Using Disambiguated Word-embeddings for Exploiting Similarities among Languages for Machine Translation

Harsh Satija

***Abstract*—Neural distributed word representations have proven useful in Natural Language Processing and serve as the core underlying method for many NLP tasks. In this work we try to eliminate one of the shortcoming of word representation, by introducing disambiguated word representations and then extend them to build better translation dictionaries by exploiting the similarities between word representations of different languages. We experiment with different mapping functions (linear and non-linear) for converting the word representations from one language to another. We test our approach by translating words from French to English. Our work is the direct extension of Rocher et al. [1], Mikolov et al. [2] and Trask et al. [3]. Our method makes little assumption about the languages, and can be extended and improving translation tables for any language pairs.**

I. Introduction

The recent advancement in deep learning methods have lead to better neural vector space models which project the word semantic and syntactic information into a fixed dimensional space. Various form of distributed representations have been shown to be used in different NLP tasks like Named Entity Recognition, Machine Translation, Natural language Generation models ([22],[23]). However one shortcoming of these word representations is that they generate the same representation of a word even when it is being used in different context. For example,a word-vector model trained on Wikipedia data, the most similar words vectors to word "Apple" are [Microsoft, iPhone, iPad, orange, Steve Jobs, pear]. While evaluation the model disregards the context in which the word Apple appears and therefore give the same representation either its being used in context to fruits or technology. We take ideas from work by Trask et al. [3] to find the contextual representations of the word by disambiguation on the basis of Parts-Of-Speech tags. For the above example, Apple when tagged as Common Noun will lead to representation similar to fruits and when tagged as Proper Noun will refer to representation similar to technology.

Previous studies [1] have shown that there exists geometric relationship between word representations between different languages and they can be used to find similar words when mapped in a same dimensional space. However, we extend the work by Mikolov et al. [2] to test our new representations results against the traditional word-vector approach. In the above work the authors try to infer missing entries in word-translation dictionary using distributed representations of words and phrases. They achieve that by learning linear projection between vector spaces that represent each language, but in our work we try with both linear and non-linear projections. The method consists of two steps. First, we build monolingual word-vector models of languages using large amounts of text. Next, we use a small bilingual data to learn a linear projection between the languages. At the test time, we can translate any word that has been seen in the monolingual corpora by projecting its vector representation from the source language space to the target language space using the mapping we obtained. Once we obtain the vector in the target language space, we output the most similar word vector as the translation.

An intuitive explanation of why this method works can be seen from Figure 1 where word vectors are projected to two dimensional space using PCA. It can be seen that these concepts have similar geometric arrangements in both spaces, suggesting that it is possible to learn an accurate linear mapping from one space to another. This is the key idea behind our method of translation. A more extensive study on why the word representations are similar in both spaces has been studied by [4].

II. Related Work

For distributed representations we refer to work by Mikolov et al. [5], word2vec, where the authors proposed two simple methods for learning continuous word embeddings using neural networks based on Skip-gram or Continuous-Bag-of-Word (CBOW). Word vectors built from these methods map words to points in space that effectively encode semantic and syntactic information and also showed linear relations such as, vector(France) - vector(Paris) is similar to vector(Italy) - vector(Rome).

For disambiguated word vectors we refer to work by Trask et al. [3], where authors represented sense vectors, sense2vec, by using various features such as Part-of-Speech (POS) tags, Sentiment, Named-Entity tags, but in our work we only use POS tags for context resolution.

We then extend our work to Mikolov et al. [2], to compare the performance of both systems. Our system is built in similar fashion as above author's system, however we also experiment with non-linear mappings along with using the disambiguated vectors.

III. Data Sets and Preprocessing

For our experiments we used data sets from WMT13 translation task [7], in particular the News Commentary data

