 20 (1987), 53–65.

[194] Dominic Seyler, Tatiana Dembelova, Luciano Del Corro, Johannes Hoffart, and Gerhard Weikum. 2018. A Study of the Importance of External Knowledge in the Named Entity Recognition Task. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 241–246.

[195] Zohreh Sharafi, Yu Huang, Kevin Leach, and Westley Weimer. 2021. Toward an Objective Measure of Developers' Cognitive Activities. *ACM Transactions on Software Engineering and Methodology* 30, 3, Article 30 (2021), 40 pages.

[196] Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. *IEEE Transactions on Knowledge and Data Engineering* 27, 2 (2015), 443–460.

[197] Jonathan Sillito, Gail C. Murphy, and Kris De Volder. 2008. Asking and Answering Questions during a Programming Change Task. *IEEE Transactions on Software Engineering* 34, 4 (2008), 434–451.

[198] Robert R. Sokal and Charles D. Michener. 1958. A Statistical Method for Evaluating Systematic Relationships. *The University of Kansas Science Bulletin* 38 (1958), 1409–1438.

[199] Mauro Sozio and Aristides Gionis. 2010. The Community-search Problem and How to Plan a Successful Cocktail Party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. 939–948.

[200] Stack Exchange Inc. 2024. *All Sites - Stack Exchange*. Retrieved 2024-03-18 from https://stackexchange.com/sites

[201] Stack Overflow Labs. 2023. *Stack Overflow Developer Survey 2023*. Retrieved 2024-03-23 from https://survey.stackoverflow.co/2023/

[202] Sarah Sterman, Molly Jane Nicholas, Janaki Vivrekar, Jessica R. Mindel, and Eric Paulos. 2023. Kaleidoscope: A Reflective Documentation Tool for a User Interface Design Course. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 702, 19 pages.

[203] Klaas-Jan Stol and Brian Fitzgerald. 2018. The ABC of Software Engineering Research. *ACM Transactions on Software Engineering and Methodology* 27, 3, Article 11 (2018), 51 pages.

[204] Jeffrey Stylos, Andrew Faulring, Zizhuang Yang, and Brad A. Myers. 2009. Improving API Documentation Using API Usage Information. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*. 119–126.

[205] Beth M. Sundheim. 1995. Overview of Results of the MUC-6 Evaluation. In *Proceedings of the 6th Conference on Message Understanding*. 13–31.

[206] Julian Szymański and Maciej Naruszewicz. 2019. Review on Wikification methods. *AI Communications* 32, 3 (2019), 235–251.

[207] Craig S. Tashman and W. Keith Edwards. 2011. LiquidText: A Flexible, Multitouch Environment to Support Active Reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3285–3294.

[208] Jaime Teevan, Christine Alvarado, Mark S. Ackerman, and David R. Karger. 2004. The Perfect Search Engine Is Not Enough: A Study of Orienteering Behavior in Directed Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 415–422.

[209] Maartje ter Hoeve, Robert Sim, Elnaz Nouri, Adam Fourney, Maarten de Rijke, and Ryen W. White. 2020. Conversations with Documents: An Exploration of Document-Centered Assistance. In *Proceedings of the Conference on Human Information Interaction and Retrieval*. 43–52.

[210] The MITRE Corporation. 1999. *CVE Website*. Retrieved 2024-03-26 from https://www.cve.org/

[211] The R Foundation. 2024. *R: What is R?* Retrieved 2024-03-22 from https://www.r-project.org/about.html

[212] Stephen W. Thomas, Hadi Hemmati, Ahmed E. Hassan, and Dorothea Blostein. 2014. Static test case prioritization using topic models. *Empirical Software Engineering* 19, 1 (2014), 182–212.

[213] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL*. 142–147.

[214] Kashyap Todi, Luis A. Leiva, Daniel Buschek, Pin Tian, and Antti Oulasvirta. 2021. Conversations with GUIs. In *Proceedings of the ACM Designing Interactive Systems Conference*. 1447–1457.

[215] Christoph Treude and Martin P. Robillard. 2016. Augmenting API Documentation with Insights from Stack Overflow. In *Proceedings of the 38th International Conference on Software Engineering*. 392–403.

[216] Gias Uddin and Martin P. Robillard. 2015. How API Documentation Fails. *IEEE Software* 32, 4 (2015), 68–75.

[217] Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, Paolo Ferragina, Christiane Lemke, Andrea Moro, Roberto Navigli, Francesco Piccinno, Giuseppe Rizzo, Harald Sack, René Speck, Raphaël Troncy, Jörg Waitelonis, and Lars Wesemann. 2015. GERBIL: General Entity Annotator Benchmarking Framework. In *Proceedings of the 24th International Conference on World Wide Web*. 1133–1143.

[218] Hans van der Meij, Joyce Karreman, and Michaël Steehouder. 2009. Three Decades of Research and Professional Practice on Printed Software Tutorials for Novices. *Technical Communication* 56, 3 (2009), 265–292.

[219] Hans van der Meij and Jan van der Meij. 2014. A comparison of paper-based and video tutorials for software learning. *Computers & Education* 78 (2014), 150–159.

[220] J. van der Meij and H. van der Meij. 2015. A test of the design of a video tutorial for software training. *Journal of Computer Assisted Learning* 31, 2 (2015), 116–132.

[221] Bret Victor. 2011. Explorable Explanations. https://worrydream.com/ExplorableExplanations/.

[222] Bret Victor. 2012. Learnable Programming. https://worrydream.com/LearnableProgramming/.

[223] Melina Vidoni and Zadia Codabux. 2023. Towards a taxonomy of Roxygen documentation in R packages. *Empirical Software Engineering* 28, 4, Article 106 (2023), 48 pages.

[224] Nicolas Vincent, Isaac Johnson, and Brent Hecht. 2018. Examining Wikipedia With a Broader Lens: Quantifying the Value of Wikipedia's Relationship with Other Large-Scale Online Communities. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. Article 566, 13 pages.

[225] Lars Vogel. 2020. *Android Development Tutorials*. Retrieved 2024-03-28 from https://www.vogella.com/tutorials/android.html

[226] Denny Vradečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[227] Chong Wang, Xin Peng, Mingwei Liu, Zhenchang Xing, Xuefang Bai, Bing Xie, and Tuo Wang. 2019. A Learning-Based Approach for Automatic Construction of Domain Glossary from Source Code and Documentation. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 97–108.

[228] Chong Wang, Xin Peng, Zhenchang Xing, Yue Zhang, Mingwei Liu, Rong Luo, and Xiujie Meng. 2023. XCoS: Explainable Code Search Based on Query Scoping and Knowledge Graph. *ACM Transactions on Software Engineering and Methodology* 32, 6, Article 140 (2023), 28 pages.

[229] Lu Wang, Xiaobing Sun, Jingwei Wang, Yucong Duan, and Li Bin. 2017. Construct Bug Knowledge Graph for Bug Resolution. In *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering Companion*. 189–191.

[230] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. *ArXiv preprint* (2023), 19 pages. arXiv:2302.11382

[231] Wikipedia. 2019. *Wikipedia:Manual of Style/Linking*. Retrieved 2024-03-28 from https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking

[232] David Wong-Aitken, Diana Cukierman, and Parmit K. Chilana. 2022. "It Depends on Whether or Not I'm Lucky" How Students in an Introductory Programming Course Discover, Select, and Assess the Utility of Web-Based Resources. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education*. 512–518.

[233] Di Wu, Xiao-Yuan Jing, Hongyu Zhang, Yang Feng, Haowen Chen, Yuming Zhou, and Baowen Xu. 2023. Retrieving API Knowledge from Tutorials and Stack Overflow Based on Natural Language Queries. *ACM Transactions on Software Engineering and Methodology* 32, 5, Article 109 (2023), 36 pages.

[234] Wan-Ching Wu, Diane Kelly, and Avneesh Sud. 2014. Using Information Scent and Need for Cognition to Understand Online Search Behavior. In *Proceedings of the 37th International ACM SIGIR conference on Research & development in information retrieval*. 557–566.

[235] Tao Xiao, Sebastian Baltes, Hideaki Hata, Christoph Treude, Raula Gaikovina Kula, Takashi Ishio, and Kenichi Matsumoto. 2023. 18 million links in commits messages: purpose, evolution, and decay. *Empirical Software Engineering* 28, 4, Article 91 (2023), 29 pages.

[236] Guangxu Xun, Xiaowei Jia, Vishrawas Gopalakrishnan, and Aidong Zhang. 2017. A Survey on Context Learning. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2017), 38–56.

[237] Masahiro Yamaguchi, Shohei Mori, Peter Mohr, Markus Tatzgern, Ana Stanescu, Hideo Saito, and Denis Kalkofen. 2020. Video-Annotated Augmented Reality Assembly Tutorials. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software Technology*. 1010–1022.

[238] Litao Yan, Miryung Kim, Björn Hartmann, Tianyi Zhang, and Elena L. Glassman. 2022. Concept-Annotated Examples for Library Comparison. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. Article 65, 16 pages.

[239] Jinqui Yang and Lin Tan. 2014. SWordNet: Inferring semantically related words from software context. *Empirical Software Engineering* 19, 6 (2014), 1856–1886.

[240] Deheng Ye, Lingfeng Bao, Zhenchang Xing, and Shang-Wei Lin. 2018. APIReal: an API recognition and linking approach for online developer forums. *Empirical Software Engineering* 23, 6 (2018), 3129–3160.

[241] Deheng Ye, Zhenchang Xing, Chee Yong Foo, Zi Qun Ang, Jing Li, and Nachiket Kapre. 2016. Software-Specific Named Entity Recognition in Software Engineering Social Content. In *Proceedings of the IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering*. 90–101.

[242] Zining Ye, Xinran Yuan, Shaurya Gaur, Aaron Halfaker, Jodi Forlizzi, and Haiyi Zhu. 2021. Wikipedia ORES Explorer: Visualizing Trade-offs For Designing Applications With Machine Learning API. In *Proceedings of the ACM Designing Interactive Systems Conference*. 1554–1565.

[243] Annie T. T. Ying and Martin P. Robillard. 2014. Selection and Presentation Practices for Code Example Summarization. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on the Foundations of Software Engineering*. 460–471.

[244] Haoxiang Zhang, Shaowei Wang, Tse-Hsun (Peter) Chen, and Ahmed E. Hassan. 2021. Are Comments on Stack Overflow Well Organized for Easy Retrieval by Developers? *ACM Transactions on Software Engineering and Methodology* 30, 2, Article 22 (2021), 31 pages.

[245] Xuejiao Zhao, Zhenchang Xing, Muhammad Ashad Kabir, Naoya Sawada, Jing Li, and Shang-Wei Lin. 2017. HDSKG: Harvesting Domain Specific Knowledge Graph from Content of Webpages. In *Proceedings of the IEEE 24th International Conference on Software Analysis, Evolution and Reengineering*. 56–67.

[246] Cheng Zhou. 2018. Intelligent Bug Fixing with Software Bug Knowledge Graph. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 944–947.

Appendix A

Replication Data for Conceptual Dependencies

We published the data necessary to verify and replicate our comparison of wikifiers (Section 3.2) at https://zenodo.org/records/4442458 [155], and the data related to our evaluation of Scode (Section 3.5) at https://zenodo.org/records/7835197 [157]. Table A.1 details the content of each artifact. The last column indicates which parts of each artifact is reproduced in the sections of this appendix.

Table A.1: Content of the Data Artifacts for Our Studies of Conceptual Dependencies

Path	Description	Sec.
	https://zenodo.org/records/4442458	
README.md	Description of the artifact.	
computing-articles.txt	List of all Wikipedia articles marked as *correct* at least once.	
stackoverflow-posts-sample.txt	List of the 500 Stack Overflow post IDs used for the evaluation.	
Wikifications/	Folder containing the output of each wikifier on the 500 Stack Overflow posts. The folder contains 19 files, each one corresponding to one configuration of one wikifier.	
Annotation_Guide/ Wikifier Annotation Guide.md	Guidelines provided to the annotators to validate the wikification results.	A.1
Annotation_Guide/ sample-annotations-5posts.xlsx	Set of five posts already annotated to serve as an example for annotators.	A.2
Annotations/	Folder containing the results of the annotation tasks. The folder contains a file readme.txt that describes each annotation set.	
	https://zenodo.org/records/7835197	
README.md	Description of the artifact.	
Concept_Relatedness/results-[...].tsv	Output of Scode and the two other techniques on the 100 Java classes.	
Concept_Relatedness/coding-guide.md	Guidelines provided to the annotators to validate the relatedness of the class–concept pairs.	A.3
Concept_Relatedness/codes-[...].tsv	Results of the annotation tasks.	
Concept_Relatedness/resolved-[...].tsv	Outcome of the conflict resolution phase.	
Wikipedia_Graph/nodes.tsv	List of 6746 Wikipedia articles related to computing.	A.4
Wikipedia_Graph/edges.tsv	Undirected edges of the computing-specific Wikipedia subgraph.	
Topic_Cohesiveness/android-projects.txt	List of the 227 open source Android projects used during the evaluation.	A.5
Topic_Cohesiveness/sample-[...].tsv	Sample of 45 topics (i.e., sets of concepts) from each concept aggregation technique.	
Topic_Cohesiveness/annotations-[...].xlsx	Results of the cohesiveness scores and word intrusion tasks.	
Topic_Cohesiveness/keys.xlsx	Identification of the true outlier for the word intrusion tasks.	

A.1 Wikifier Annotation Guide

Content of Annotation_Guide/Wikifier Annotation Guide.md *from* https://zenodo.org/records/4442458

Context

We are investigating the ability of automated approaches to identify *concepts*, represented by Wikipedia articles, in Stack Overflow *posts*. We need your help to evaluate the performance of these automated approaches.

Your task is to annotate *concepts* found in Stack Overflow *posts*.

Data

You received an Excel file with the following columns:

- *A: URL*: URL to the Stack Overflow *post* or Wikipedia article describing the *concept* (this column is hidden);
- *B: Clickable URL*: Clickable URL from column A (for convenience);
- *C: Post*: Stack Overflow post ID;
- *D: Concept*: Concept, i.e., Wikipedia article, associated with the post ID;
- *E: Match*: Your annotation (yes or no);

Task

For each pair of Stack Overflow post and Wikipedia article, answer the following question:

Is the concept represented by the Wikipedia article ***related to computing*** *AND* ***explicitly mentioned*** *in the Stack Overflow post?*

Do not necessarily rely on the title of the Wikipedia article to guess its meaning. The title can redirect to another article or point to a missing (deleted) article. For some articles, especially on more general concepts, it is necessary to briefly look at the article to grasp what it refers to. Some special cases, such as identifying synonyms, may also require perusal of the article (see below for details).

"Related to computing"

A concept is *related to computing* if, in the context of the post, the concept implies notions of programming, software development, theory of computation, or information technologies.

Special Cases:

1. Disambiguation pages, by their nature, do not describe a single concept, so it cannot be a concept related to computing. Whether a page is a disambiguation page or not is indicated at the bottom of the Wikipedia article for the concept.

   > "Software implementation" is a disambiguation page. It points to several computing-related articles, but it is not, itself, a concept related to computing.

2. If an article is a redirect page, use the article it redirects to instead. Redirect pages automatically redirect to the correct article. If an article does not exist (usually, because it has been deleted), mark it as not related to computing.

   > "Character string" is a redirect page that points to "String (computer science)", which is related to computing.

3. The context in which the concept appears matters. A computer technology is not related to computing if it is only used as a tool for a general audience.

   > The concept "Blog", in the sentence "You can find further explanations on my blog" is not related to computing, because it refers to a location where anyone can find additional resources.
   >
   > However, in the sentence "Use Wordpress to create an interactive blog", the concept "Blog" is related to computing, because it refers to a software platform to be built by programmers.

4. Rely on the content of the Wikipedia article to help distinguish computing-related concepts.

   > The Wikipedia article for "Code" describes the concept of code it in its very general sense, and it does not discuss its use in the programming domain. Therefore, it is not related to computing. Correct articles for the concept of "code" can be "Source code", "Character encoding", or "Code (cryptography)".
   >
   > Similarly, "Definition" (of a variable, function, class) represents a feature of all programming languages, but the Wikipedia article for "Definition" does not discuss its computing sense, so it is not related to computing.

5. Concepts not uniquely specific to computing, but with considerable importance in computing, should be accepted as related to computing. In this case, however, the Wikipedia article must mention the computing-specific interpretation of the concept.

   > "Implementation" is a concept that applies to many domains, including computing. Because of its importance in computing, and because the article describes its application to computer science (section 1.1), it is related to computing.

6. If in doubt, err on the side of marking the concept as **not** related to computing.

"Explicitly mentioned"

To be accepted, a concept must appear explicitly in the *text* of the Stack Overflow post. Do not consider the post's title, comments, or other related question and answers.

Special Cases:

1. Accept concepts that are mentioned by:

   - alternate phrases or synonyms (use the content of the article to identify synonyms)

     > "map" is an explicit mention of the concept "Associative Array"

   - a different part-of-speech, i.e., nouns, verbs, adjectives, etc.

     > "concatenate" is an explicit mention of the concept "Concatenation"

   - related morphological forms, including antonyms (if they are formed by adding a prefix)

     > "rebooting" is an explicit mention of the concept "Booting"

2. Reject concepts that are not mentioned in the text, even if they are related to concepts mentioned in the text (such as hypernyms, hyponyms, meronyms, and holonyms).

   > "Twitter" is not an explicit mention of the concept "Social network", because Twitter is a hyponym (special case) of social network.

3. Some Wikipedia articles discuss a general concept in a specific context. For a specific mention in a post, accept both a general article and an article with a specific context if the context matches the one of the post. Reject a context-specific article if the post does not imply any particular context.

   > With the sentence "Paypal is a payment system", both "Payment system" and "E-commerce payment system" are acceptable concepts, because "payment system" is used in the context of e-commerce, so the context-specific "E-commerce payment system" is acceptable, as well as the more generic article "Payment system".

With the sentence "I want show the first 100 rows of a table", unless it is clear that the table comes from a database, the concept "Table (database)" must be rejected, because it is described in context that is more specific than the post. An acceptable concept would be "Table (information)".

4. *Programming language syntactic constructs:* Reject a concept about a syntactic construct, unless the syntactic construct is itself mentioned.

   With the sentence "This method returns a String", the concept "Return statement" is not an acceptable concept, because the syntactic construct (a statement) is not mentioned.

   Similarly, with the sentence "The method returns if x = 0", the concept "Conditional (computer programming)" is not an acceptable concept.

5. *URL:* Accept a concept only if it matches the whole URL. In particular, reject mentions in the path of the URL.

   "https://stackoverflow.com/questions/1108/how-does-database-indexing-work/" is not a mention of "Database", even if "database" appears as a sub-string of the URL.

6. *Code:* Do not consider mentions in code blocks, but consider inline code. An identifier can be an explicit mention of a concept if the identifier refers to the concept. Block quotes are not code, and therefore must be considered as normal text. Treat clear multi-line code fragments or stack traces that happen not to be in code blocks as if they were code blocks (i.e., do not consider them).

   Concepts "Set (abstract data type)" and "String (computer science)" are mentioned in "Set<String>" (unless it is found in a code block). However, concepts such as "Interface (computing)" or "Generic programming" are not mentioned, because the identifiers do not refer to these concepts.

A.2 Sample of Wikification Annotations

Content of Annotation_Guide/sample-annotations-5posts.xlsx *from* https://zenodo.org/records/4442458

URL	Post	Concept	Match
https://stackoverflow.com/questions/41700996	P41700996		
https://en.wikipedia.org/wiki/Data type		Data type	yes
https://en.wikipedia.org/wiki/Type species		Type species	no
https://en.wikipedia.org/wiki/Conclusion (music)		Conclusion (music)	no

URL	Post	Concept	Match
https://en.wikipedia.org/wiki/.properties		.properties	yes
https://en.wikipedia.org/wiki/Homonym		Homonym	no
https://en.wikipedia.org/wiki/At the End		At the End	no
https://en.wikipedia.org/wiki/Variable star		Variable star	no
https://en.wikipedia.org/wiki/String literal		String literal	no
https://en.wikipedia.org/wiki/Universal quantification		Universal quantification	no
https://en.wikipedia.org/wiki/Prince		Prince	no
https://en.wikipedia.org/wiki/Príncipe		Príncipe	no
https://en.wikipedia.org/wiki/Der Ring des Nibelungen		Der Ring des Nibelungen	no
https://en.wikipedia.org/wiki/Endomorphism		Endomorphism	no
https://en.wikipedia.org/wiki/Key (lock)		Key (lock)	no
https://en.wikipedia.org/wiki/Variable (computer science)		Variable (computer science)	yes
https://en.wikipedia.org/wiki/Associative array		Associative array	yes
https://en.wikipedia.org/wiki/Random variable		Random variable	no
https://en.wikipedia.org/wiki/Why (Annie Lennox song)		Why (Annie Lennox song)	no
https://en.wikipedia.org/wiki/English orthography		English orthography	no
https://en.wikipedia.org/wiki/Those, Nepal		Those, Nepal	no
https://en.wikipedia.org/wiki/Configure script		Configure script	no
https://en.wikipedia.org/wiki/Definition		Definition	no
https://en.wikipedia.org/wiki/String section		String section	no
https://en.wikipedia.org/wiki/Operators in C and C++		Operators in C and C++	no
https://en.wikipedia.org/wiki/Single-valued function		Single-valued function	no
https://en.wikipedia.org/wiki/Computer file		Computer file	yes
https://en.wikipedia.org/wiki/Teleological argument		Teleological argument	no
https://en.wikipedia.org/wiki/C preprocessor		C preprocessor	no
https://en.wikipedia.org/wiki/Unique key		Unique key	no
https://en.wikipedia.org/wiki/Map		Map	no
https://en.wikipedia.org/wiki/Property (philosophy)		Property (philosophy)	no
https://en.wikipedia.org/wiki/Access control		Access control	no
https://en.wikipedia.org/wiki/Free will		Free will	no
https://en.wikipedia.org/wiki/5"/38 caliber gun		5"/38 caliber gun	no
https://en.wikipedia.org/wiki/String (computer science)		String (computer science)	yes
https://en.wikipedia.org/wiki/Enharmonic		Enharmonic	no
https://en.wikipedia.org/wiki/Property (programming)		Property (programming)	no
https://en.wikipedia.org/wiki/Property		Property	no
https://en.wikipedia.org/wiki/Freedom of the City		Freedom of the City	no
https://en.wikipedia.org/wiki/Syllable		Syllable	no
https://en.wikipedia.org/wiki/That		That	no
https://en.wikipedia.org/wiki/Bracket		Bracket	no
https://en.wikipedia.org/wiki/Reference		Reference	no
https://en.wikipedia.org/wiki/Access key		Access key	no
https://en.wikipedia.org/wiki/Canning		Canning	no
https://en.wikipedia.org/wiki/End of World War II in Europe		End of World War II in Europe	no
https://en.wikipedia.org/wiki/Variable (mathematics)		Variable (mathematics)	no
https://en.wikipedia.org/wiki/Orgasm		Orgasm	no
https://en.wikipedia.org/wiki/Key (music)		Key (music)	no
https://en.wikipedia.org/wiki/Surrender of Japan		Surrender of Japan	no
https://en.wikipedia.org/wiki/5%22/38 caliber gun		5"/38 caliber gun	no
https://en.wikipedia.org/wiki/You Can		You Can	no
https://stackoverflow.com/questions/11366327	P11366327		
https://en.wikipedia.org/wiki/Harmonic		Harmonic	no
https://en.wikipedia.org/wiki/Metre (music)		Metre (music)	no
https://en.wikipedia.org/wiki/Monotheism		Monotheism	no
https://en.wikipedia.org/wiki/Income		Income	no
https://en.wikipedia.org/wiki/Music examination		Music examination	no
https://en.wikipedia.org/wiki/Small-signal model		Small-signal model	no
https://en.wikipedia.org/wiki/Human physical appearance		Human physical appearance	no
https://en.wikipedia.org/wiki/Exonym and endonym		Exonym and endonym	no
https://en.wikipedia.org/wiki/Vertex (graph theory)		Vertex (graph theory)	no
https://en.wikipedia.org/wiki/Image		Image	no
https://en.wikipedia.org/wiki/Universal quantification		Universal quantification	no
https://en.wikipedia.org/wiki/English modal verbs		English modal verbs	no
https://en.wikipedia.org/wiki/Graph of a function		Graph of a function	no
https://en.wikipedia.org/wiki/Sphere		Sphere	no
https://en.wikipedia.org/wiki/Dacians		Dacians	no
https://en.wikipedia.org/wiki/Java (programming language)		Java (programming language)	yes
https://en.wikipedia.org/wiki/A- and B-class destroyer		A- and B-class destroyer	no
https://en.wikipedia.org/wiki/Digital signature		Digital signature	no
https://en.wikipedia.org/wiki/Data structure		Data structure	no

URL	Post	Concept	Match
https://en.wikipedia.org/wiki/Child		Child	no
https://en.wikipedia.org/wiki/Protocol (object-oriented programming)		Protocol (object-oriented programming)	no
https://en.wikipedia.org/wiki/CPU cache		CPU cache	no
https://en.wikipedia.org/wiki/Amplitude		Amplitude	no
https://en.wikipedia.org/wiki/Object (computer science)		Object (computer science)	no
https://en.wikipedia.org/wiki/English orthography		English orthography	no
https://en.wikipedia.org/wiki/Recurrence relation		Recurrence relation	no
https://en.wikipedia.org/wiki/General relativity		General relativity	no
https://en.wikipedia.org/wiki/Social structure		Social structure	no
https://en.wikipedia.org/wiki/Bitwise operation		Bitwise operation	no
https://en.wikipedia.org/wiki/Level of measurement		Level of measurement	no
https://en.wikipedia.org/wiki/Soviet occupation zone		Soviet occupation zone	no
https://en.wikipedia.org/wiki/Multiple dispatch		Multiple dispatch	no
https://en.wikipedia.org/wiki/Theory in Practice		Theory in Practice	no
https://en.wikipedia.org/wiki/Weird fiction		Weird fiction	no
https://en.wikipedia.org/wiki/Single parent		Single parent	no
https://en.wikipedia.org/wiki/Mundos opuestos (Chilean TV series)		Mundos opuestos (Chilean TV series)	no
https://en.wikipedia.org/wiki/Multiverse		Multiverse	no
https://en.wikipedia.org/wiki/Interface (computing)		Interface (computing)	yes
https://en.wikipedia.org/wiki/Ancestor		Ancestor	no
https://en.wikipedia.org/wiki/Social change		Social change	no
https://en.wikipedia.org/wiki/Social influence		Social influence	no
https://en.wikipedia.org/wiki/Theory		Theory	no
https://en.wikipedia.org/wiki/Willie Mays		Willie Mays	no
https://en.wikipedia.org/wiki/Multiple inheritance		Multiple inheritance	yes
https://en.wikipedia.org/wiki/Biology		Biology	no
https://en.wikipedia.org/wiki/Maize		Maize	no
https://en.wikipedia.org/wiki/Structural level		Structural level	no
https://en.wikipedia.org/wiki/Suicide		Suicide	no
https://en.wikipedia.org/wiki/Implementation		Implementation	yes
https://en.wikipedia.org/wiki/Visual perception		Visual perception	no
https://en.wikipedia.org/wiki/Enharmonic		Enharmonic	no
https://en.wikipedia.org/wiki/The		The	no
https://en.wikipedia.org/wiki/Interface (Java)		Interface (Java)	yes
https://en.wikipedia.org/wiki/Interface (matter)		Interface (matter)	no
https://en.wikipedia.org/wiki/R. Kelly		R. Kelly	no
https://en.wikipedia.org/wiki/Id, ego and super-ego		Id, ego and super-ego	no
https://en.wikipedia.org/wiki/Android (operating system)		Android (operating system)	no
https://en.wikipedia.org/wiki/Anatomical terms of motion		Anatomical terms of motion	no
https://en.wikipedia.org/wiki/Racial segregation		Racial segregation	no
https://en.wikipedia.org/wiki/B (programming language)		B (programming language)	no
https://en.wikipedia.org/wiki/Bacteria		Bacteria	no
https://en.wikipedia.org/wiki/Structure		Structure	no
https://en.wikipedia.org/wiki/Professional wrestling match types		Professional wrestling match types	no
https://en.wikipedia.org/wiki/Tree (data structure)		Tree (data structure)	yes
https://en.wikipedia.org/wiki/Military		Military	no
https://en.wikipedia.org/wiki/For loop		For loop	no
https://en.wikipedia.org/wiki/Might (magazine)		Might (magazine)	no
https://en.wikipedia.org/wiki/Human nature		Human nature	no
https://en.wikipedia.org/wiki/Computer programming		Computer programming	no
https://en.wikipedia.org/wiki/Puberty		Puberty	no
https://en.wikipedia.org/wiki/V/Line A class		V/Line A class	no
https://en.wikipedia.org/wiki/A-class destroyer (1913)		A-class destroyer (1913)	no
https://en.wikipedia.org/wiki/Terrorism		Terrorism	no
https://en.wikipedia.org/wiki/Evolutionary grade		Evolutionary grade	no
https://en.wikipedia.org/wiki/Never (Heart song)		Never (Heart song)	no
https://en.wikipedia.org/wiki/Religion		Religion	no
https://en.wikipedia.org/wiki/United Kingdom		United Kingdom	no
https://en.wikipedia.org/wiki/Node (computer science)		Node (computer science)	yes
https://en.wikipedia.org/wiki/Experience point		Experience point	no
https://en.wikipedia.org/wiki/Information technology		Information technology	no
https://en.wikipedia.org/wiki/Instance (computer science)		Instance (computer science)	yes
https://en.wikipedia.org/wiki/Class (computer programming)		Class (computer programming)	yes
https://en.wikipedia.org/wiki/Medicine		Medicine	no
https://en.wikipedia.org/wiki/Canadian Hot 100		Canadian Hot 100	no
https://en.wikipedia.org/wiki/Time management		Time management	no
https://en.wikipedia.org/wiki/Breast		Breast	no
https://en.wikipedia.org/wiki/Family		Family	no
https://en.wikipedia.org/wiki/Therefore sign		Therefore sign	no

URL	Post	Concept	Match
https://en.wikipedia.org/wiki/U-boat Campaign (World War I)		U-boat Campaign (World War I)	no
https://en.wikipedia.org/wiki/Software feature		Software feature	no
https://en.wikipedia.org/wiki/Actually		Actually	no
https://en.wikipedia.org/wiki/ActionScript		ActionScript	no
https://en.wikipedia.org/wiki/Semantic memory		Semantic memory	no
https://en.wikipedia.org/wiki/Flood fill		Flood fill	no
https://en.wikipedia.org/wiki/Standardization		Standardization	no
https://en.wikipedia.org/wiki/Keep Austin Weird		Keep Austin Weird	no
https://en.wikipedia.org/wiki/My Neighbors the Yamadas		My Neighbors the Yamadas	no
https://en.wikipedia.org/wiki/Hypothesis		Hypothesis	no
https://en.wikipedia.org/wiki/Hierarchy		Hierarchy	no
https://en.wikipedia.org/wiki/Conflict (process)		Conflict (process)	no
https://en.wikipedia.org/wiki/Multiplicity (mathematics)		Multiplicity (mathematics)	no
https://en.wikipedia.org/wiki/Theory (mathematical logic)		Theory (mathematical logic)	no
https://en.wikipedia.org/wiki/27th government of Turkey		27th government of Turkey	no
https://en.wikipedia.org/wiki/Tree		Tree	no
https://en.wikipedia.org/wiki/Instance dungeon		Instance dungeon	no
https://en.wikipedia.org/wiki/Confirmation bias		Confirmation bias	no
https://en.wikipedia.org/wiki/Aishō, Shiga		Aishō, Shiga	no
https://en.wikipedia.org/wiki/Semiconductor device fabrication		Semiconductor device fabrication	no
https://en.wikipedia.org/wiki/This Is... (TV series)		This Is... (TV series)	no
https://en.wikipedia.org/wiki/Those, Nepal		Those, Nepal	no
https://en.wikipedia.org/wiki/Assistive technology		Assistive technology	no
https://en.wikipedia.org/wiki/Phylogenetic tree		Phylogenetic tree	no
https://en.wikipedia.org/wiki/Level (video gaming)		Level (video gaming)	no
https://en.wikipedia.org/wiki/Proto-Norse language		Proto-Norse language	no
https://en.wikipedia.org/wiki/What You're On		What You're On	no
https://en.wikipedia.org/wiki/Honda Life		Honda Life	no
https://en.wikipedia.org/wiki/This Is Not		This Is Not	no
https://en.wikipedia.org/wiki/Tree structure		Tree structure	no
https://en.wikipedia.org/wiki/Extensibility		Extensibility	no
https://en.wikipedia.org/wiki/Tetrahedral molecular geometry		Tetrahedral molecular geometry	no
https://en.wikipedia.org/wiki/Object-oriented programming		Object-oriented programming	no
https://en.wikipedia.org/wiki/Function (engineering)		Function (engineering)	no
https://en.wikipedia.org/wiki/Psalms		Psalms	no
https://en.wikipedia.org/wiki/Inheritance (object-oriented programming)		Inheritance (object-oriented programming)	yes
https://en.wikipedia.org/wiki/Application programming interface		Application programming interface	no
https://en.wikipedia.org/wiki/Programming language		Programming language	yes
https://en.wikipedia.org/wiki/Homeomorphism		Homeomorphism	no
https://en.wikipedia.org/wiki/Causality		Causality	no
https://en.wikipedia.org/wiki/Is-a		Is-a	no
https://en.wikipedia.org/wiki/Potentiality and actuality		Potentiality and actuality	no
https://en.wikipedia.org/wiki/Social support		Social support	no
https://en.wikipedia.org/wiki/Bootstrapping (compilers)		Bootstrapping (compilers)	no
https://en.wikipedia.org/wiki/Hereditarily countable set		Hereditarily countable set	no
https://en.wikipedia.org/wiki/Parent		Parent	no
https://en.wikipedia.org/wiki/Inheritance (computer science)		Inheritance (computer science)	yes
https://en.wikipedia.org/wiki/Unix-like		Unix-like	no
https://en.wikipedia.org/wiki/Reciprocal lattice		Reciprocal lattice	no
https://en.wikipedia.org/wiki/Schizophrenia		Schizophrenia	no
https://en.wikipedia.org/wiki/Affect (psychology)		Affect (psychology)	no
https://en.wikipedia.org/wiki/List of nuclides		List of nuclides	no
https://en.wikipedia.org/wiki/Family tree		Family tree	no
https://en.wikipedia.org/wiki/Mathematics		Mathematics	no
https://en.wikipedia.org/wiki/Stellar classification		Stellar classification	no
https://en.wikipedia.org/wiki/Domain Name System		Domain Name System	no
https://en.wikipedia.org/wiki/Hot R&B/Hip-Hop Songs		Hot R&B/Hip-Hop Songs	no
https://en.wikipedia.org/wiki/Neon		Neon	no
https://en.wikipedia.org/wiki/Budha		Budha	no
https://en.wikipedia.org/wiki/Language		Language	no
https://en.wikipedia.org/wiki/United States Constitution		United States Constitution	no
https://en.wikipedia.org/wiki/Graph (discrete mathematics)		Graph (discrete mathematics)	no
https://en.wikipedia.org/wiki/Dialect		Dialect	no
https://en.wikipedia.org/wiki/Technical standard		Technical standard	no
https://en.wikipedia.org/wiki/Hungarian Academy of Sciences		Hungarian Academy of Sciences	no
https://en.wikipedia.org/wiki/Graph (mathematics)		Graph (mathematics)	no
https://en.wikipedia.org/wiki/There (virtual world)		There (virtual world)	no
https://en.wikipedia.org/wiki/Sexual differentiation		Sexual differentiation	no