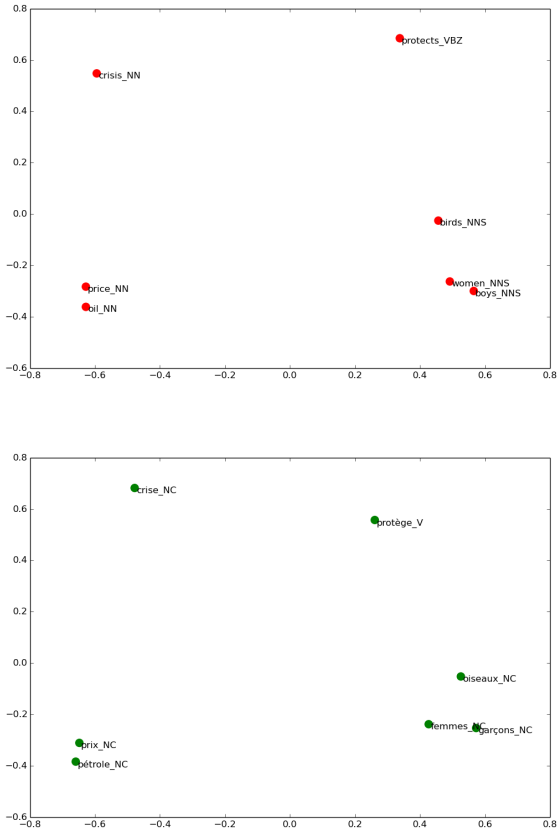

Fig. 1. Distributed word vector representations of various in English (top) and French (bottom). The word vectors in each language were projected down to two dimensions using PCA. Note that the relative positions of the words are same in both spaces.

set for monolingual training phase. The details of the corpora are listed in table I. The following pre-processing steps were applied to the dataset:

- Tokenization of text using the scripts from www.statmt.org [6].
- Converting text to lowercase.
- Removing special characters such as !"#$&'()+.,-;¡=¿*/:?@[]{}

The dataset was then tagged using Stanford Parts-Of-Speech tagger [8] to add POS tags to the the data, after which every word got appended with a POS tag after '_' deilimeter. The POS tags used for English are defined by Penn Treebank [11], and for French are defined by french Treebank [10]. Therefore, a word such as "sell" in the corpora became either of "sell_NNS", "sell_VBP" or "sell_VB".

To obtain dictionaries between languages, we used the top 8000 most frequent words from the monolingual source datasets (stop-words were excluded using NLTK's [12] stop-word corpora for both english and french), and translated these words using on-line translator service, Bing Translator [13]. As not all words that translator produces are in vocabularies, we took the intersection of both the sets and finally got a dictionary of size 5100 words. From the dictionary, 4100 words were kept aside for learning the mapping function and the remaining 1000 words were used as test set for measuring the results.

TABLE I
SIZES OF MONOLINGUAL TRAINING DATASETS FROM WMT13.

Language	Training tokens	Vocabulary size
English	3535313	45055
French	4086635	69129

IV. METHODOLOGY

A. Word2Vec

We use skig-gram model of word2vec as suggested by [2] for distributed representations as follows:

1) Introduction: Given a sequence of training words $w_1, w_2, w_3, ..., w_T$, the objective of the Skip-gram model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}[\sum_{j=-k}^{k} logp(w_{t+j}|w_t)]$$

where k is the size of the training window , which is a function of the center word w_t. The inner summation goes from $-k$ to k to compute the log probability of correctly predicting the word w_{t+j} given the word in the middle wt. The outer summation goes over all words in the training corpus.

In the Skip-gram model, every word w is associated with two learnable parameter vectors, u_w and v_w. They are the "input" and "output" vectors of the w respectively. The probability of correctly predicting the word w_i given the word w_j is defined as

$$P(w_i = w_i) = \frac{\exp(u_{w_i}^T v_{w_j})}{\sum_{l=1}^{V} \exp(u_l^T v_j)}$$

where V is the number of words in the vocabulary. [2] [5] [15]

The Skip-gram models are typically trained using stochastic gradient descent. The gradient is computed using backpropagation rule [14].

2) Implementation: We used Gensim [16] module for training the word2vec model. The parameters used for training were:

- num_features = for french 600, for english 300
- min_word_count = 5
- context = 10
- downsampling = 0.001

B. Mapping Function

We are given a N pairs of type $\{x_i, y_i\}$, where $x_i \in \mathcal{R}^{in}$ is distributed representation of word i in source language and $y_i \in \mathcal{R}^{out}$ is distributed representation of word i in target language, both having different dimensions. Our goal is to find a function $func$ which approximately maps x_i to y_i, such that:

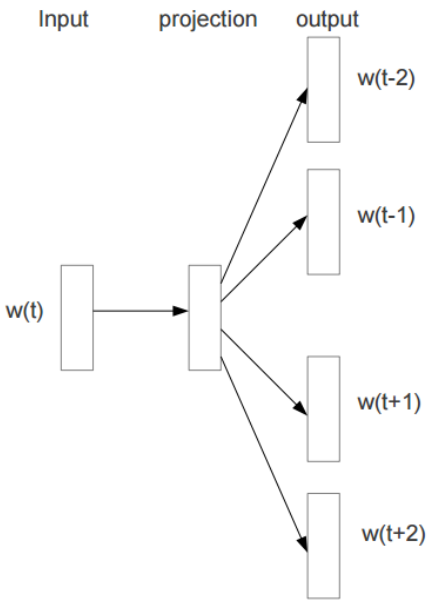

Fig. 2. The Skip-gram model architecture. The training objective is to learn word vector representations that are good at predicting the nearby words.