URL	Post	Concept	Match
https://en.wikipedia.org/wiki/Functional group		Functional group	no
https://en.wikipedia.org/wiki/Structure (mathematical logic)		Structure (mathematical logic)	no
https://en.wikipedia.org/wiki/That		That	no
https://en.wikipedia.org/wiki/Occam's razor		Occam's razor	no
https://en.wikipedia.org/wiki/Every Child (film)		Every Child (film)	no
https://en.wikipedia.org/wiki/Practice (learning method)		Practice (learning method)	no
https://en.wikipedia.org/wiki/Uniqueness quantification		Uniqueness quantification	no
https://en.wikipedia.org/wiki/Royal Oak		Royal Oak	no
https://en.wikipedia.org/wiki/Sathya Sai Baba		Sathya Sai Baba	no
https://en.wikipedia.org/wiki/As One Black Eagles		As One Black Eagles	no
https://en.wikipedia.org/wiki/Graph theory		Graph theory	no
https://en.wikipedia.org/wiki/Vaccine		Vaccine	no
https://en.wikipedia.org/wiki/WAGR K class		WAGR K class	no
https://en.wikipedia.org/wiki/Inventive step and non-obviousness		Inventive step and non-obviousness	no
https://en.wikipedia.org/wiki/San Fernando, Trinidad and Tobago		San Fernando, Trinidad and Tobago	no
https://en.wikipedia.org/wiki/Identity (philosophy)		Identity (philosophy)	no
https://en.wikipedia.org/wiki/Abstraction layer		Abstraction layer	no
https://en.wikipedia.org/wiki/Subtraction		Subtraction	no
https://en.wikipedia.org/wiki/Screw		Screw	no
https://en.wikipedia.org/wiki/Binary search tree		Binary search tree	no
https://en.wikipedia.org/wiki/Imagination		Imagination	no
https://en.wikipedia.org/wiki/But/Aishō		But/Aishō	no
https://en.wikipedia.org/wiki/Logical conjunction		Logical conjunction	no
https://en.wikipedia.org/wiki/Reflexive pronoun		Reflexive pronoun	no
https://en.wikipedia.org/wiki/Java virtual machine		Java virtual machine	no
https://en.wikipedia.org/wiki/Grandparent		Grandparent	no
https://en.wikipedia.org/wiki/You		You	no
https://stackoverflow.com/questions/55228245	P55228245		
https://en.wikipedia.org/wiki/Gratitude		Gratitude	no
https://en.wikipedia.org/wiki/Magnetohydrodynamics		Magnetohydrodynamics	no
https://en.wikipedia.org/wiki/Import		Import	no
https://en.wikipedia.org/wiki/Parkinson's disease		Parkinson's disease	no
https://en.wikipedia.org/wiki/C (programming language)		C (programming language)	no
https://en.wikipedia.org/wiki/R10 (Rodalies de Catalunya)		R10 (Rodalies de Catalunya)	no
https://en.wikipedia.org/wiki/Safe		Safe	no
https://en.wikipedia.org/wiki/HTML		HTML	no
https://en.wikipedia.org/wiki/This (computer programming)		This (computer programming)	no
https://en.wikipedia.org/wiki/Last (unit)		Last (unit)	no
https://en.wikipedia.org/wiki/Software bug		Software bug	yes
https://en.wikipedia.org/wiki/Library (computing)		Library (computing)	no
https://en.wikipedia.org/wiki/Universal quantification		Universal quantification	no
https://en.wikipedia.org/wiki/NumPy		NumPy	no
https://en.wikipedia.org/wiki/Py (cipher)		Py (cipher)	no
https://en.wikipedia.org/wiki/Init		Init	no
https://en.wikipedia.org/wiki/Paris Métro Line 3		Paris Métro Line 3	no
https://en.wikipedia.org/wiki/Last.fm		Last.fm	no
https://en.wikipedia.org/wiki/Package manager		Package manager	no
https://en.wikipedia.org/wiki/Giant panda		Giant panda	no
https://en.wikipedia.org/wiki/Taxation in Canada		Taxation in Canada	no
https://en.wikipedia.org/wiki/Image scaling		Image scaling	no
https://en.wikipedia.org/wiki/Seoul Subway Line 3		Seoul Subway Line 3	no
https://en.wikipedia.org/wiki/Equals sign		Equals sign	no
https://en.wikipedia.org/wiki/I Was Warned		I Was Warned	no
https://en.wikipedia.org/wiki/Translation		Translation	no
https://en.wikipedia.org/wiki/Trial		Trial	no
https://en.wikipedia.org/wiki/Barcelona Metro line 3		Barcelona Metro line 3	no
https://en.wikipedia.org/wiki/Tariff		Tariff	no
https://en.wikipedia.org/wiki/Palladium		Palladium	no
https://en.wikipedia.org/wiki/ANSI escape code		ANSI escape code	no
https://en.wikipedia.org/wiki/Meaning (linguistics)		Meaning (linguistics)	no
https://en.wikipedia.org/wiki/End user		End user	no
https://en.wikipedia.org/wiki/Dimension (vector space)		Dimension (vector space)	no
https://en.wikipedia.org/wiki/Computer file		Computer file	yes
https://en.wikipedia.org/wiki/Line 23 (Shanghai Metro)		Line 23 (Shanghai Metro)	no
https://en.wikipedia.org/wiki/Pinyin		Pinyin	no
https://en.wikipedia.org/wiki/.py		.py	no
https://en.wikipedia.org/wiki/Humanitarian aid		Humanitarian aid	no
https://en.wikipedia.org/wiki/ImageMagick		ImageMagick	no
https://en.wikipedia.org/wiki/Line 3 (Mumbai Metro)		Line 3 (Mumbai Metro)	no

URL	Post	Concept	Match
https://en.wikipedia.org/wiki/Holocene		Holocene	no
https://en.wikipedia.org/wiki/Lp space		Lp space	no
https://en.wikipedia.org/wiki/Madlib		Madlib	no
https://en.wikipedia.org/wiki/Northern Pacific Railway		Northern Pacific Railway	no
https://en.wikipedia.org/wiki/Paris Métro Line 10		Paris Métro Line 10	no
https://en.wikipedia.org/wiki/Widescreen signaling		Widescreen signaling	no
https://en.wikipedia.org/wiki/Comma-separated values		Comma-separated values	yes
https://en.wikipedia.org/wiki/Free will		Free will	no
https://en.wikipedia.org/wiki/Subroutine		Subroutine	no
https://en.wikipedia.org/wiki/Website		Website	no
https://en.wikipedia.org/wiki/Backslash		Backslash	no
https://en.wikipedia.org/wiki/Suicide		Suicide	no
https://en.wikipedia.org/wiki/Env		Env	no
https://en.wikipedia.org/wiki/Currency appreciation and depreciation		Currency appreciation and depreciation	no
https://en.wikipedia.org/wiki/Last Glacial Maximum		Last Glacial Maximum	no
https://en.wikipedia.org/wiki/Stack trace		Stack trace	no
https://en.wikipedia.org/wiki/Python (programming language)		Python (programming language)	no
https://en.wikipedia.org/wiki/Pandas (software)		Pandas (software)	no
https://en.wikipedia.org/wiki/Attribute (computing)		Attribute (computing)	no
https://en.wikipedia.org/wiki/Hed PE		Hed PE	no
https://en.wikipedia.org/wiki/Modular programming		Modular programming	no
https://en.wikipedia.org/wiki/Online help		Online help	no
https://en.wikipedia.org/wiki/User (computing)		User (computing)	no
https://en.wikipedia.org/wiki/Unitary patent		Unitary patent	no
https://en.wikipedia.org/wiki/Line (geometry)		Line (geometry)	no
https://en.wikipedia.org/wiki/Asterisk		Asterisk	no
https://en.wikipedia.org/wiki/Pure Data		Pure Data	no
https://en.wikipedia.org/wiki/Software versioning		Software versioning	no
https://en.wikipedia.org/wiki/Bracket		Bracket	no
https://en.wikipedia.org/wiki/Drive letter assignment		Drive letter assignment	no
https://en.wikipedia.org/wiki/Religion		Religion	no
https://en.wikipedia.org/wiki/Error		Error	no
https://en.wikipedia.org/wiki/Rotation		Rotation	no
https://en.wikipedia.org/wiki/Libretto		Libretto	no
https://en.wikipedia.org/wiki/File system		File system	no
https://en.wikipedia.org/wiki/XML		XML	yes
https://en.wikipedia.org/wiki/Telephone call		Telephone call	no
https://en.wikipedia.org/wiki/WILL		WILL	no
https://en.wikipedia.org/wiki/Pythonidae		Pythonidae	no
https://en.wikipedia.org/wiki/W-inds discography		W-inds discography	no
https://en.wikipedia.org/wiki/ActionScript		ActionScript	no
https://en.wikipedia.org/wiki/20th Empire Awards		20th Empire Awards	no
https://stackoverflow.com/questions/22969451	P22969451		
https://en.wikipedia.org/wiki/Gratitude		Gratitude	no
https://en.wikipedia.org/wiki/Reference type		Reference type	no
https://en.wikipedia.org/wiki/Data type		Data type	yes
https://en.wikipedia.org/wiki/Metre		Metre	no
https://en.wikipedia.org/wiki/Web service		Web service	yes
https://en.wikipedia.org/wiki/Booting		Booting	no
https://en.wikipedia.org/wiki/This (computer programming)		This (computer programming)	no
https://en.wikipedia.org/wiki/Software bug		Software bug	yes
https://en.wikipedia.org/wiki/Namespace (computer science)		Namespace (computer science)	yes
https://en.wikipedia.org/wiki/MPEG-4 Part 2		MPEG-4 Part 2	no
https://en.wikipedia.org/wiki/When (Amanda Lear song)		When (Amanda Lear song)	no
https://en.wikipedia.org/wiki/Identifier		Identifier	no
https://en.wikipedia.org/wiki/Middle English		Middle English	no
https://en.wikipedia.org/wiki/Command (computing)		Command (computing)	yes
https://en.wikipedia.org/wiki/Windows ME		Windows ME	no
https://en.wikipedia.org/wiki/Missing in action		Missing in action	no
https://en.wikipedia.org/wiki/Thank You (The Walking Dead)		Thank You (The Walking Dead)	no
https://en.wikipedia.org/wiki/Existence		Existence	no
https://en.wikipedia.org/wiki/English orthography		English orthography	no
https://en.wikipedia.org/wiki/Compiler		Compiler	no
https://en.wikipedia.org/wiki/Thank You in Advance		Thank You in Advance	no
https://en.wikipedia.org/wiki/Addition		Addition	no
https://en.wikipedia.org/wiki/Trial		Trial	no
https://en.wikipedia.org/wiki/Atheism		Atheism	no
https://en.wikipedia.org/wiki/Assembly language		Assembly language	no
https://en.wikipedia.org/wiki/Missing person		Missing person	no

URL	Post	Concept	Match
https://en.wikipedia.org/wiki/Scheme (linguistics)		Scheme (linguistics)	no
https://en.wikipedia.org/wiki/Recursion		Recursion	no
https://en.wikipedia.org/wiki/Deer		Deer	no
https://en.wikipedia.org/wiki/Namespace		Namespace	yes
https://en.wikipedia.org/wiki/Advance payment		Advance payment	no
https://en.wikipedia.org/wiki/Tab key		Tab key	no
https://en.wikipedia.org/wiki/RSS		RSS	no
https://en.wikipedia.org/wiki/Toponymy		Toponymy	no
https://en.wikipedia.org/wiki/Runway		Runway	no
https://en.wikipedia.org/wiki/Execution (computing)		Execution (computing)	yes
https://en.wikipedia.org/wiki/Trench warfare		Trench warfare	no
https://en.wikipedia.org/wiki/South Sudanese Civil War		South Sudanese Civil War	no
https://en.wikipedia.org/wiki/Complementary event		Complementary event	no
https://en.wikipedia.org/wiki/Start-stop system		Start-stop system	no
https://en.wikipedia.org/wiki/Offensive (military)		Offensive (military)	no
https://en.wikipedia.org/wiki/X-class submarine		X-class submarine	no
https://en.wikipedia.org/wiki/Host (network)		Host (network)	no
https://en.wikipedia.org/wiki/Daemon (computing)		Daemon (computing)	yes
https://en.wikipedia.org/wiki/Evaluation strategy		Evaluation strategy	no
https://en.wikipedia.org/wiki/Et cetera		Et cetera	no
https://en.wikipedia.org/wiki/Saudi Arabia at the 2010 Summer Youth Olympics		Saudi Arabia at the 2010 Summer Youth Olympics	no
https://en.wikipedia.org/wiki/National Assembly (South Korea)		National Assembly (South Korea)	no
https://en.wikipedia.org/wiki/World Wide Web		World Wide Web	yes
https://en.wikipedia.org/wiki/Error (baseball)		Error (baseball)	no
https://en.wikipedia.org/wiki/System		System	yes
https://en.wikipedia.org/wiki/ETC (TV channel)		ETC (TV channel)	no
https://en.wikipedia.org/wiki/Factorial		Factorial	no
https://en.wikipedia.org/wiki/Classic Mac OS		Classic Mac OS	no
https://en.wikipedia.org/wiki/The		The	no
https://en.wikipedia.org/wiki/Popular assembly		Popular assembly	no
https://en.wikipedia.org/wiki/Address space		Address space	no
https://en.wikipedia.org/wiki/Pointer (computer programming)		Pointer (computer programming)	no
https://en.wikipedia.org/wiki/Windows service		Windows service	yes
https://en.wikipedia.org/wiki/Computer graphics		Computer graphics	no
https://en.wikipedia.org/wiki/Type system		Type system	no
https://en.wikipedia.org/wiki/Operating system		Operating system	no
https://en.wikipedia.org/wiki/Reference (computer science)		Reference (computer science)	yes
https://en.wikipedia.org/wiki/Are You In%3F		Are You In?	no
https://en.wikipedia.org/wiki/Error		Error	no
https://en.wikipedia.org/wiki/Disney+		Disney+	no
https://en.wikipedia.org/wiki/Information technology		Information technology	no
https://en.wikipedia.org/wiki/Microsoft Windows		Microsoft Windows	yes
https://en.wikipedia.org/wiki/I Am (2010 Indian film)		I Am (2010 Indian film)	no
https://en.wikipedia.org/wiki/I am (biblical term)		I am (biblical term)	no
https://en.wikipedia.org/wiki/Command-line interface		Command-line interface	no
https://en.wikipedia.org/wiki/Organ stop		Organ stop	no
https://en.wikipedia.org/wiki/Preprocessor		Preprocessor	no
https://en.wikipedia.org/wiki/Royal Navy		Royal Navy	no
https://stackoverflow.com/questions/23088709	P23088709		
https://en.wikipedia.org/wiki/Polymorphism (biology)		Polymorphism (biology)	no
https://en.wikipedia.org/wiki/Project commissioning		Project commissioning	no
https://en.wikipedia.org/wiki/Term limit		Term limit	no
https://en.wikipedia.org/wiki/Word		Word	no
https://en.wikipedia.org/wiki/Information system		Information system	no
https://en.wikipedia.org/wiki/Potsdam		Potsdam	no
https://en.wikipedia.org/wiki/Exonym and endonym		Exonym and endonym	no
https://en.wikipedia.org/wiki/Image		Image	no
https://en.wikipedia.org/wiki/Universal quantification		Universal quantification	no
https://en.wikipedia.org/wiki/English modal verbs		English modal verbs	no
https://en.wikipedia.org/wiki/Genetic code		Genetic code	no
https://en.wikipedia.org/wiki/Object lifetime		Object lifetime	no
https://en.wikipedia.org/wiki/Limited liability company		Limited liability company	no
https://en.wikipedia.org/wiki/Complement (set theory)		Complement (set theory)	no
https://en.wikipedia.org/wiki/Scientific method		Scientific method	no
https://en.wikipedia.org/wiki/Middle English		Middle English	no
https://en.wikipedia.org/wiki/Political criticism		Political criticism	no
https://en.wikipedia.org/wiki/Determine		Determine	no
https://en.wikipedia.org/wiki/Fly Me to the Moon		Fly Me to the Moon	no

URL	Post	Concept	Match
https://en.wikipedia.org/wiki/Wikimedia Foundation		Wikimedia Foundation	no
https://en.wikipedia.org/wiki/Thank You (The Walking Dead)		Thank You (The Walking Dead)	no
https://en.wikipedia.org/wiki/Strike action		Strike action	no
https://en.wikipedia.org/wiki/Conceptual model		Conceptual model	no
https://en.wikipedia.org/wiki/Who is a Jew?		Who is a Jew?	no
https://en.wikipedia.org/wiki/English orthography		English orthography	no
https://en.wikipedia.org/wiki/Object (computer science)		Object (computer science)	yes
https://en.wikipedia.org/wiki/Bookend		Bookend	no
https://en.wikipedia.org/wiki/Berthold Carl Seemann		Berthold Carl Seemann	no
https://en.wikipedia.org/wiki/Computer simulation		Computer simulation	no
https://en.wikipedia.org/wiki/John Doe		John Doe	no
https://en.wikipedia.org/wiki/String searching algorithm		String searching algorithm	yes
https://en.wikipedia.org/wiki/Neologism		Neologism	no
https://en.wikipedia.org/wiki/Creativity		Creativity	no
https://en.wikipedia.org/wiki/Deer		Deer	no
https://en.wikipedia.org/wiki/YouTube		YouTube	no
https://en.wikipedia.org/wiki/Chemical equilibrium		Chemical equilibrium	no
https://en.wikipedia.org/wiki/Public relations		Public relations	no
https://en.wikipedia.org/wiki/Computer file		Computer file	no
https://en.wikipedia.org/wiki/Model theory		Model theory	no
https://en.wikipedia.org/wiki/Article (publishing)		Article (publishing)	no
https://en.wikipedia.org/wiki/Subject (grammar)		Subject (grammar)	no
https://en.wikipedia.org/wiki/Ambiguity		Ambiguity	no
https://en.wikipedia.org/wiki/Mystery fiction		Mystery fiction	no
https://en.wikipedia.org/wiki/Justice		Justice	no
https://en.wikipedia.org/wiki/Access control		Access control	no
https://en.wikipedia.org/wiki/Strike Me Pink (film)		Strike Me Pink (film)	no
https://en.wikipedia.org/wiki/Perspective distortion (photography)		Perspective distortion (photography)	no
https://en.wikipedia.org/wiki/Subroutine		Subroutine	yes
https://en.wikipedia.org/wiki/Comment (computer programming)		Comment (computer programming)	no
https://en.wikipedia.org/wiki/Better Way		Better Way	no
https://en.wikipedia.org/wiki/Specification (technical standard)		Specification (technical standard)	no
https://en.wikipedia.org/wiki/Set (mathematics)		Set (mathematics)	no
https://en.wikipedia.org/wiki/Website		Website	yes
https://en.wikipedia.org/wiki/Euclid		Euclid	no
https://en.wikipedia.org/wiki/French World War II destroyers		French World War II destroyers	no
https://en.wikipedia.org/wiki/Accountant		Accountant	no
https://en.wikipedia.org/wiki/I Have a Dream		I Have a Dream	no
https://en.wikipedia.org/wiki/String (computer science)		String (computer science)	yes
https://en.wikipedia.org/wiki/Web content		Web content	yes
https://en.wikipedia.org/wiki/Implementation		Implementation	yes
https://en.wikipedia.org/wiki/Being		Being	no
https://en.wikipedia.org/wiki/I Am... World Tour		I Am... World Tour	no
https://en.wikipedia.org/wiki/Where (SQL)		Where (SQL)	no
https://en.wikipedia.org/wiki/Content management system		Content management system	no
https://en.wikipedia.org/wiki/User (computing)		User (computing)	yes
https://en.wikipedia.org/wiki/Shipbuilding		Shipbuilding	no
https://en.wikipedia.org/wiki/Transformativeness		Transformativeness	no
https://en.wikipedia.org/wiki/Correlation and dependence		Correlation and dependence	no
https://en.wikipedia.org/wiki/English grammar		English grammar	no
https://en.wikipedia.org/wiki/For loop		For loop	no
https://en.wikipedia.org/wiki/Computer programming		Computer programming	no
https://en.wikipedia.org/wiki/Voluntary association		Voluntary association	no
https://en.wikipedia.org/wiki/Reference		Reference	no
https://en.wikipedia.org/wiki/Musical form		Musical form	no
https://en.wikipedia.org/wiki/Religion		Religion	no
https://en.wikipedia.org/wiki/Source code		Source code	yes
https://en.wikipedia.org/wiki/Canning		Canning	no
https://en.wikipedia.org/wiki/Information technology		Information technology	no
https://en.wikipedia.org/wiki/Limit of a sequence		Limit of a sequence	no
https://en.wikipedia.org/wiki/Applied mathematics		Applied mathematics	no
https://en.wikipedia.org/wiki/Work (physics)		Work (physics)	no
https://en.wikipedia.org/wiki/United States Department of Energy		United States Department of Energy	no
https://en.wikipedia.org/wiki/Breast		Breast	no
https://en.wikipedia.org/wiki/Writing		Writing	no
https://en.wikipedia.org/wiki/When (The Kalin Twins song)		When (The Kalin Twins song)	no
https://en.wikipedia.org/wiki/Need To		Need To	no
https://en.wikipedia.org/wiki/Instrument landing system		Instrument landing system	no
https://en.wikipedia.org/wiki/Codification (law)		Codification (law)	no

URL	Post	Concept	Match
https://en.wikipedia.org/wiki/The Pattern (The Chronicles of Amber)		The Pattern (The Chronicles of Amber)	no
https://en.wikipedia.org/wiki/Rugby union bonus points system		Rugby union bonus points system	no
https://en.wikipedia.org/wiki/Gratitude		Gratitude	no
https://en.wikipedia.org/wiki/Who is a Jew%3F		Who is a Jew?	no
https://en.wikipedia.org/wiki/Metre		Metre	no
https://en.wikipedia.org/wiki/Email		Email	no
https://en.wikipedia.org/wiki/String-searching algorithm		String-searching algorithm	yes
https://en.wikipedia.org/wiki/User-generated content		User-generated content	yes
https://en.wikipedia.org/wiki/Dear J (song)		Dear J (song)	no
https://en.wikipedia.org/wiki/Reliability engineering		Reliability engineering	no
https://en.wikipedia.org/wiki/Element (mathematics)		Element (mathematics)	no
https://en.wikipedia.org/wiki/This (computer programming)		This (computer programming)	no
https://en.wikipedia.org/wiki/Traverse (surveying)		Traverse (surveying)	no
https://en.wikipedia.org/wiki/The Set-Up (1949 film)		The Set-Up (1949 film)	no
https://en.wikipedia.org/wiki/Learning		Learning	no
https://en.wikipedia.org/wiki/Radiocarbon dating		Radiocarbon dating	no
https://en.wikipedia.org/wiki/CAN bus		CAN bus	no
https://en.wikipedia.org/wiki/Racing video game		Racing video game	no
https://en.wikipedia.org/wiki/Instance dungeon		Instance dungeon	no
https://en.wikipedia.org/wiki/Outfielder		Outfielder	no
https://en.wikipedia.org/wiki/Subtyping		Subtyping	no
https://en.wikipedia.org/wiki/Tree traversal		Tree traversal	no
https://en.wikipedia.org/wiki/Grade (climbing)		Grade (climbing)	no
https://en.wikipedia.org/wiki/Which (Unix)		Which (Unix)	no
https://en.wikipedia.org/wiki/Weasel word		Weasel word	no
https://en.wikipedia.org/wiki/Demand		Demand	no
https://en.wikipedia.org/wiki/Subject (philosophy)		Subject (philosophy)	no
https://en.wikipedia.org/wiki/Ares		Ares	no
https://en.wikipedia.org/wiki/Association (object-oriented programming)		Association (object-oriented programming)	yes
https://en.wikipedia.org/wiki/Book of Isaiah		Book of Isaiah	no
https://en.wikipedia.org/wiki/Fallacy of quoting out of context		Fallacy of quoting out of context	no
https://en.wikipedia.org/wiki/Microcontroller		Microcontroller	no
https://en.wikipedia.org/wiki/Not Yet (Monotonix album)		Not Yet (Monotonix album)	no
https://en.wikipedia.org/wiki/Almost surely		Almost surely	no
https://en.wikipedia.org/wiki/Object-oriented programming		Object-oriented programming	no
https://en.wikipedia.org/wiki/Frameup		Frameup	no
https://en.wikipedia.org/wiki/Chemistry		Chemistry	no
https://en.wikipedia.org/wiki/Context (language use)		Context (language use)	no
https://en.wikipedia.org/wiki/Economic model		Economic model	no
https://en.wikipedia.org/wiki/Metric prefix		Metric prefix	no
https://en.wikipedia.org/wiki/Programming language		Programming language	no
https://en.wikipedia.org/wiki/Dude Ranch (album)		Dude Ranch (album)	no
https://en.wikipedia.org/wiki/Modus ponens		Modus ponens	no
https://en.wikipedia.org/wiki/Big Bang		Big Bang	no
https://en.wikipedia.org/wiki/Is-a		Is-a	no
https://en.wikipedia.org/wiki/Most (Most District)		Most (Most District)	no
https://en.wikipedia.org/wiki/Representation of the People Act 1918		Representation of the People Act 1918	no
https://en.wikipedia.org/wiki/Complementary event		Complementary event	no
https://en.wikipedia.org/wiki/Roseanne Barr		Roseanne Barr	no
https://en.wikipedia.org/wiki/Determinism		Determinism	no
https://en.wikipedia.org/wiki/Mode (literature)		Mode (literature)	no
https://en.wikipedia.org/wiki/Bridge of Independent Lists		Bridge of Independent Lists	no
https://en.wikipedia.org/wiki/Science		Science	no
https://en.wikipedia.org/wiki/It Still Moves		It Still Moves	no
https://en.wikipedia.org/wiki/Article (grammar)		Article (grammar)	no
https://en.wikipedia.org/wiki/Wikipedia		Wikipedia	no
https://en.wikipedia.org/wiki/Time		Time	no
https://en.wikipedia.org/wiki/Information		Information	no
https://en.wikipedia.org/wiki/Diffraction-limited system		Diffraction-limited system	no
https://en.wikipedia.org/wiki/Factorial		Factorial	no
https://en.wikipedia.org/wiki/User interface		User interface	no
https://en.wikipedia.org/wiki/Employment		Employment	no
https://en.wikipedia.org/wiki/United States Constitution		United States Constitution	no
https://en.wikipedia.org/wiki/Ketuvim		Ketuvim	no
https://en.wikipedia.org/wiki/About URI scheme		About URI scheme	no
https://en.wikipedia.org/wiki/Mathematical model		Mathematical model	no
https://en.wikipedia.org/wiki/Method (computer programming)		Method (computer programming)	yes
https://en.wikipedia.org/wiki/Electric current		Electric current	no

URL	Post	Concept	Match
https://en.wikipedia.org/wiki/Pseudonym		Pseudonym	no
https://en.wikipedia.org/wiki/Square root		Square root	no
https://en.wikipedia.org/wiki/Copula (linguistics)		Copula (linguistics)	no
https://en.wikipedia.org/wiki/Bitcoin		Bitcoin	no
https://en.wikipedia.org/wiki/Artin transfer (group theory)		Artin transfer (group theory)	no
https://en.wikipedia.org/wiki/Linguistic prescription		Linguistic prescription	no
https://en.wikipedia.org/wiki/Napoleonic Wars		Napoleonic Wars	no
https://en.wikipedia.org/wiki/Collectivization in the Soviet Union		Collectivization in the Soviet Union	no
https://en.wikipedia.org/wiki/Camino de Santiago		Camino de Santiago	no
https://en.wikipedia.org/wiki/Graph traversal		Graph traversal	no
https://en.wikipedia.org/wiki/Not Yet (band)		Not Yet (band)	no
https://en.wikipedia.org/wiki/Software design pattern		Software design pattern	no
https://en.wikipedia.org/wiki/HTML element		HTML element	no
https://en.wikipedia.org/wiki/Allusion		Allusion	no
https://en.wikipedia.org/wiki/I am (biblical term)		I am (biblical term)	no
https://en.wikipedia.org/wiki/Myspace		Myspace	no
https://en.wikipedia.org/wiki/Set theory		Set theory	no
https://en.wikipedia.org/wiki/Control theory		Control theory	no
https://en.wikipedia.org/wiki/Material conditional		Material conditional	no
https://en.wikipedia.org/wiki/Personal pronoun		Personal pronoun	no
https://en.wikipedia.org/wiki/Right to work		Right to work	no
https://en.wikipedia.org/wiki/Instrument approach		Instrument approach	no
https://en.wikipedia.org/wiki/War in Darfur		War in Darfur	no
https://en.wikipedia.org/wiki/Pattern matching		Pattern matching	yes
https://en.wikipedia.org/wiki/Polymorphism (computer science)		Polymorphism (computer science)	yes

A.3 Scode Relatedness Annotation Guide

Content of Concept_Relatedness/coding-guide.md *from* https://zenodo.org/records/7835197
Use Java 15's source code and documentation to understand the class

- Source code: https://github.com/openjdk/jdk15 (search by pressing T on the GitHub repo, then using the query src/[name of class], the right result is typically the first, and has a path similar to src/[modulename]/share/classes/[package/name]/[class].java)
- Reference documentation: https://docs.oracle.com/en/java/javase/15/docs/api/index.html

After carefully reading the source code and reference documentation of the class, code each association with one of the following categories, in order of precedence

- [**general**] **General programming or computer science concept:** Is the concept inherently tied to programming (with modern languages) and related to almost any programming project?
- [**yes**] **Related concept:** Is the concept directly related to the implementation of the class, its usage, or the abstraction it represents?
- [**no**] **Unrelated concept:** None of the above. The concept is completely unrelated to the class.
- [**?**] **Don't know:** If unable to make a decision, use this category, but also indicate which categories are considered.

"General" Category:

- Marking a concept as "general" means that it would be related to almost all classes. This includes concepts like *object*, *class*, *method*, *function*, *branching statement*, etc., that are highly coupled with the language.
- Categories of concepts that are "general" include: programming languages, markup languages (e.g., *html*, *json*), development tools (e.g., *eclipse*). However, specialized concepts within these categories (e.g., a development tool used only in specific contexts) are not general.

Additional Rules / Special Cases

- Everyday or non-programming concepts can be "related" or "unrelated", depending on whether they relate to the abstraction of the class (e.g., *east* is related to the BorderLayout, because it uses this concept to arrange its elements). They can't, however, be "general".
  - **Rationale:** Classes can relate to non-programming concepts. However, the general category is specific to programming and computer science concepts.
- Non-concepts are "unrelated".
  - **Rationale:** They are false positives.
- If a file contains multiple classes, include all classes when deciding if the concept is related or not (i.e., it is "related" if related to at least one of the classes).
  - **Rationale:** Sampling was done at the file level.
- Don't consider the boilerplate license text at the top of files when deciding if a concept is related or not.
  - **Rationale:** Licensing is an orthogonal concern. The boilerplate license was removed from the input to the tools, so license-related concepts should be related to the class.
- Some concepts appear twice for the same class (usually the 2nd time with a trailing space). Mark both with the same category.
  - **Rationale:** it's a small bug in the generation, with no impact on the list of results other than the duplicates.
- Each term is a concept, whether it refers to abstract concepts, softwares, technical standards, organizations, etc., and the "type" of concept (abstract, software, etc.) doesn't prevent coding an association with any category.

  - **Rationale:** the term "concept" is used in a loose sense.

- Not all concepts are from Wikipedia. Code the concept, not the Wikipedia article. Use online resources, including Wikipedia, to understand the concept if necessary.

  - **Rationale:** 1 tool doesn't use Wikipedia.

- If a concept has the form "comparison of [XYZ]", "list of [XYZ]", "list of [XYZ] by [ABC]", etc., code as if the concept was only "[XYZ]".

  - **Rationale:** 2 of 3 tools use Wikipedia, which has many redirects from less popular technologies to these articles, and sometimes even the main article for "XYZ" actually redirects to the list article.

- If a concept is ambiguous or has multiple meanings, assume the most closely related meaning, i.e., the one that would give the highest category.

  - **Rationale:** this most related meaning could be useful.

- If a concept is mentioned by the reference documentation or by a code identifier, it is "related", even if the mention is not in a core part of the code.

  - **Rationale:** a developer reading the code/documentation would be expected to understand this concept.

- Concepts do not need to be mentioned to be "related".

  - **Rationale:** human judgment is sufficient.

- If a concept would be "related" for a parent class or interface of the target class, it is also "related" for the target class.

  - **Rationale:** understanding a subtype requires understanding its supertypes.

- If a concept is coded as "general" at least once, it can never be coded as "related" or "unrelated".

  - **Rationale:** answering the "general" question does not require to compare the concept with the class, so the answer should not change for a different class. General concepts are related to virtually any software project, so it isn't meaningful to make the related/unrelated decision for each class.

- If unsure about "related" or "general", err on the side of "yes" and "no" respectively. Use the "?" category if it could really be more than one category.

– **Rationale:** even slightly less related concepts can be part of the understanding of a class and useful when using or developing the class. "General" concepts, however, are not as insightful.

A.4 List of Computing-Related Wikipedia Articles

First 300 articles listed in Wikipedia_Graph/nodes.tsv *from* https://zenodo.org/records/7835197. *Each line shows the page ID of the Wikipedia article in brackets, followed by the article's title.*

[586] ASCII
[775] Algorithm
[856] Apple Inc.
[1164] Artificial intelligence
[1242] Ada (programming language)
[1368] Assembly language
[1451] APL (programming language)
[1453] ALGOL
[1456] AWK
[2014] Atomic semantics
[2052] Array data structure
[2114] IBM AIX
[2230] Analysis of algorithms
[2316] Audio file format
[2323] Amdahl's law
[2349] Abstract data type
[2581] Apache HTTP Server
[2726] Atlas Autocode
[2883] Active Server Pages
[3233] Acceptance testing
[3364] Bit
[3365] Byte
[4015] BASIC
[4052] BCPL
[4086] Brainfuck
[4266] Binary search algorithm
[4321] Binary tree
[4459] Backward compatibility
[4475] B (programming language)
[4547] Bash (Unix shell)
[4801] BeOS
[5213] Computing
[5218] Central processing unit
[5244] Cipher
[5295] Character encoding
[5300] Computer data storage
[5309] Software
[5311] Computer programming
[5323] Computer science
[5715] Cryptanalysis
[5739] Compiler
[5783] Computer program
[5926] Computation
[6021] C (programming language)
[6068] Common Lisp
[6211] Context-sensitive grammar
[6212] Context-sensitive language
[6429] Compact disc
[6513] Client-server model
[6557] Control unit
[6559] Control store
[6596] Computer vision
[6604] Rendering (computer graphics)
[6667] CPAN
[6734] Garbage collection (computer science)
[6759] Context-free grammar
[6799] COBOL
[6806] Computer memory
[6829] Cache (computing)
[6857] Computer multitasking
[7030] Code coverage
[7056] Computer mouse
[7077] Computer file
[7144] Content-control software
[7237] Common Language Infrastructure
[7262] Coral 66
[7291] CuteFTP
[7392] Class (computer programming)
[7398] Computer security
[7492] Capability Maturity Model
[7543] Computational complexity theory
[7575] CLU (programming language)
[7645] Cyclone (programming language)
[7677] Computer monitor
[7850] Chomsky normal form
[7962] Logical disjunction
[8013] Data compression
[8276] Digital data
[8339] Domain Name System
[8377] Database
[8472] Disk storage
[8495] Data set
[8501] Distributed computing
[8519] Data structure
[8525] Digital signal processing
[8640] Database normalization
[8733] Digital video
[8741] Dylan (programming language)
[8743] Document Object Model
[8904] Double-ended queue
[9101] Device driver
[9251] Engineering
[9310] Enterprise resource planning
[9499] Ethernet
[9646] Erlang (programming language)
[9647] Euphoria (programming language)
[9672] Entscheidungsproblem
[9685] Earley parser
[9738] Email
[9773] EBCDIC
[9838] Eiffel (programming language)
[9845] JavaScript
[9875] Exploit (computer security)
[10136] Expert system
[10294] Encryption
[10375] Error detection and correction
[10377] Euclidean algorithm
[10635] Free software
[10891] Floppy disk
[10931] Finite-state machine
[10933] Functional programming
[10939] Formal language
[11012] Forth (programming language)
[11168] Fortran
[11178] Foobar
[11347] FIFO (computing and electronics)
[11367] Fourth-generation programming language
[11376] Floating-point arithmetic
[11402] FileMan
[11545] Feedback
[11592] Freeware
[11691] Functional decomposition
[11856] Gnutella
[11875] GNU
[12293] Graphical user interface
[12570] Gigabyte
[12702] GIF
[12823] Garbage in, garbage out
[13191] HTML
[13259] Home page
[13263] Hexadecimal
[13443] Hypertext Transfer Protocol
[13501] Source tracking
[13777] Hard disk drive
[13790] Hash function
[13833] Hash table
[13834] "Hello, World!" program
[13995] Heapsort
[13996] Heap (data structure)
[14539] Internet
[14617] Intel
[14739] IEEE 802.11
[14773] Information theory
[14791] IEEE 802.3
[14794] Integer (computer science)
[14801] Icon (programming language)
[14921] IP address
[14934] International Organization for Standardization
[15019] ISO/IEC 8859-1
[15046] IA-32
[15072] Instruction register
[15075] INTERCAL
[15089] Interpreted language
[15144] International Electrotechnical Commission
[15145] ISO 9660
[15205] Insertion sort
[15215] Internet Explorer
[15222] IEEE 802.2
[15289] Interrupt
[15305] Integrated development environment
[15323] Internet Protocol
[15476] Internet protocol suite
[15881] Java (programming language)
[16009] JPEG
[16226] JUnit

[16389] Java virtual machine
[16629] KDE
[16750] Kleene star
[16794] Kilobyte
[17178] KL0
[17212] KOMPILER
[17224] Kent Recursive Calculator
[17731] LiveScript
[17739] Local area network
[17927] Logic programming
[18004] LALR parser
[18016] Lisp (programming language)
[18030] LR parser
[18136] Literate programming
[18152] Logical conjunction
[18155] Lazy evaluation
[18167] Linked list
[18171] Linear search
[18195] LaTeX
[18203] Lambda calculus
[18334] Logo (programming language)
[18508] Lightweight Directory Access Protocol
[18529] Lynx (web browser)
[18530] Lynx (programming language)
[18566] Linker (computing)
[18692] Lint (software)
[18826] MD5
[18847] Multics
[18890] Microsoft Windows
[18910] Markup language
[19001] Microsoft
[19045] MIME
[19545] MySQL
[19550] Multiple inheritance
[19553] Microprocessor
[19609] Memory leak
[19673] MP3
[19723] MUMPS
[19726] Mercury (programming language)
[19918] Megabyte
[19945] Motherboard
[19962] Mesa (programming language)
[19999] Microcode
[20003] Multitier architecture
[20029] Multics Relational Data Store
[20034] Mutual recursion
[20039] Merge sort
[20055] Moving Picture Experts Group
[20070] Memory address register
[20072] Microassembler
[20170] MIPS architecture
[20178] MOO (programming language)
[20266] Mainframe computer
[20272] Minicomputer
[20340] Mary (programming language)
[20362] Merge algorithm
[20412] MATLAB
[20556] Meta element
[20560] Macro (computer science)
[20607] ML (programming language)
[20640] MacOS
[20683] Machine code
[20824] Modula
[20901] Malware
[21150] Nibble
[21506] Numerical analysis
[21523] Artificial neural network
[21571] Nial
[21652] Natural language processing
[21796] Namespace
[22194] Operating system
[22290] Open-source license
[22330] Octal
[22373] Object code
[22496] Oberon (programming language)
[22660] Occam (programming language)
[22693] Operator overloading
[22758] List of object-oriented programming languages
[22826] Object database
[23015] Programming language
[23485] Prolog
[23577] Partial function
[23630] Programmed Data Processor
[23659] Plug-in (computing)
[23665] Pixel
[23708] PL/I
[23773] Pascal (programming language)
[23801] Procedural programming
[23824] PostgreSQL
[23862] Python (programming language)
[23939] Perl
[24077] PDF
[24080] PostScript
[24107] Peer-to-peer
[24131] PHP
[24281] PowerPC
[24304] Password
[24306] Portable Network Graphics
[24400] Pair programming
[24444] Page description language
[24485] Priority queue
[24510] Pushdown automaton
[24722] P-code machine
[24829] Primitive recursive function
[24947] Pong
[24970] PA-RISC
[25030] Plain text
[25204] Qt (software)
[25213] QWERTY
[25220] Quantum computing
[25231] QuickTime
[25265] Queue (abstract data type)
[25270] Quine (computing)
[25407] Recursion
[25540] Request for Comments
[25612] Random access
[25717] Regular expression
[25723] Regular language
[25742] Raster graphics
[25748] Router (computing)
[25750] Routing
[25768] Ruby (programming language)
[25855] Regular grammar
[25871] Code refactoring
[25873] Relational database
[25983] Regular semantics
[25989] RGB color model
[26123] Real-time operating system
[26201] Reduced instruction set computer
[26220] Relational model
[26344] Register transfer language
[26346] Remote procedure call
[26384] Rebol
[26490] Reference counting
[26526] Referential transparency