$$func(\mathcal{R}^{in}) \rightarrow \mathcal{R}^{out}$$

We experiment with two ways of getting an mapping function from the training data :

1) Translation Matrix - Linear Function: In this section we try to find a linear function which maps the vectors from one space to other. More formally, our goal to find a transformation matrix W such that Wx_i approximates y_i. In practice, W can be learned by the following optimization problem

$$\min \sum_{i=1}^{N} \|Wx_i - y_i\|^2$$

which we solve with stochastic gradient descent. [2]

We implemented the above learning function using Theano [17] to prevent manually deriving gradients for each weight parameter in the matrix W. We also included L2 regularization during the training phase. The training parameters with which we trained are final model are following:

- iterations = 10,000
- learning_rate = 0.01
- L2_regularization = 0.1
- batch_size = 500

2) Neural Net - Non-Linear Function: In this section we use a feed forward neural network (FFNN) in order to get a non-linear approximator. FFNNs are known to be universal function approximators with a single hidden layer, as long as this hidden layer can be arbitrarily large. They also have the flexibility of being able to easily scale to any number of inputs and outputs. These properties make them a reasonable candidate for the task at hand. FFNN are also trained by gradient descent.

Given a row vector of inputs $x^{(i)}$, a weight matrix $W^{(i)}$, and a bias vector $b^{(i)}$, the output of layer i in the FFNN is

$$\sigma(x^{(i)}W^{(i)} + b^{(i)})$$

The error signal for each layer also has a simple formulation:

$$\delta^{(\text{out})} = -(y - \hat{y}) \circ \hat{y} \circ (1 - \hat{y})$$

for the output layer ($\hat{y}$ is the predicted output, $o^{(i)}$ is the output of layer i, (out) represent the output layer), and

$$\delta^{(i)} = \delta^{(i+1)}(W^{(i)})^T \circ o^{(i)} \circ (1 - o^{(i)})$$

Where $\circ$ is the Hadamard product.

We implemented the neural net using Keras [18], which is a lightweight abstraction layer over Theano [17]. The parameters for our best model are as follows:

- hidden_layers = 2 (with number of neurons 500 and 400 respectively)
- activation_function = "tanh", inverse tan
- epochs = 5000
- batch_size = 500

TABLE II
DIFFERENT REPRESENTATIONS OF WORD "FIT"

fit	**VB**	**1.0**	**fit**	**JJ**	**1.0**
belong	VB	0.63	practice	VBP	0.75
wish	VBP	0.63	nutritional	JJ	0.71
matter	VB	0.62	treatable	JJ	0.71

TABLE III
DIFFERENT REPRESENTATIONS OF WORD "CZECH"

czech	**NN**	**1.0**	**czech**	**JJ**	**1.0**
slovakia	NN	0.87	helmut	JJ	0.78
hungary	NN	0.86	gilani	NNS	0.78
romania	NN	0.73	wolfgang	NN	0.77

V. EXPERIMENTS AND RESULTS

We first describe the results of our disambiguated word representations. As we can see in table II, our model has learned different representations for word "fit", when fit occurs as verb then it gives representation similar to belong, wish and matter (the notion of fitting in), and when it appears as adjective its representation is similar to practice, nutrition, treatment (the notion of fitness). Similar conclusions can be made from table III, where word czech as noun represents the country and as adjective czech nationality, and table IV, where word last as verb represents the notion of prediction/wait (as in "how long will it last") whereas as adjective it refers to past notion of position ("he arrived last today").

For words in test set, we can translate the word by projecting its vector representation from the source language space to the target language space using the mapping we obtained during learning. Once we map the vector to the target language space, we output the most similar word vector as the translation.

TABLE IV
DIFFERENT REPRESENTATIONS OF WORD "LAST"

last	VB	1.0	last	JJ	1.0
predict	VB	0.68	earlier	RBR	0.56
happen	VB	0.65	next	JJ	0.53
wait	VB	0.65	ago	RB	0.52

TABLE V
RESULTS, METRIC: TOP5 ACCURACY

Model	Test Set	Train + Test
Translation Matrix (Linear)	16.8	41.76
Neural Network (Non-linear)	7.1	26.42

Since there can be many similar possible candidates, our approach generates many candidate translations.Therefore we report the top-5 accuracy. In the top-5 accuracy metric, a match is considered positive if any of the most 5 similar translations matches the target word. We report the top-5 accuracy on the test set in table V

To better understand the behavior of our translation system, we show a number of example translations from French to English in Table VI.