A.5 Open Source Android Projects

Content of Topic_Cohesiveness/android-projects.txt *from* https://zenodo.org/records/7835197

acr.browser.lightning_101
ar.rulosoft.mimanganu_129
at.bitfire.davdroid_301000002
at.linuxtage.companion_1700002
be.digitalia.fosdem_1700201
be.mygod.vpnhotspot_220
ca.rmen.android.networkmonitor_13200
ca.rmen.android.poetassistant_113001
ca.rmen.android.scrumchatter_10606
ch.bailu.aat_31
ch.blinkenlights.android.vanilla_10850
ch.hgdev.toposuite_69
chat.rocket.android_2077
com.achep.acdisplay_76
com.adam.aslfms_61
com.adonai.manman_171
com.amaze.filemanager_54
com.android.music_2
com.app.missednotificationsreminder_2010302021
com.apps.adrcotfas.goodtime_118
com.atr.tedit_14
com.averi.worldscribe_23
com.bernaferrari.changedetection_32
com.better.alarm_30505
com.biglybt.android.client_1020600
com.dalthed.tucan_38
com.danvelazco.fbwrapper_20170312
com.dfa.hubzilla_android_46
com.dozingcatsoftware.bouncy_25
com.etesync.syncadapter_107
com.forrestguice.suntimeswidget_61
com.fsck.k9_27014

com.geecko.QuickLyric_131
com.gelakinetic.mtgfam_73
com.ghostsq.commander_396
com.github.axet.binauralbeats_157
com.github.dfa.diaspora_android_45
com.github.ruleant.getback_gps_60
com.google.android.stardroid_1480
com.google.zxing.client.android_108
com.gulshansingh.hackerlivewallpaper_13
com.haringeymobile.ukweather_27
com.hexad.bluezime_20
com.hobbyone.HashDroid_20
com.ichi2.anki_20907300
com.junjunguo.pocketmaps_34
com.keylesspalace.tusky_71
com.liato.bankdroid_222
com.llamacorp.equate_9
com.lukekorth.screennotifications_22
com.mantz_it.rfanalyzer_1303
com.matoski.adbm_27
com.maxfour.music_12
com.mde.potdroid_82
com.metinkale.prayer_215
com.mikifus.padland_20
com.moez.QKSMS_2213
com.morlunk.mumbleclient_73
com.namelessdev.mpdroid_58
com.nbossard.packlist_19
com.newsblur_163
com.nextcloud.client_30110190
com.nononsenseapps.feeder_74
com.nononsenseapps.notepad_57130
com.nutomic.syncthingandroid_4227
com.orgzly_152
com.oriondev.moneywallet_71
com.owncloud.android_21400201
com.perflyst.twire_513
com.pitchedapps.frost_2040400
com.poupa.vinylmusicplayer_168
com.quaap.bookymcbookface_430
com.quaap.launchtime_850
com.quaap.primary_33
com.rareventure.gps2_91
com.ruesga.android.wallpapers.photophase_1036
com.seafile.seadroid2_101
com.sunyata.kindmind_65
com.termux_94
com.u17od.upm_20
com.ulicae.cinelog_26
com.vonglasow.michael.satstat_3030
com.vuze.android.remote_82
com.wangdaye.mysplash_348
com.wmstein.tourcount_322
com.wmstein.transektcount_324
com.zoffcc.applications.aagtl_36
cx.ring_238
damo.three.ie_19
de.blau.android_1407
de.danoeh.antennapod_1080195
de.geeksfactory.opacclient_218
de.k3b.android.androFotoFinder_47
de.karbach.tac_6
de.kugihan.dictionaryformids.hmi_android_131
de.luhmer.owncloudnewsreader_152
de.marmaro.krt.ffupdater_34
de.qspool.clementineremote_759
de.retujo.bierverkostung_4
de.skubware.opentraining_31
de.srlabs.snoopsnitch_39
de.sudoq_26
de.syss.MifareClassicTool_48
de.tobiasbielefeld.solitaire_71
de.t_dankworth.secscanqr_19
de.vanitasvitae.enigmandroid_18
de.vibora.viborafeed_28
de.westnordost.streetcomplete_1904
de.yaacc_26
dev.ukanth.ufirewall_19450
ee.ioc.phon.android.speak_1712
es.usc.citius.servando.calendula_42
eu.kanade.tachiyomi_42
eu.siacs.conversations_383
eu.sum7.conversations_383
felixwiemuth.lincal_13
fi.kroon.vadret_24
fr.free.nrw.commons_561
fr.neamar.kiss_179
github.daneren2005.dsub_158
godau.fynn.dsbdirect_36
in.blogspot.anselmbros.torchie_34
indrora.atomic_21
info.schnatterer.nusic_22
io.github.hidroh.materialistic_79
io.github.tjg1.nori_15
io.pslab_21
it.niedermann.owncloud.notes_2012000
it.reyboz.bustorino_28
jackpal.androidterm_72
joshuatee.wx_55420
kaljurand_at_gmail_dot_com.diktofon_983
libretasks.app_22
me.blog.korn123.easydiary_195
me.ccrama.redditslide_323
me.kuehle.carreport_79
mobi.maptrek_77
mobi.omegacentauri.SendReduced_1600
net.cyclestreets_1579
net.czlee.debatekeeper_23
net.ddns.mlsoftlaberge.trycorder_523
net.eneiluj.nextcloud.phonetrack_18
net.gorry.android.input.nicownng_201412041
net.i2p.android.router_4745255
net.kervala.comicsreader_28
net.kourlas.voipms_sms_123
net.mabako.steamgifts_1005511
net.osmand.plus_363
net.osmtracker_48
net.reichholf.dreamdroid_439
net.sf.times_37
net.sourceforge.opencamera_77
net.usikkert.kouchat.android_16
net.wigle.wigleandroid_251
nodomain.freeyourgadget.gadgetbridge_173
ohi.andre.consolelauncher_205
ohm.quickdice_48
openfoodfacts.github.scrachx.openfood_328
org.adaway_40304
org.andstatus.app_320
org.bienvenidoainternet.app_13
org.bottiger.podcast_424
org.connectbot_10906000
org.coolreader_32380
org.cprados.wificellmanager_19
org.c_base.c_beam_29
org.dmfs.tasks_78500
org.dolphinemu.dolphinemu_14523
org.ea.sqrl_52
org.epstudios.epmobile_60
org.fdroid.fdroid_1008050
org.fitchfamily.android.dejavu_21
org.floens.chan_30002
org.freshrss.easyrss_706

org.gateshipone.malp_32
org.gateshipone.odyssey_31
org.gdroid.gdroid_10002
org.isoron.uhabits_38
org.kontalk_440
org.mariotaku.twidere_511
org.moire.opensudoku_20200323
org.moire.ultrasonic_72
org.openbmap_27
org.openhab.habdroid_269
org.openintents.filemanager_40
org.openintents.notepad_10084
org.openintents.shopping_100213
org.petero.droidfish_91
org.pocketworkstation.pckeyboard_1041001
org.primftpd_49
org.quantumbadger.redreader_89
org.schabi.newpipelegacy_60
org.schabi.newpipe_930
org.schabi.terminightor_12
org.secuso.privacyfriendlydame_3
org.secuso.privacyfriendlyintervaltimer_4
org.secuso.privacyfriendlysudoku_8
org.secuso.privacyfriendlyweather_6
org.servalproject_2371
org.sipdroid.sipua_115
org.smssecure.smssecure_211
org.softeg.slartus.forpdaplus_637
org.sufficientlysecure.keychain_55000
org.sugr.gearshift_88
org.telegram.messenger_19479
org.tomdroid_14
org.totschnig.myexpenses_410
org.toulibre.capitoledulibre_11
org.transdroid.search_36
org.ttrssreader_1946
org.videolan.vlc_13021208
org.wikipedia_50308
org.xbmc.kore_26
org.yaaic_13
org.yaxim.androidclient_53
org.zephyrsoft.trackworktime_28
pk.contender.earmouse_30
privacyfriendlyshoppinglist.secuso.org.privacyfriendlyshoppinglist_7
rkr.simplekeyboard.inputmethod_69
ru.gelin.android.weather.notification_54
ru.playsoftware.j2meloader_80
ryey.easer_123
se.leap.bitmaskclient_147
uk.co.bitethebullet.android.token_6
uk.co.yahoo.p1rpp.calendartrigger_7
vocabletrainer.heinecke.aron.vocabletrainer_20
wseemann.media.romote_16

Appendix B

Replication Data for Tutorial Design Variations

We published the data necessary to verify and replicate our study of tutorial design (Chapter 4) at https://zenodo.org/records/5075903 [14]. Table B.1 details the content of the artifact. The last column indicates which parts of the artifact is reproduced in the sections of this appendix.

Table B.1: Content of the Data Artifact for Our Study of Tutorial Design

Path	**Description**	**Sec.**
README.md	Description of the artifact.	
topics/android-tags.csv	List of all Stack Overflow tags containing the substring android.	
topics/topics.json	List of all tags used in our study, with additional information.	B.1
tutorials/urls.txt	List of URLs of all the tutorial pages used in our study.	B.2
tutorials/sections.csv	List of parsed tutorial sections.	
mapping/annotations_[...].csv	Results of the manual mapping between Android topics (i.e., Stack Overflow tags) and tutorial sections.	
mapping/all_annotations.csv	Incidence matrix of the coverage of the 393 topics (rows) by the 3 tutorials (columns).	B.3
cooccurrences/counts.csv	Number of Stack Overflow posts associated with each pair of tags.	

B.1 Android-Related Stack Overflow Tags Used in Our Study

Tags listed in topics/topics.json *from* https://zenodo.org/records/5075903

android-studio
android-layout
android-fragments
android-intent
android-activity
android-recyclerview
android-listview
android-asynctask
android-ndk
android-gradle-plugin
android-edittext
android-emulator
android-viewpager
android-actionbar
android-sqlite
android-webview
android-volley
android-service
android-camera
android-widget
react-native-android
android-alertdialog
android-manifest
android-animation
android-linearlayout
android-mediaplayer
android-arrayadapter
android-notifications
android-imageview
android-view
android-canvas
android-support-library
android-room
android-spinner
android-custom-view
android-contentprovider
android-permissions
android-adapter
android-espresso
android-toolbar
android-xml
android-appcompat
android-softkeyboard
android-videoview
android-mapview
android-source
android-resources
android-constraintlayout
android-tabhost
android-button
android-contacts
android-cardview
android-fragmentactivity
android-databinding
android-lifecycle
android-theme
facebook-android-sdk
android-glide
android-context
android-testing
android-tablayout
androidx
android-styles
android-dialogfragment
android-bluetooth
mpandroidchart
android-library
rx-android
android-pendingintent
android-scrollview
android-gridview
android-livedata
android-bitmap
android-sensors
android-progressbar
android-sdcard
android-architecture-components
android-coordinatorlayout
android-relativelayout
android-dialog
android-ui
android-location
android-menu
android-tablelayout
android-listfragment
android-broadcast
android-collapsingtoolbarlayout
android-gallery
android-logcat
android-alarms
android-sdk-tools
android-proguard
android-tv
android-jetpack
android-calendar
android-appwidget
android-camera-intent
android-file
android-viewholder
android-youtube-api
android-broadcastreceiver
android-cursoradapter
android-networking
appium-android
android-maps
android-contentresolver
android-cursor
android-orientation
android-download-manager
google-drive-android-api
android-uiautomator
android-handler
android-keystore
android-ksoap2
android-appbarlayout
android-design-library
android-c2dm
android-multidex
android-viewmodel
android-audiomanager
android-navigation
android-checkbox
android-datepicker
android-actionbar-compat
android-mediarecorder
android-workmanager
android-architecture-navigation
android-gridlayout
android-launcher
android-facebook
android-gps
android-syncadapter
android-textinputlayout
android-external-storage
android-things
android-seekbar
android-vectordrawable
android-nestedscrollview
kotlin-android-extensions
android-sharedpreferences
ibeacon-android
opencv4android
android-assets
android-toast
android-geofence
android-install-apk
android-browser
android-jobscheduler
android-fileprovider
android-cursorloader
android-pay
android-productflavors
android-windowmanager
android-annotations
android-database
material-components-android
android-imagebutton
android-intentservice
android-instant-apps
android-graphview
androiddesignsupport
android-security
android-keypad
android-pageradapter
android-layout-weight
androidhttpclient
android-instrumentation
android-loadermanager
cocos2d-android
android-search
android-service-binding
android-snackbar
android-debug
android-navigationview
android-wake-lock
android-ffmpeg
mapbox-android
android-notification-bar
android-billing
android-memory
android-homebutton
android-timepicker
android-popupwindow
android-radiobutton
android-optionsmenu
android-event

android-statusbar
android-lvl
android-radiogroup
android-developer-api
android-sharing
android-fusedlocation
android-lint
android-fullscreen
android-ibeacon
android-wear-data-api
android-adapterview
android-nested-fragment
android-app-bundle
android-accessibility
android-icons
android-tabactivity
android-async-http
android-bundle
android-mvp
android-gesture
calabash-android
android-api-levels
android-paging
android-settings
android-support-design
activeandroid
android-actionbaractivity
android-ble
basic4android
rxandroidble
android-holo-everywhere
android-backup-service
android-maven-plugin
android-hardware
android-vision
android-binder
android-loader
android-contextmenu
androidviewclient
android.mk
android-actionmode
android-elevation
android-switch
android-shape
android-textwatcher
android-togglebutton
android-mediascanner
android-runonuithread
android-auto
android-touch-event
android-searchmanager
android-doze
android-storage
android-wallpaper
android-app-indexing
quickblox-android
android-beam
android-looper
android-applicationinfo
android-mediaprojection
android-internal-storage
android-chips
android-strictmode
android-management-api
android-bottomnav
android-lru-cache
android-instant-run
pocketsphinx-android
android-task
android-recents
android-mediasession
android-authenticator
android-xmlpullparser
android-account
android-wear-notification
android-jetpack-compose
titanium-android
android-viewbinder
android-bottomappbar
android-query
android-implicit-intent
android-app-signing
android-mediacodec
android-maps-utils
android-usb
sqlcipher-android
android-build-type
android-immersive
android-biometric-prompt
android-print-framework
android-number-picker
android-dialer
parse-android-sdk
android-application-class
android-intent-chooser
android-wrap-content
android-market-filtering
android-device-monitor
android-for-work
android-espresso-recorder
spring-android
android-shapedrawable
android-junit
android-profiler
android-jack-and-jill
android-powermanager
qtandroidextras
android-parsequeryadapter
nineoldandroids
android-applicationrecord
android-traceview
android-speech-api
android-customtabs
android-simple-facebook
android-screen-pinning
vimeo-android
chrome-for-android
android-ondestroy
android-percentrelativelayout
android-jetifier
android-pdf-api
xamarin-android-player
androidasync-koush
android-enterprise
androidpdfviewer
android-components
android-job
air-android
android-emulator-plugin
android-crop
android-capture
android-side-navigation
apollo-android
android-jsinterface
android-afilechooser
android-palette
android-automotive
sbt-android-plugin
android-ide
android-multiple-users
android-custom-drawable
androidappsonchromeos
android-monkey
svg-android
android-guava
android-viewbinding
android-open-accessory
android-vertical-seekbar
android-things-console
android-os-handler
android-wheel
android-droidtext
android-view-invalidate
android-percent-library
qandroidjniobject
android-bootstrap
android-apt
android-tools-namespace
android-drm
android-kenburnsview
android-soong
android-cts
dronekit-android
appsflyer-android-sdk
android-ktx
android-managed-profile
android-wear-complication
android-vts
android-cling
android-vitals
android-largeheap
android-json-rpc
parse-sdk-android
android-appwidget-list
android-sparsearray
android-navigation-editor
android-spellcheck
android-restrictions
android-time-square
android-sdk-plugin
android-activityrecord
android-contact-mimetype
android-app-ops
android-subscriptionmanager
android-tap-and-pay
android-singleline
android-assertj
android-bootstrap-widgets
turbolinks-android
android-notification.mediastyle
android-tradefederation
android-extracted-text
ms-android-emulator
android-pixel-copy
android-storm
android-simon-datepicker
android-snapshot
android-standout
android-slider
android-drawable-importer
android-cognalys
android-kripton
wikimedia-android-data-client
httpclientandroidlib
paper-android
android-ble-library
androidx-lifecycle

B.2 List of Tutorial Pages

Content of tutorials/urls.txt *from* https://zenodo.org/records/5075903

AppBasic

https://developer.android.com/training/basics/firstapp
https://developer.android.com/training/basics/firstapp/creating-project
https://developer.android.com/training/basics/firstapp/running-app
https://developer.android.com/training/basics/firstapp/building-ui
https://developer.android.com/training/basics/firstapp/starting-activity
https://developer.android.com/guide/components/fundamentals
https://developer.android.com/guide/topics/resources/providing-resources
https://developer.android.com/guide/topics/resources/runtime-changes
https://developer.android.com/guide/topics/resources/localization
https://developer.android.com/guide/topics/resources/pseudolocales
https://developer.android.com/guide/topics/resources/internationalization
https://developer.android.com/guide/topics/resources/multilingual-support
https://developer.android.com/guide/topics/resources/complex-xml-resources
https://developer.android.com/guide/topics/resources/available-resources
https://developer.android.com/guide/topics/resources/animation-resource
https://developer.android.com/guide/topics/resources/color-list-resource
https://developer.android.com/guide/topics/resources/drawable-resource
https://developer.android.com/guide/topics/resources/layout-resource
https://developer.android.com/guide/topics/resources/menu-resource
https://developer.android.com/guide/topics/resources/string-resource
https://developer.android.com/guide/topics/resources/style-resource
https://developer.android.com/guide/topics/resources/font-resource
https://developer.android.com/guide/topics/resources/more-resources
https://developer.android.com/guide/topics/manifest/manifest-intro
https://developer.android.com/guide/topics/manifest/action-element
https://developer.android.com/guide/topics/manifest/activity-element
https://developer.android.com/guide/topics/manifest/activity-alias-element
https://developer.android.com/guide/topics/manifest/application-element
https://developer.android.com/guide/topics/manifest/category-element
https://developer.android.com/guide/topics/manifest/compatible-screens-element
https://developer.android.com/guide/topics/manifest/data-element
https://developer.android.com/guide/topics/manifest/grant-uri-permission-element
https://developer.android.com/guide/topics/manifest/instrumentation-element
https://developer.android.com/guide/topics/manifest/intent-filter-element
https://developer.android.com/guide/topics/manifest/manifest-element
https://developer.android.com/guide/topics/manifest/meta-data-element
https://developer.android.com/guide/topics/manifest/path-permission-element
https://developer.android.com/guide/topics/manifest/permission-element
https://developer.android.com/guide/topics/manifest/permission-group-element
https://developer.android.com/guide/topics/manifest/permission-tree-element
https://developer.android.com/guide/topics/manifest/provider-element
https://developer.android.com/guide/topics/manifest/receiver-element
https://developer.android.com/guide/topics/manifest/service-element
https://developer.android.com/guide/topics/manifest/supports-gl-texture-element
https://developer.android.com/guide/topics/manifest/supports-screens-element
https://developer.android.com/guide/topics/manifest/uses-configuration-element
https://developer.android.com/guide/topics/manifest/uses-feature-element
https://developer.android.com/guide/topics/manifest/uses-library-element
https://developer.android.com/guide/topics/manifest/uses-permission-element
https://developer.android.com/guide/topics/manifest/uses-permission-sdk-23-element
https://developer.android.com/guide/topics/manifest/uses-sdk-element
https://developer.android.com/guide/topics/permissions/overview
https://developer.android.com/training/permissions/requesting
https://developer.android.com/training/permissions/usage-notes
https://developer.android.com/guide/topics/permissions/default-handlers
https://developer.android.com/guide/topics/permissions/defining

DevFundamental

https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/
.../unit-1-get-started/lesson-1-build-your-first-app/1-0-c-introduction-to-android/1-0-c-introduction-to-android.html

.../unit-1-get-started/lesson-1-build-your-first-app/1-1-c-your-first-android-app/1-1-c-your-first-android-app.html
.../unit-1-get-started/lesson-1-build-your-first-app/1-2-c-layouts-and-resources-for-the-ui/1-2-c-layouts-and-resources-for-the-ui.html
.../unit-1-get-started/lesson-1-build-your-first-app/1-3-c-text-and-scrolling-views/1-3-c-text-and-scrolling-views.html
.../unit-1-get-started/lesson-1-build-your-first-app/1-4-c-resources_to_help_you_learn/1-4-c-resources_to_help_you_learn.html
.../unit-1-get-started/lesson-2-activities-and-intents/2-1-c-activities-and-intents/2-1-c-activities-and-intents.html
.../unit-1-get-started/lesson-2-activities-and-intents/2-2-c-activity-lifecycle-and-state/2-2-c-activity-lifecycle-and-state.html
.../unit-1-get-started/lesson-2-activities-and-intents/2-3-c-implicit-intents/2-3-c-implicit-intents.html
.../unit-1-get-started/lesson-3-testing,-debugging,-and-using-support-libraries/3-1-c-the-android-studio-debugger/3-1-c-the-android-studio-debugger.html
.../unit-1-get-started/lesson-3-testing,-debugging,-and-using-support-libraries/3-2-c-app-testing/3-2-c-app-testing.html
.../unit-1-get-started/lesson-3-testing,-debugging,-and-using-support-libraries/3-3-c-the-android-support-library/3-3-c-the-android-support-library.html
.../unit-2-user-experience/lesson-4-user-interaction/4-1-c-buttons-and-clickable-images/4-1-c-buttons-and-clickable-images.html
.../unit-2-user-experience/lesson-4-user-interaction/4-2-c-input-controls/4-2-c-input-controls.html
.../unit-2-user-experience/lesson-4-user-interaction/4-3-c-menus-and-pickers/4-3-c-menus-and-pickers.html
.../unit-2-user-experience/lesson-4-user-interaction/4-4-c-user-navigation/4-4-c-user-navigation.html
.../unit-2-user-experience/lesson-4-user-interaction/4-5-c-recyclerview/4-5-c-recyclerview.html
.../unit-2-user-experience/lesson-5-delightful-user-experience/5-1-c-drawables-styles-and-themes/5-1-c-drawables-styles-and-themes.html
.../unit-2-user-experience/lesson-5-delightful-user-experience/5-2-c-material-design/5-2-c-material-design.html
.../unit-2-user-experience/lesson-5-delightful-user-experience/5-3-c-resources-for-adaptive-layouts/5-3-c-resources-for-adaptive-layouts.html
.../unit-2-user-experience/lesson-6-testing-your-ui/6-1-c-ui-testing/6-1-c-ui-testing.html
.../unit-3-working-in-the-background/lesson-7-background-tasks/7-1-c-asynctask-and-asynctaskloader/7-1-c-asynctask-and-asynctaskloader.html
.../unit-3-working-in-the-background/lesson-7-background-tasks/7-2-c-internet-connection/7-2-c-internet-connection.html
.../unit-3-working-in-the-background/lesson-7-background-tasks/7-3-c-broadcasts/7-3-c-broadcasts.html
.../unit-3-working-in-the-background/lesson-7-background-tasks/7-4-c-services/7-4-c-services.html
.../unit-3-working-in-the-background/lesson-8-alarms-and-schedulers/8-1-c-notifications/8-1-c-notifications.html
.../unit-3-working-in-the-background/lesson-8-alarms-and-schedulers/8-2-c-alarms/8-2-c-alarms.html
.../unit-3-working-in-the-background/lesson-8-alarms-and-schedulers/8-3-c-efficient-data-transfer/8-3-c-efficient-data-transfer.html
.../unit-4-saving-user-data/lesson-9-preferences-and-settings/9-0-c-data-storage/9-0-c-data-storage.html
.../unit-4-saving-user-data/lesson-9-preferences-and-settings/9-1-c-shared-preferences/9-1-c-shared-preferences.html
.../unit-4-saving-user-data/lesson-9-preferences-and-settings/9-2-c-app-settings/9-2-c-app-settings.html
.../unit-4-saving-user-data/lesson-10-storing-data-with-room/10-0-c-sqlite-primer/10-0-c-sqlite-primer.html
.../unit-4-saving-user-data/lesson-10-storing-data-with-room/10-1-c-room-livedata-viewmodel/10-1-c-room-livedata-viewmodel.html
.../appendix/appendix-utilities/appendix-utilities.html

DevStarter

http://www.vogella.com/tutorials/Android/article.html
http://www.vogella.com/tutorials/AndroidStudioTooling/article.html
http://www.vogella.com/tutorials/AndroidKotlin/article.html
http://www.vogella.com/tutorials/AndroidIntent/article.html
http://www.vogella.com/tutorials/AndroidLifeCycle/article.html
http://www.vogella.com/tutorials/AndroidPermissions/article.html
http://www.vogella.com/tutorials/AndroidRecyclerView/article.html
http://www.vogella.com/tutorials/AndroidDatabinding/article.html
http://www.vogella.com/tutorials/AndroidListView/article.html
http://www.vogella.com/tutorials/AndroidLogging/article.html

B.3 Topic Coverage of Three Android Tutorials

Content of mappings/all_annotations.csv *from* https://zenodo.org/records/5075903

Topic	Covered in AppBasic	Covered in DevFundamental	Covered in DevStarter
activeandroid	FALSE	FALSE	FALSE
air-android	FALSE	FALSE	FALSE
android-accessibility	FALSE	FALSE	FALSE
android-account	FALSE	FALSE	FALSE
android-actionbar	FALSE	TRUE	FALSE
android-actionbar-compat	FALSE	TRUE	FALSE
android-actionbaractivity	FALSE	FALSE	FALSE
android-actionmode	FALSE	TRUE	TRUE
android-activity	TRUE	TRUE	TRUE
android-activityrecord	FALSE	FALSE	FALSE
android-adapter	FALSE	TRUE	TRUE
android-adapterview	FALSE	TRUE	TRUE
android-afilechooser	FALSE	FALSE	FALSE

Topic	Covered in AppBasic	Covered in DevFundamental	Covered in DevStarter
android-alarms	FALSE	TRUE	FALSE
android-alertdialog	FALSE	TRUE	FALSE
android-animation	TRUE	TRUE	TRUE
android-annotations	FALSE	FALSE	FALSE
android-api-levels	TRUE	TRUE	TRUE
android-app-bundle	FALSE	TRUE	FALSE
android-app-indexing	FALSE	FALSE	FALSE
android-app-ops	FALSE	FALSE	FALSE
android-app-signing	FALSE	FALSE	FALSE
android-appbarlayout	FALSE	TRUE	FALSE
android-appcompat	FALSE	TRUE	FALSE
android-application-class	FALSE	FALSE	TRUE
android-applicationinfo	TRUE	TRUE	TRUE
android-applicationrecord	FALSE	FALSE	FALSE
android-appwidget	FALSE	TRUE	FALSE
android-appwidget-list	FALSE	FALSE	FALSE
android-apt	FALSE	FALSE	FALSE
android-architecture-components	TRUE	TRUE	TRUE
android-architecture-navigation	FALSE	TRUE	TRUE
android-arrayadapter	FALSE	TRUE	TRUE
android-assertj	FALSE	FALSE	FALSE
android-assets	FALSE	TRUE	TRUE
android-async-http	FALSE	FALSE	FALSE
android-asynctask	FALSE	TRUE	FALSE
android-audiomanager	FALSE	FALSE	FALSE
android-authenticator	FALSE	FALSE	FALSE
android-auto	FALSE	FALSE	FALSE
android-automotive	TRUE	FALSE	FALSE
android-backup-service	FALSE	TRUE	FALSE
android-beam	FALSE	FALSE	FALSE
android-billing	FALSE	FALSE	FALSE
android-binder	FALSE	TRUE	FALSE
android-biometric-prompt	FALSE	FALSE	FALSE
android-bitmap	TRUE	TRUE	TRUE
android-ble	FALSE	FALSE	FALSE
android-ble-library	FALSE	FALSE	FALSE
android-bluetooth	TRUE	FALSE	FALSE
android-bootstrap	FALSE	FALSE	FALSE
android-bootstrap-widgets	FALSE	FALSE	FALSE
android-bottomappbar	FALSE	FALSE	FALSE
android-bottomnav	FALSE	FALSE	FALSE
android-broadcast	TRUE	TRUE	TRUE
android-broadcastreceiver	TRUE	TRUE	TRUE
android-browser	FALSE	FALSE	FALSE
android-build-type	FALSE	TRUE	FALSE
android-bundle	TRUE	TRUE	TRUE
android-button	TRUE	TRUE	TRUE
android-c2dm	FALSE	FALSE	FALSE
android-calendar	FALSE	FALSE	FALSE
android-camera	TRUE	FALSE	TRUE
android-camera-intent	FALSE	FALSE	FALSE
android-canvas	FALSE	FALSE	FALSE
android-capture	TRUE	FALSE	FALSE
android-cardview	FALSE	TRUE	FALSE
android-checkbox	FALSE	TRUE	TRUE
android-chips	FALSE	FALSE	FALSE
android-cling	FALSE	FALSE	FALSE
android-cognalys	FALSE	FALSE	FALSE
android-collapsingtoolbarlayout	FALSE	FALSE	FALSE
android-components	FALSE	TRUE	FALSE
android-constraintlayout	TRUE	TRUE	TRUE
android-contact-mimetype	FALSE	FALSE	FALSE
android-contacts	FALSE	FALSE	TRUE
android-contentprovider	TRUE	FALSE	TRUE
android-contentresolver	TRUE	FALSE	FALSE
android-context	FALSE	TRUE	TRUE
android-contextmenu	FALSE	TRUE	FALSE
android-coordinatorlayout	FALSE	FALSE	FALSE
android-crop	FALSE	FALSE	FALSE

Topic	Covered in AppBasic	Covered in DevFundamental	Covered in DevStarter
android-cts	FALSE	FALSE	FALSE
android-cursor	FALSE	TRUE	FALSE
android-cursoradapter	FALSE	FALSE	TRUE
android-cursorloader	FALSE	FALSE	FALSE
android-custom-drawable	FALSE	TRUE	FALSE
android-custom-view	TRUE	FALSE	FALSE
android-customtabs	FALSE	FALSE	FALSE
android-database	FALSE	TRUE	FALSE
android-databinding	FALSE	FALSE	TRUE
android-datepicker	FALSE	TRUE	TRUE
android-debug	TRUE	TRUE	FALSE
android-design-library	FALSE	TRUE	FALSE
android-developer-api	FALSE	TRUE	TRUE
android-device-monitor	FALSE	FALSE	FALSE
android-dialer	FALSE	FALSE	FALSE
android-dialog	FALSE	TRUE	FALSE
android-dialogfragment	FALSE	TRUE	FALSE
android-download-manager	FALSE	FALSE	FALSE
android-doze	FALSE	TRUE	FALSE
android-drawable-importer	FALSE	FALSE	FALSE
android-drm	FALSE	FALSE	FALSE
android-droidtext	FALSE	FALSE	FALSE
android-edittext	TRUE	TRUE	TRUE
android-elevation	FALSE	TRUE	TRUE
android-emulator	TRUE	TRUE	TRUE
android-emulator-plugin	FALSE	FALSE	FALSE
android-enterprise	FALSE	FALSE	FALSE
android-espresso	FALSE	TRUE	FALSE
android-espresso-recorder	FALSE	TRUE	FALSE
android-event	FALSE	TRUE	TRUE
android-external-storage	FALSE	TRUE	FALSE
android-extracted-text	FALSE	FALSE	FALSE
android-facebook	FALSE	FALSE	FALSE
android-ffmpeg	FALSE	FALSE	FALSE
android-file	TRUE	TRUE	FALSE
android-fileprovider	FALSE	FALSE	FALSE
android-for-work	FALSE	FALSE	FALSE
android-fragmentactivity	FALSE	FALSE	FALSE
android-fragments	FALSE	TRUE	TRUE
android-fullscreen	FALSE	FALSE	FALSE
android-fusedlocation	FALSE	FALSE	FALSE
android-gallery	FALSE	FALSE	FALSE
android-geofence	FALSE	FALSE	FALSE
android-gesture	FALSE	TRUE	TRUE
android-glide	FALSE	TRUE	TRUE
android-gps	TRUE	FALSE	FALSE
android-gradle-plugin	TRUE	TRUE	TRUE
android-graphview	FALSE	FALSE	FALSE
android-gridlayout	FALSE	TRUE	TRUE
android-gridview	FALSE	FALSE	TRUE
android-guava	FALSE	FALSE	FALSE
android-handler	FALSE	TRUE	TRUE
android-hardware	TRUE	TRUE	TRUE
android-holo-everywhere	FALSE	TRUE	FALSE
android-homebutton	FALSE	FALSE	TRUE
android-ibeacon	FALSE	FALSE	FALSE
android-icons	TRUE	FALSE	TRUE
android-ide	TRUE	TRUE	TRUE
android-imagebutton	FALSE	TRUE	FALSE
android-imageview	TRUE	TRUE	TRUE
android-immersive	TRUE	FALSE	FALSE
android-implicit-intent	TRUE	TRUE	TRUE
android-install-apk	TRUE	TRUE	FALSE
android-instant-apps	FALSE	FALSE	FALSE
android-instant-run	FALSE	FALSE	FALSE
android-instrumentation	TRUE	TRUE	FALSE
android-intent	TRUE	TRUE	TRUE
android-intent-chooser	TRUE	TRUE	FALSE
android-intentservice	FALSE	TRUE	FALSE

Topic	Covered in AppBasic	Covered in DevFundamental	Covered in DevStarter
android-internal-storage	FALSE	TRUE	FALSE
android-jack-and-jill	FALSE	FALSE	FALSE
android-jetifier	FALSE	FALSE	FALSE
android-jetpack	FALSE	FALSE	FALSE
android-jetpack-compose	FALSE	FALSE	FALSE
android-job	TRUE	FALSE	FALSE
android-jobscheduler	TRUE	TRUE	FALSE
android-jsinterface	FALSE	FALSE	FALSE
android-json-rpc	FALSE	FALSE	FALSE
android-junit	FALSE	TRUE	FALSE
android-kenburnsview	FALSE	FALSE	FALSE
android-keypad	TRUE	FALSE	FALSE
android-keystore	FALSE	FALSE	TRUE
android-kripton	FALSE	FALSE	FALSE
android-ksoap2	FALSE	FALSE	FALSE
android-ktx	FALSE	FALSE	FALSE
android-largeheap	TRUE	FALSE	FALSE
android-launcher	FALSE	TRUE	FALSE
android-layout	TRUE	TRUE	TRUE
android-layout-weight	FALSE	FALSE	TRUE
android-library	FALSE	FALSE	TRUE
android-lifecycle	FALSE	TRUE	TRUE
android-linearlayout	FALSE	TRUE	TRUE
android-lint	FALSE	FALSE	FALSE
android-listfragment	FALSE	FALSE	TRUE
android-listview	TRUE	FALSE	TRUE
android-livedata	FALSE	TRUE	TRUE
android-loader	FALSE	TRUE	FALSE
android-loadermanager	FALSE	TRUE	FALSE
android-location	FALSE	FALSE	FALSE
android-logcat	FALSE	TRUE	TRUE
android-looper	FALSE	FALSE	FALSE
android-lru-cache	FALSE	FALSE	FALSE
android-lvl	FALSE	FALSE	FALSE
android-managed-profile	FALSE	FALSE	FALSE
android-management-api	FALSE	FALSE	FALSE
android-manifest	TRUE	TRUE	TRUE
android-maps	FALSE	FALSE	FALSE
android-maps-utils	FALSE	FALSE	FALSE
android-mapview	FALSE	FALSE	FALSE
android-market-filtering	TRUE	FALSE	FALSE
android-maven-plugin	FALSE	FALSE	FALSE
android-mediacodec	FALSE	FALSE	FALSE
android-mediaplayer	FALSE	FALSE	FALSE
android-mediaprojection	FALSE	FALSE	FALSE
android-mediarecorder	FALSE	FALSE	FALSE
android-mediascanner	FALSE	FALSE	FALSE
android-mediasession	FALSE	FALSE	FALSE
android-memory	TRUE	TRUE	TRUE
android-menu	TRUE	TRUE	TRUE
android-monkey	FALSE	FALSE	FALSE
android-multidex	FALSE	FALSE	FALSE
android-multiple-users	FALSE	FALSE	FALSE
android-mvp	FALSE	FALSE	TRUE
android-navigation	FALSE	TRUE	FALSE
android-navigation-editor	TRUE	FALSE	FALSE
android-navigationview	FALSE	TRUE	FALSE
android-ndk	FALSE	FALSE	FALSE
android-nested-fragment	FALSE	FALSE	FALSE
android-nestedscrollview	FALSE	FALSE	FALSE
android-networking	FALSE	TRUE	FALSE
android-notification-bar	FALSE	TRUE	FALSE
android-notification.mediastyle	FALSE	TRUE	FALSE
android-notifications	FALSE	TRUE	TRUE
android-number-picker	FALSE	FALSE	FALSE
android-ondestroy	FALSE	TRUE	TRUE
android-open-accessory	FALSE	FALSE	FALSE
android-optionsmenu	FALSE	TRUE	FALSE
android-orientation	TRUE	TRUE	TRUE

Topic	Covered in AppBasic	Covered in DevFundamental	Covered in DevStarter
android-os-handler	FALSE	TRUE	FALSE
android-pageradapter	FALSE	TRUE	FALSE
android-paging	FALSE	TRUE	FALSE
android-palette	FALSE	TRUE	FALSE
android-parsequeryadapter	FALSE	FALSE	FALSE
android-pay	FALSE	FALSE	FALSE
android-pdf-api	FALSE	FALSE	FALSE
android-pendingintent	FALSE	TRUE	FALSE
android-percent-library	FALSE	FALSE	FALSE
android-percentrelativelayout	FALSE	FALSE	FALSE
android-permissions	TRUE	TRUE	TRUE
android-pixel-copy	FALSE	FALSE	FALSE
android-popupwindow	FALSE	FALSE	FALSE
android-powermanager	FALSE	FALSE	FALSE
android-print-framework	FALSE	FALSE	FALSE
android-productflavors	FALSE	FALSE	FALSE
android-profiler	FALSE	TRUE	FALSE
android-progressbar	FALSE	FALSE	FALSE
android-proguard	FALSE	TRUE	FALSE
android-query	FALSE	FALSE	FALSE
android-radiobutton	FALSE	TRUE	FALSE
android-radiogroup	FALSE	TRUE	FALSE
android-recents	FALSE	TRUE	FALSE
android-recyclerview	FALSE	TRUE	TRUE
android-relativelayout	FALSE	TRUE	TRUE
android-resources	TRUE	TRUE	TRUE
android-restrictions	FALSE	FALSE	FALSE
android-room	FALSE	TRUE	FALSE
android-runonuithread	FALSE	FALSE	FALSE
android-screen-pinning	FALSE	FALSE	FALSE
android-scrollview	FALSE	TRUE	TRUE
android-sdcard	FALSE	FALSE	FALSE
android-sdk-plugin	FALSE	FALSE	FALSE
android-sdk-tools	TRUE	TRUE	TRUE
android-search	FALSE	FALSE	FALSE
android-searchmanager	FALSE	FALSE	FALSE
android-security	FALSE	FALSE	FALSE
android-seekbar	FALSE	TRUE	FALSE
android-sensors	TRUE	TRUE	FALSE
android-service	TRUE	TRUE	TRUE
android-service-binding	TRUE	TRUE	FALSE
android-settings	FALSE	TRUE	FALSE
android-shape	TRUE	TRUE	FALSE
android-shapedrawable	TRUE	TRUE	FALSE
android-sharedpreferences	FALSE	TRUE	FALSE
android-sharing	TRUE	TRUE	TRUE
android-side-navigation	FALSE	TRUE	FALSE
android-simon-datepicker	FALSE	FALSE	FALSE
android-simple-facebook	FALSE	FALSE	FALSE
android-singleline	FALSE	FALSE	FALSE
android-slider	FALSE	FALSE	FALSE
android-snackbar	FALSE	TRUE	FALSE
android-snapshot	FALSE	FALSE	FALSE
android-softkeyboard	FALSE	TRUE	FALSE
android-soong	FALSE	FALSE	FALSE
android-source	FALSE	FALSE	FALSE
android-sparsearray	FALSE	FALSE	FALSE
android-speech-api	FALSE	FALSE	FALSE
android-spellcheck	FALSE	FALSE	FALSE
android-spinner	FALSE	TRUE	TRUE
android-sqlite	FALSE	TRUE	FALSE
android-standout	FALSE	FALSE	FALSE
android-statusbar	FALSE	TRUE	FALSE
android-storage	FALSE	TRUE	FALSE
android-storm	FALSE	FALSE	FALSE
android-strictmode	TRUE	FALSE	FALSE
android-studio	TRUE	TRUE	TRUE
android-styles	TRUE	TRUE	TRUE
android-subscriptionmanager	FALSE	FALSE	FALSE

Topic	Covered in AppBasic	Covered in DevFundamental	Covered in DevStarter
android-support-design	FALSE	TRUE	FALSE
android-support-library	FALSE	TRUE	FALSE
android-switch	FALSE	TRUE	TRUE
android-syncadapter	FALSE	FALSE	FALSE
android-tabactivity	FALSE	FALSE	FALSE
android-tabhost	FALSE	FALSE	FALSE
android-tablayout	FALSE	TRUE	FALSE
android-tablelayout	FALSE	TRUE	FALSE
android-tap-and-pay	FALSE	FALSE	FALSE
android-task	TRUE	TRUE	FALSE
android-testing	TRUE	TRUE	FALSE
android-textinputlayout	FALSE	FALSE	FALSE
android-textwatcher	FALSE	FALSE	FALSE
android-theme	FALSE	TRUE	TRUE
android-things	FALSE	FALSE	FALSE
android-things-console	FALSE	FALSE	FALSE
android-time-square	FALSE	FALSE	FALSE
android-timepicker	FALSE	TRUE	FALSE
android-toast	FALSE	TRUE	FALSE
android-togglebutton	FALSE	TRUE	FALSE
android-toolbar	FALSE	TRUE	FALSE
android-tools-namespace	FALSE	FALSE	FALSE
android-touch-event	FALSE	TRUE	TRUE
android-traceview	FALSE	TRUE	FALSE
android-tradefederation	FALSE	FALSE	FALSE
android-tv	FALSE	FALSE	FALSE
android-ui	TRUE	TRUE	TRUE
android-uiautomator	FALSE	TRUE	FALSE
android-usb	TRUE	FALSE	FALSE
android-vectordrawable	TRUE	TRUE	FALSE
android-vertical-seekbar	FALSE	FALSE	FALSE
android-videoview	FALSE	FALSE	FALSE
android-view	TRUE	TRUE	TRUE
android-view-invalidate	FALSE	FALSE	FALSE
android-viewbinder	FALSE	FALSE	FALSE
android-viewbinding	FALSE	FALSE	FALSE
android-viewholder	FALSE	TRUE	FALSE
android-viewmodel	TRUE	TRUE	TRUE
android-viewpager	FALSE	TRUE	FALSE
android-vision	FALSE	FALSE	FALSE
android-vitals	FALSE	FALSE	FALSE
android-volley	FALSE	FALSE	FALSE
android-vts	FALSE	FALSE	FALSE
android-wake-lock	FALSE	FALSE	FALSE
android-wallpaper	FALSE	FALSE	FALSE
android-wear-complication	FALSE	FALSE	FALSE
android-wear-data-api	FALSE	FALSE	FALSE
android-wear-notification	FALSE	FALSE	FALSE
android-webview	FALSE	TRUE	TRUE
android-wheel	FALSE	FALSE	FALSE
android-widget	TRUE	TRUE	FALSE
android-windowmanager	FALSE	FALSE	FALSE
android-workmanager	FALSE	TRUE	FALSE
android-wrap-content	TRUE	TRUE	TRUE
android-xml	TRUE	TRUE	TRUE
android-xmlpullparser	FALSE	FALSE	FALSE
android-youtube-api	FALSE	FALSE	FALSE
android.mk	FALSE	FALSE	FALSE
androidappsonchromeos	FALSE	FALSE	FALSE
androidasync-koush	FALSE	FALSE	FALSE
androiddesignsupport	FALSE	TRUE	TRUE
androidhttpclient	FALSE	TRUE	FALSE
androidpdfviewer	FALSE	FALSE	FALSE
androidviewclient	FALSE	FALSE	FALSE
androidx	FALSE	FALSE	FALSE
androidx-lifecycle	FALSE	FALSE	FALSE
apollo-android	FALSE	FALSE	FALSE
appium-android	FALSE	FALSE	FALSE
appsflyer-android-sdk	FALSE	FALSE	FALSE

Topic	Covered in AppBasic	Covered in DevFundamental	Covered in DevStarter
basic4android	FALSE	FALSE	FALSE
calabash-android	FALSE	FALSE	FALSE
chrome-for-android	FALSE	FALSE	FALSE
cocos2d-android	FALSE	FALSE	FALSE
dronekit-android	FALSE	FALSE	FALSE
facebook-android-sdk	FALSE	FALSE	FALSE
google-drive-android-api	FALSE	FALSE	FALSE
httpclientandroidlib	FALSE	FALSE	FALSE
ibeacon-android	FALSE	FALSE	FALSE
kotlin-android-extensions	FALSE	FALSE	FALSE
mapbox-android	FALSE	FALSE	FALSE
material-components-android	FALSE	FALSE	FALSE
mpandroidchart	FALSE	FALSE	FALSE
ms-android-emulator	FALSE	FALSE	FALSE
nineoldandroids	FALSE	FALSE	FALSE
opencv4android	FALSE	FALSE	FALSE
paper-android	FALSE	FALSE	FALSE
parse-android-sdk	FALSE	FALSE	FALSE
parse-sdk-android	FALSE	FALSE	FALSE
pocketsphinx-android	FALSE	FALSE	FALSE
qandroidjniobject	FALSE	FALSE	FALSE
qtandroidextras	FALSE	FALSE	FALSE
quickblox-android	FALSE	FALSE	FALSE
react-native-android	FALSE	FALSE	FALSE
rx-android	FALSE	FALSE	FALSE
rxandroidble	FALSE	FALSE	FALSE
sbt-android-plugin	FALSE	FALSE	FALSE
spring-android	FALSE	FALSE	FALSE
sqlcipher-android	FALSE	FALSE	FALSE
svg-android	FALSE	FALSE	FALSE
titanium-android	FALSE	FALSE	FALSE
turbolinks-android	FALSE	FALSE	FALSE
vimeo-android	FALSE	FALSE	FALSE
wikimedia-android-data-client	FALSE	FALSE	FALSE
xamarin-android-player	FALSE	FALSE	FALSE

Appendix C

Replication Data for Casdoc

We published the data necessary to verify and replicate our laboratory study of Casdoc (Section 5.5) at https://zenodo.org/records/10637079 [161]. Table C.1 details the content of the artifact. The last column indicates which parts of the artifact is reproduced in the sections of this appendix.

Table C.1: Content of the Data Artifact for Our Laboratory Study of Casdoc

Path	Description	Sec.
README.md	Description of the artifact.	
study-replication/documentation/	Folder containing the content of the static website available to participants during their session.	
study-replication/documentation-sources.txt	List of online resources used for the creation of the documents provided to participants.	C.1
study-replication/tasks/	Folder containing the programming environment for the tasks. The folder is a Git repository that participants can clone. The task instructions start on line 118 of the file src/task/JdbcTask.java in this folder.	C.2
study-replication/sample-solutions/JdbcTask.java	Reference solution for the tasks. Participants could implement alternative, functionally-equivalent solutions to complete the tasks.	C.3
study-replication/intervention-guide.md	Intervention guidelines for the investigator during the study sessions.	C.4
analysis-data/phase1-session-events/	Folder containing the results of the first phase of manual annotation of the sessions' recordings. The folder contains one file per participant.	
analysis-data/phase2-search-fragments.tsv	Results of the second phase of manual annotation.	
analysis-data/phase3-navigation-patterns.tsv	Results of the third phase of manual annotation.	
analysis-data/coding-guide.md	Guidelines for each phase of data annotation.	C.5

C.1 Documentation Resources About the JDBC API

Content of study-replication/documentation *from* https://zenodo.org/records/10637079

- https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html
- https://www.tutorialspoint.com/sqlite/sqlite_java.htm
- https://www.sqlitetutorial.net/sqlite-java/sqlite-jdbc-driver/
- https://www.tutorialspoint.com/jdbc/index.htm
- https://en.wikipedia.org/wiki/SQL_injection
- https://www.w3schools.com/sql/sql_constraints.asp

C.2 Programming Tasks for the Laboratory Study

Content of study-replication/tasks/src/task/JdbcTask.java *from* https://zenodo.org/records/10637079

```
package task;

import java.nio.file.*;
import java.sql.*;
import java.util.*;

import ca.mcgill.cs.mnassif.taskutil.TaskUtil;
import task.JdbcTask.*;

@SuppressWarnings("unused")
public class JdbcTask {

    private static final List<Item> SPECIAL_ITEMS = Arrays.asList(new Item(101, "(;)\";’;;’;));"),
    new Item(102, "\"\"’’;;’&;."),
    new Item(103, ""),
    new Item(104, ";\n;’’% -- \"\" /* ;;"));
    private static final List<Item> ITEM_GROUP_FRUITS = Arrays.asList(new Item(201, "strawberry"),
    new Item(202, "mango"),
    new Item(203, "banana"),
    new Item(204, "passion fruit"));
    private static final List<Item> ITEM_GROUP_DAIRIES = Arrays.asList(
    new Item(301, "chocolate milk"),
    new Item(302, "heavy cream"),
    new Item(303, "cheese"),
    new Item(304, "cream cheese"));
    private static final List<Item> ITEM_GROUP_MEATS = Arrays.asList(new Item(401, "duck"),
    new Item(402, "turkey"),
    new Item(403, "pork"),
    new Item(404, "bison"),
    new Item(405, "moose"),
    new Item(406, null));

    private static Connection staffConnection;

    public static void main(String[] args) throws Exception {
        TaskUtil.resetDatabases();
        try (Connection connection = TaskUtil.getFoodConnection()) {
            System.out.println("----- Objective 1 -----");
            int totalStock = objective1(connection);
            System.out.println("Total inventory stock: " + totalStock + " / 18000");
            System.out.println();

            System.out.println("----- Objective 2 -----");
            System.out.println("Content of Food table:");
            objective2(connection);
            System.out.println();

            System.out.println("----- Objective 3 -----");
            int breadId = objective3(connection);
            System.out.println("ID for the bread: " + breadId);
            System.out.println();

            System.out.println("----- Objective 4 -----");
            objective4(connection, SPECIAL_ITEMS);
            System.out.println(TaskUtil.checkInsertedItems(connection, "food", SPECIAL_ITEMS));
            System.out.println();

            System.out.println("----- Objective 5 -----");
            objective5(connection);
            if (TaskUtil.isValidConnection(connection, "people")) {
                System.out.println("Table successfully created");
                System.out.println("See ’FINAL CONTENT’ to check that the 2 rows were correctly inserted");
            }
            System.out.println();
```

```
        System.out.println("----- Objective 6 -----");
        objective6(connection, ITEM_GROUP_FRUITS); // should fail
        objective6(connection, ITEM_GROUP_DAIRIES); // should succeed
        objective6(connection, ITEM_GROUP_MEATS); // should fail
        System.out.println(TaskUtil.checkInsertedItems(connection, "food", ITEM_GROUP_FRUITS));
        System.out.println(TaskUtil.checkInsertedItems(connection, "food", ITEM_GROUP_DAIRIES));
        System.out.println(TaskUtil.checkInsertedItems(connection, "food", ITEM_GROUP_MEATS));
        System.out.println();

        System.out.println("----- Objective 7 -----");
        staffConnection = objective7();
        if (TaskUtil.isValidConnection(staffConnection, "staff")) {
            System.out.println("Valid connection established");
        }
        else {
            System.out.println("Invalid connection");
        }
        System.out.println();

        System.out.println("----- Objective 8 -----");
        objective8(staffConnection);
        if (TaskUtil.isStaffUpdated(staffConnection)) {
            System.out.println("Update successful");
        }
        System.out.println();

        System.out.println("---- FINAL CONTENT ----");
        TaskUtil.listTables(connection);
        TaskUtil.listTables(staffConnection);
    }
}

// Represents a food item to add to the database
public static class Item {

    private final int id;
    private final String name;

    public Item(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }
}

/*
* Interacting with a SQLite database with JDBC
* ============================================
*
* Implement the methods below to achieve the objective stated in the block comments,
* related to interacting with a SQLite database file using the JDBC API.
*
* You can:
* - write new helper code outside the methods;
* - run the main method as often as you want to check if an objective is complete;
* - look at the content of the database by running the main method.
*
* You can't:
* - change the signature (name, parameters, etc.) of the objectiveX() methods;
* - modify the code above this comment.
*
* Other important notes:
* - You don't need to worry about exception handling or leaked resources:
*       write only the code to handle a normal scenario, and assume the database is well-formed.
```

```
* - The quality of the code you write is not important:
*       just write code that solves the objective, and move on to the next.
* - There are subtle complexities to the objectives:
*       you will likely need to read about some corner cases in the provided documentation.
*/

public static int objective1(Connection connection) throws Exception {
    /*
    * Fetch the content of the "food" table, which has 3 columns:
    * - `id`: integer;
    * - `name`: string;
    * - `stock`: integer, may be missing (i.e., NULL).
    *
    * Return the sum of all item stocks in the table.
    * Ignore items with missing stock values.
    *
    * Use the Connection object provided as argument to this method
    * to complete this objective.
    */
    return 0;
}

public static void objective2(Connection connection) throws Exception {
    /*
    * Print each row of the "food" table in the console on a separate line,
    * separating values with tabs ("\t").
    * If the `stock` value is missing, leave it blank (i.e., not "0").
    *
    * The top 3 lines should look like:
    * 15    mango   4500
    * 19    peach
    * 27    milk    10000
    */
}

public static int objective3(Connection connection) throws Exception {
    /*
    * Insert a new food item named "bread".
    * The ID for this category should be automatically generated by the database.
    * The `stock` value should be missing (i.e., NULL).
    *
    * Then, return the automatically generated ID.
    */
    return -1;
}

public static void objective4(Connection connection, List<Item> items) throws Exception {
    /*
    * Insert in the "food" table all items indicated by the second argument.
    * The `id` and `name` of the items are provided by the Item objects.
    * The `stock` value should be missing (i.e., NULL) for all items.
    *
    * Assume the list of items comes from an unknown source. You should write the
    * code to prevent SQL injection attacks.
    *
    * Note: The Item class is declared at the top of this file (lines 99 to 116).
    */
}

public static void objective5(Connection connection) throws Exception {
    /*
    * Create a new table named "people" with the following columns and constraints:
    * 1. `id`: integer
    *       - automatically generated,
    *       - unique,
    *       - always present,
    *       - the primary key of the table.
    * 2. `name`: string
    *       - unique,
    *       - always present
    * 3. `fav_food`: integer
```

```
        *       - may be missing
        *
        * After creating the table, insert two rows:
        * (autogenerated ID, "Mathieu", 15)
        * (autogenerated ID, "Martin", NULL)
        */
    }

    public static void objective6(Connection connection, List<Item> items) throws Exception {
        /*
        * Insert new items in the "food" table the same way as for objective 4.
        *
        * However, perform all insertions in a single transaction.
        * If some items can't be inserted (e.g., because they violate a constraint
        * of one of the columns), roll back the transaction, so that none of the items
        * are inserted.
        */
    }

    public static Connection objective7() throws Exception {
        /*
        * Create a connection to the SQLite database contained in the file res/staff.db
        * (i.e., the file "staff.db" located in the "res" folder), and return this connection.
        *
        * The necessary libraries containing the SQLite JDBC driver version 4.0 are already
        * on the classpath. You only need to write code in this method to finish the objective.
        *
        * The database has the following properties:
        *     database vendor: SQLite
        *     driver version: 4.0
        *     file path: res/staff.db
        *     username/password: none required
        */
        return null;
    }

    public static void objective8(Connection staffDbConnection) throws Exception {
        /*
        * The database contained in staff.db contains a table named "staff" with the
        * following columns:
        * - `id`: integer;
        * - `name`: string;
        * - `role`: string, may be missing (NULL).
        *
        * Update this table to replace each missing value in the `role` column by the
        * value `regular`. Do not modify other values, or add or remove rows in the table.
        */
    }
}
```

C.3 Reference Solutions to the Programming Tasks

Content of study-replication/sample-solutions/JdbcTasks.java *from* https://zenodo.org/records/10637079

```
package task;

import java.sql.*;
import java.util.*;

import ca.mcgill.cs.mnassif.taskutil.*;

@SuppressWarnings("unused")
public class JdbcTask {

    private static final List<Item> SPECIAL_ITEMS = Arrays.asList(new Item(101, "(;)\";';;';));"),
    new Item(102, "\"\"'';;'&;."),
    new Item(103, ""),
```

```
new Item(104, ";\n;''% -- \"\" /* ;;"));
private static final List<Item> ITEM_GROUP_FRUITS = Arrays.asList(new Item(201, "strawberry"),
new Item(202, "mango"),
new Item(203, "banana"),
new Item(204, "passion fruit"));
private static final List<Item> ITEM_GROUP_DAIRIES = Arrays.asList(
new Item(301, "chocolate milk"),
new Item(302, "heavy cream"),
new Item(303, "cheese"),
new Item(304, "cream cheese"));
private static final List<Item> ITEM_GROUP_MEATS = Arrays.asList(new Item(401, "duck"),
new Item(402, "turkey"),
new Item(403, "pork"),
new Item(404, "bison"),
new Item(405, "moose"),
new Item(406, null));

private static Connection staffConnection;

public static void main(String[] args) throws Exception {
    TaskUtil.resetDatabases();
    try (Connection connection = TaskUtil.getFoodConnection()) {
        System.out.println("----- Objective 1 -----");
        int totalStock = objective1(connection);
        System.out.println("Total inventory stock: " + totalStock + " / 18000");
        System.out.println();

        System.out.println("----- Objective 2 -----");
        System.out.println("Content of Food table:");
        objective2(connection);
        System.out.println();

        System.out.println("----- Objective 3 -----");
        int breadId = objective3(connection);
        System.out.println("ID for the bread: " + breadId);
        System.out.println();

        System.out.println("----- Objective 4 -----");
        objective4(connection, SPECIAL_ITEMS);
        System.out.println(TaskUtil.checkInsertedItems(connection, "food", SPECIAL_ITEMS));
        System.out.println();

        System.out.println("----- Objective 5 -----");
        objective5(connection);
        if (TaskUtil.isValidConnection(connection, "people")) {
            System.out.println("Table successfully created");
            System.out.println(
            "See 'FINAL CONTENT' to check that the 2 rows were correctly inserted");
        }
        System.out.println();

        System.out.println("----- Objective 6 -----");
        objective6(connection, ITEM_GROUP_FRUITS); // should fail
        objective6(connection, ITEM_GROUP_DAIRIES); // should succeed
        objective6(connection, ITEM_GROUP_MEATS); // should fail
        System.out.println(TaskUtil.checkInsertedItems(connection, "food", ITEM_GROUP_FRUITS));
        System.out.println(TaskUtil.checkInsertedItems(connection, "food", ITEM_GROUP_DAIRIES));
        System.out.println(TaskUtil.checkInsertedItems(connection, "food", ITEM_GROUP_MEATS));
        System.out.println();

        System.out.println("----- Objective 7 -----");
        staffConnection = objective7();
        if (TaskUtil.isValidConnection(staffConnection, "staff")) {
            System.out.println("Valid connection established");
        }
        else {
            System.out.println("Invalid connection");
        }
        System.out.println();

        System.out.println("----- Objective 8 -----");
```

```
        objective8(staffConnection);
        if (TaskUtil.isStaffUpdated(staffConnection)) {
            System.out.println("Update successful");
        }
        System.out.println();

        System.out.println("---- FINAL CONTENT ----");
        TaskUtil.listTables(connection);
        TaskUtil.listTables(staffConnection);
    }
}

// Represents a food item to add to the database
public static class Item {

    private final int id;
    private final String name;

    public Item(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }
}

/*
* Interacting with a SQLite database with JDBC
* ============================================
*
* Implement the methods below to achieve the objective stated in the block comments,
* related to interacting with a SQLite database file using the JDBC API.
*
* You can:
* - write new helper code outside the methods;
* - run the main method as often as you want to check if an objective is complete;
* - look at the content of the database by running the main method.
*
* You can't:
* - change the signature (name, parameters, etc.) of the objectiveX() methods;
* - modify the code above this comment.
*
* Other important notes:
* - You don't need to worry about exception handling or leaked resources:
*       write only the code to handle a normal scenario, and assume the database is well-formed.
* - The quality of the code you write is not important:
*       just write code that solves the objective, and move on to the next.
* - There are subtle complexities to the objectives:
*       you will likely need to read about some corner cases in the provided documentation.
*/

public static int objective1(Connection connection) throws Exception {
    /*
    * Fetch the content of the "food" table, which has 3 columns:
    * - `id`: integer;
    * - `name`: string;
    * - `stock`: integer, may be missing (i.e., NULL).
    *
    * Return the sum of all item stocks in the table.
    * Ignore items with missing stock values.
    *
    * Use the Connection object provided as argument to this method
    * to complete this objective.
    */
    try (
```

```
    Statement stmt = connection.createStatement();
    ResultSet rs = stmt
    .executeQuery("SELECT stock FROM food WHERE stock IS NOT NULL;");) {
        int sum = 0;
        while (rs.next()) {
            sum += rs.getInt("stock");
        }
        return sum;
    }
}

public static void objective2(Connection connection) throws Exception {
    /*
    * Print each row of the "food" table in the console on a separate line,
    * separating values with tabs ("\t").
    * If the `stock` value is missing, leave it blank (i.e., not "0").
    *
    * The top 3 lines should look like:
    * 15    mango   4500
    * 19    peach
    * 27    milk    10000
    */
    try (
    Statement stmt = connection.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM food;")) {
        while (rs.next()) {
            int id = rs.getInt("id");
            String name = rs.getString("name");
            int stock = rs.getInt("stock");
            if (rs.wasNull()) {
                System.out.println(id + "\t" + name);
            }
            else {
                System.out.println(id + "\t" + name + "\t" + stock);
            }
        }
    }
}

public static int objective3(Connection connection) throws Exception {
    /*
    * Insert a new food item named "bread".
    * The ID for this category should be automatically generated by the database.
    * The `stock` value should be missing (i.e., NULL).
    *
    * Then, return the automatically generated ID, i.e., the ID of the new row
    * where name = 'bread'.
    */
    try (Statement stmt = connection.createStatement()) {
        stmt.executeUpdate("INSERT INTO food VALUES (NULL, 'bread', NULL);");
        try (ResultSet rs = stmt.executeQuery("SELECT id FROM food WHERE name = 'bread';")) {
            rs.next();
            return rs.getInt(1);
        }
    }
}

public static void objective4(Connection connection, List<Item> items) throws Exception {
    /*
    * Insert in the "food" table all items indicated by the first argument.
    * The `id` and `name` of the items are provided by the Item objects.
    * The `stock` value should be missing (i.e., NULL) for all items.
    *
    * Assume the list of items comes from an unknown source. You should write the
    * code to prevent SQL injection attacks.
    */
    try (
    PreparedStatement stmt = connection
    .prepareStatement("INSERT INTO food VALUES (?, ?, NULL);")) {
        for (Item item : items) {
            stmt.setInt(1, item.getId());
```

```
            stmt.setString(2, item.getName());
            stmt.executeUpdate();
        }
    }
}

public static void objective5(Connection connection) throws Exception {
    /*
    * Create a new table named "people" with the following columns and constraints:
    * 1. ‘id‘: integer
    *       - automatically generated,
    *       - unique,
    *       - always present,
    *       - the primary key of the table.
    * 2. ‘name‘: string
    *       - unique,
    *       - always present
    * 3. ‘fav_food‘: integer
    *       - may be missing
    *
    * After creating the table, insert two rows:
    * (autogenerated ID, "Mathieu", 15)
    * (autogenerated ID, "Martin", NULL)
    */
    try (Statement stmt = connection.createStatement()) {
        stmt.executeUpdate("CREATE TABLE people (id INTEGER PRIMARY KEY, name TEXT UNIQUE NOT NULL, fav_food INT);");
        stmt.executeUpdate("INSERT INTO people VALUES (NULL, 'Mathieu', 15);");
        stmt.executeUpdate("INSERT INTO people VALUES (NULL, 'Martin', NULL);");
    }
}

public static void objective6(Connection connection, List<Item> items) throws Exception {
    /*
    * Insert new items in the "food" table the same way as for objective 4.
    *
    * However, perform all insertions in a single transaction.
    * If some items can't be inserted (e.g., because they violate a constraint
    * of one of the columns), roll back the transaction, so that none of the items
    * are inserted.
    */
    connection.setAutoCommit(false);
    try {
        objective4(connection, items);
        connection.commit();
    }
    catch (SQLException e) {
        connection.rollback();
    }
}

public static Connection objective7() throws Exception {
    /*
    * Create a connection to the SQLite database contained in the file res/staff.db
    * (i.e., the file "staff.db" located in the "res" folder), and return this connection.
    *
    * The necessary libraries containing the SQLite JDBC driver version 4.0 are already
    * on the classpath. You only need to write code in this method to finish the objective.
    *
    * The database has the following properties:
    *     database vendor: SQLite
    *     driver version: 4.0
    *     file path: res/staff.db
    *     username/password: none required
    */
    return DriverManager.getConnection("jdbc:sqlite:res/staff.db");
}

public static void objective8(Connection staffDbConnection) throws Exception {
    /*
    * The database contained in staff.db contains a table named "staff" with the
    * following columns:
```

```
        * - `id`: integer;
        * - `name`: string;
        * - `role`: string, may be missing (NULL).
        *
        * Update this table to replace each missing value in the `role` column by the
        * value `regular`. Do not modify other values, or add or remove rows in the table.
        */
        try (Statement stmt = staffDbConnection.createStatement()) {
            stmt.executeUpdate("UPDATE staff SET role = 'regular' WHERE role IS NULL;");
        }
    }
}
```

C.4 Intervention Guide for the Investigator During the Study Sessions

Content of study-replication/intervention-guide.md *from* https://zenodo.org/records/10637079

The investigator should help the participant in the following situations:

- to confirm the success of a task if the participant is unsure;
- to reveal semantic errors in the current solution if the participant is unaware of them and moves on to the next task;
- to provide technical information about Java syntax elements (e.g., the syntax of a for loop);
- to help the participant progress after:
  - they spend more than 3 minutes stuck on an issue with no visible strategy to progress;
  - they spend more than 5 minutes looking for the same information without progress;
  - they spend more than 15 minutes on the same task;
- to notify the participant that 40 minutes have elapsed;
- to answer questions asked by the participant about the tasks' environment and requirements (decline answering questions about technical aspects of the tasks).

C.5 Annotation Guide for the Session Recordings

Content of analysis-data/coding-guide.md *from* https://zenodo.org/records/10637079

Phase 1

Identify each of the following events during the programming sessions. Indicate the time of the video recording (in minutes and seconds) at which the even occurred or started. Indicate in the Details column further details about the event when useful or required by the event. All event definitions are described from the perspective of the participant (e.g., "read the instructions" denotes that the participant is reading the instructions).

- **Task:** Events related to the progression on the tasks.
  - instr-N: Read the instruction for task N. Use instr-0 for the introductory instructions. Generally indicates the start of a task.
  - code: Start writing code towards a solution or continue writing code after searching for information or testing a previous solution.
  - paste: Paste code copied from the documentation. Use in the same context as and as a replacement for code, but when participants start from copied code. test: Execute the program to verify their current solution.
- **Window:** Events related to the documents seen by participants. Ignore spurious events, e.g., where a participant cycles through multiple opened tabs before finding the right one.
  - main: Open or look at the study website's home page (index.html).
  - ide: Open or look at their IDE.
  - establish-connection, read-values, etc.: Open or look at the provided document. Use the document's file name, visible in the browser's URL bar, as the code.
  - ResultSet, Statement, etc.: Open or look at the API reference documentation on Oracle's website. Use the name of the type (e.g., ResultSet) or package (e.g., java.sql) as the code. Use javadoc:root to denote the landing page of the reference documentation.
  - side-by-side: Place the IDE and web browser side by side. In addition to this code, keep tracking which window is looked at with other codes.
  - single: Expand a single window to full screen. Opposite of side-by-side, all participants are assumed to start in this configuration, unless coded otherwise. Also use this code if one window takes most of the screen, with other windows partly visibles.
- **Search:** Events related to information searches.
  - familiar: Start looking through the documents to get familiar with a general topic.

  - search: Start searching for information. Do not code information searches that are only within the IDE.
  - quick: Start a short interaction with the documentation, e.g., to confirm an intuition, re-find information, or revisit the result of the previous search while implementing the solution. Mutually exclusive with search.
  - found: Find information that ends the current search or changes the search target. Use this code to denote events that visibly affect the participant's search progress.
  - solved: End the search after finding sufficient information to satisfy the original query.
  - copy: End the search by copying code that satisfies the original query. Alternative to solved.
  - drop: Abandon the search without finding sufficient information.
  - stopped: End the search due to an intervention by the investigator.

- **Component:** Events related to the usage of different components of a document.
  - code: Read the code example at the top of a document. Do not use this code when reading other code fragments in the Casdoc annotations or in the expanded text.
  - text: Read or scan the expanded text below the top code example.
  - casdoc: Use a feature of the Casdoc formt, such as popovers, dialogs, and the search bar. Indicate in Details which element is used.
  - source: Look for information in the source code of the task file, in the IDE. Only use this code during information searches.
  - other: Other events, such as using the web browser's native search tool. Describe in Details the nature of the event.

- **Intervention:** Events related to interactions between the participant and the investigator.
  - qa: Ask a question to the investigator.
  - help: Receive unprompted help by the investigator.
  - process: Receive unprompted instructions from the investigator related to the session process.
  - end: End the current interaction with the investigator.

Phase 2

Start by identifying search fragments within each programming session. A search fragment is a continuous period during which the participant searches for some information without making progress in their current task. Consider the following constraints when identifying search fragments:

- A search fragment should require the participant to *search* for information. If the participant only looks in document to read information they had already located, it does not constitute a search fragment.
- The information sought during a single fragment can change. The participant does not need to search for the same information throughout the fragment.
- A search fragment should involve at least one web document. Searches performed only within the IDE should be excluded.
- The participant can alternate between reading web documents and looking at the IDE during the same search fragment.
- When trying to debug an error in the code, the participant may alternate between searching web documents and writing (and executing) code. This behavior constitutes a single search fragment as long as there is no progression in the task (i.e., attempted fixes fail and the participant does not change new parts of the code).

After identifying each search fragment, extract the following properties:

- ID: a sequential ID, unique among all participants;
- Participant: the ID of the participant involved;
- Task: the task that the participant is currently working on;
- Start: the time at which the search fragment starts, relative to the video recording (in minutes and seconds);
- End: the time at which the search fragment ends;
- Documents: the web document(s) in which the participant looked for information during the search fragment (exclude documents *seen* but not *used* by the participant);
- Intention: the intention of the search fragment, selected among the values below;
  - *familiar*: the participant is trying to become more familiar with a concept or the abstraction represented by a JDBC class, without a specific information need about that concept or class.

  - *general solution*: the participant is trying to find an initial implementation that can be adapted to solve the task.
  - *targeted*: the participant is looking for a specific piece of information about some aspect of the task.
  - *debug*: the participant is trying to find the cause of an error they are getting.

- Components: the list of documentation components used during the search, selected among the values below;
  - *code*: the code example at the top of each document.
  - *popover*: an annotation in their temporary form (exclude unintentional popovers).
  - *dialog*: an annotation in their pinned form (exclude unintentional dialogs).
  - *popover**: an unintentional popover that distracts the participant.
  - *search*: the Casdoc search bar in the top right corner.
  - *text*: any part of the expanded text below the top code example.
  - *browser-search*: the web browser's native search tool.
- Order: the order in which the components identified in the previous column were used (see below for the format);
- Info (Casdoc): the type(s) of information sought when using a Casdoc-related component (see below for the possible values);
- Info (Text): the type(s) of information sought when using a Text-related component (see below for the possible values).

Order Values and Format

- List the sequence of components in the order they were used, separated by an arrow (->).
- Repeat components as necessary.
- Include transitions to and from the IDE using the code ide.
- If a participant alternates between two components, use a bidirectional arrow (<>). For example, code <> popover is equivalent to code -> popover -> code -> popover -> ....
- Consider the following variation to the previous rule: If one of the components acts as a secondary source of information to support the other component, show the secondary component in square brackets. For example, code [<> popover] indicates that the participant relied mainly on the code, but occasionally used popovers.

- If the participant used multiple documents, use one sequence per document, and separate sequences with a semi-colon (;).
- Start each sequence with either new if this is the first time the participant uses the document, or known if the participant already used the document. If the document is an API reference documentation page, start the sequence with javadoc, regardless of prior activity.
- End a sequence with link if the participant clicked on a hyperlink to navigate to the next document.

Information types

- *how to*: information about how to perform a specific operation.
- *error cause*: information that may help the participant understand what causes an error in their code.
- *element detail*: information about a specific API element.
- *code example explanation*: information about the design, context, or implementation choices of the code example at the top of the document.
- *SQL syntax*: information about the syntax of the SQL language or about SQL functions and operators.
- *concept*: information about a concept, the abstraction represented by an API class, or the domain of the tasks.
- *content overview*: overview of the topics addressed by a document, either to get familiar with the document or when checking whether the document contains any more information about the current search.
- Special cases
  - *link*: use this code to indicate that the component was only used to access a hyperlink (only applicable for Casdoc annotations).
  - *continue X*: prepend the type with *continue* when a participant searched for the information in another format first without success, and continues their search in a different format.
  - *confirm X*: prepend the type with *confirm* when a participant found the information in another format first, but continues searching in a different to confirm their previous finding.

  - *X (Y)*: indicate a secondary type in parentheses when the participant searches for some type of information (X) as part of a broader information search for a different type (Y).
  - *confirm content overview*: this special case indicates that the participant found some information in one format, then looked for an overview of the document in the other format to confirm that there was no more information related to their query.

Phase 3

For each search fragment (identified by its ID), indicate the presence of the following navigation patterns by an x (or, when indicated, by the number of occurrences). Leave the column empty if the pattern did not occur. The first level of the list, shown here for readability, is not represented in the data file. The columns in the data file are in the same order as the second level presented here.

- RQ1.1, Format Used: each fragment falls in exactly one of those categories
  - code only: the participant did not use either the Casdoc or the expanded text formats.
  - casdoc: the participant used the Casdoc format but not the expanded text format.
  - text: the participant used the expanded text format but not the Casdoc format.
  - casdoc + text: the participant used both the Casdoc and the expanded text formats.
- RQ1.2a, Information Types for the Casdoc Format: for each column, mark an occurrence only if the Casdoc format was used as the first format for each type (i.e., excluding *continue* and *confirm*) and if the type was the primary (i.e., excluding types in parentheses)
  - how to
  - element detail
  - error cause
  - code example explanation
  - SQL syntax
  - concept
  - link
- RQ1.2b, Information Types for the Expanded Format: follow the same rules as the previous columns
  - how to
  - element detail

  - error cause
  - code example explanation
  - SQL syntax
  - concept

- RQ1.3, Second Format: indicate whether the participant continued a search started in a different format (i.e., the Info columns in the second phase contains a *continue X* code)
  - continue text: use the expanded text format to continue a search started in the Casdoc format.
  - continue annotations: use Casdoc annotations (without the search bar) to continue a search started in the expanded text format.
  - continue search: use the Casdoc search bar to continue a search started in the expanded text format (the participant can reveal annotations found with the search).

- RQ1.4, Confirmation: indicate whether the participant confirmed a search started in a different format (i.e., the Info columns in the second phase contains a *confirm X* code)
  - confirm text
  - confirm casdoc

- RQ2.1: Code Usage: indicate reliance on the code example at the top of each document
  - code only: the participant only used the code example (same as the first column).
  - no code: the participant did not use the code example at all during the entire search fragment.
  - new -> code: the participant first looked at the code example when opening a new document (count the number of occurrences within the search fragment).
  - new -> other: the participant first looked at any component other than the code example when opening a new document (count the number of occurrences within the search fragment).

- RQ2.2, Content Overview: indicate instances of participant looking for an overview of a document's content
  - content overview (primary): *content overview* as the primary information type in either of the Info columns from the second phase (include instances of *confirm content overview*).

  - content overview (secondary): *content overview* as the secondary information type (i.e., coded as "*(content overview)*") in either of the Info columns from the second phase.

- RQ2.3, Accidental Popovers: indicate instances of distracting popovers
  - popover*: instance of the code *popover** in the Components column from the second phase