VI. DISCUSSION

We can check that using POS alone can provide some notion of context and can lead to better contextually aware distributed representations. However, right now our trained word embedding models are dependent on POS tagger systems. Although, Stanford POS tagger has state-of-art accuracy for English but using only this approach, our system can't be extended to languages which have poor POS tagging systems.

From the Table V we can see that the accuracy on held-out dataset is approximately 17% , which seems pretty low on first glance, but it should be noted that 17% accuracy on training corpus of size 3M is highly underestimated. Even in the work by Mikolov et al. [2], the top-5 accuracy of the system for corpus of size 10^7 was around 15%. However, when word2vec model is trained on larger monolingual corpora, of size around 10^{11}, then the same system leads to 80% accuracy. This correlates well with the fact that often these neural embedding models require much larger corpus to outshine the other methods. However, due to the computation limitations of our machine we are unable to replicate the results for larger corpus. Thus, we need to compare results on larger corpus before arriving to any conclusion. Also training error of 60% (accuracy on our train + test data is around 40%) is pretty decent and shows some promise in our approach.

TABLE VI
EXAMPLES OF FRENCH TO ENGLISH TRANSLATIONS. THESE EXAMPLES WERE CHOSEN AT RANDOM FROM 5000 DATASET

French Word	Computed English Translations	Dictionary Word
pays_NC	country_NN countries_NNS regions_NNS	country_NN
politique_ADJ	legitimacy_NN reform_NN politics_NNS	policy_NN
années_NC	years_NNS decades_NNS decade_NN	years_NNS
croissance_NC	growth_NN recovery_NN slowdown_NN	growth_NN
vers_P	ivf_NN tritium_NN progeny_NN	worms_NNS
banques_NC	banks_NNS creditors_NNS loans_NNS	banks_NNS
moyen_ADJ	achieve_VB path_NN commitment_NN	way_NN

VII. FUTURE WORK

Although our method doesn't beats the state-of-art system as discussed in Discussion section, but it still presents some insights which we hope will encourage more research in direction of better context aware distributed representations. Validating our approach on a bigger corpus is the foremost task looking ahead. Apart from that, we also present a few limitations and future directions for our work:

- As we mentioned in discussion section, right now our model is dependent on POS tagger, which doesn't generalize well with other languages. Therefore, we need to find a way to include POS in the embedding itself and learn it using the training phase.
- We also need to explore how can we add other context features like Named-Entity and Sentiment into the embedding itself.
- Recent work in distributed representations using character level features has shown some promising results [19], [20], [21]. It would be interesting to try a combination of character and word based embedding for this task.
- Also learning the representations in the same space might results to better result, so that we don't have a need of an explicit mapping function.

STATEMENT I

We hereby state that all the work presented in this report is that of the author.

REFERENCES

[1] Bilingual Word Embeddings for Phrase-Based Machine Translation, Will Zou, Richard Socher, Daniel Cer and Christopher Manning. Conference on Empirical Methods in Natural Language Processing (EMNLP 2013, Short).

[2] Exploiting Similarities among Languages for Machine Translation,Tomas Mikolov, Quoc V. Le, and Ilya Sutskever, arXiv, 2013 , http://arxiv.org/abs/1309.4168

[3] sense2vec - A Fast and Accurate Method for Word Sense Disambiguation In Neural Word Embeddings, Andrew Trask, Phil Michalak, John Liu http://arxiv.org/abs/1511.06388

[4] http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

[5] Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. CoRR, abs/1301.3781, 2013a. URL http://arxiv.org/abs/ 1301.3781.

[6] http://www.statmt.org/

[7] http://www.statmt.org/wmt13/translation-task.html#download

[8] http://nlp.stanford.edu/software/tagger.shtml

[9] http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html

[10] http://www.llf.cnrs.fr/en/Gens/Abeille/French-Treebank-fr.php

[11] http://www.cis.upenn.edu/~treebank/
[12] http://www.nltk.org/
[13] https://www.bing.com/translator
[14] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. Nature, 323(6088):533–536.
[15] Distributed Representations of Words and Phrases and their Compositionality Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, http://arxiv.org/pdf/1310.4546v1.pdf
[16] https://radimrehurek.com/gensim/models/word2vec.html
[17] http://deeplearning.net/tutorial/mlp.html
[18] http://keras.io/
[19] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), 1746–1751.
[20] Weston, J., & Adams, K. (2014). #TAGSPACE : Semantic Embeddings from Hashtags, 1822–1827.
[21] Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2015). Character-Aware Neural Language Models.
[22] Al-Rfou, Rami, Kulkarni, Vivek, Perozzi, Bryan, and Skiena, Steven. Polyglot-NER: Massive multilingual named entity recognition. Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, British Columbia, Canada, April 30 - May 2, 2015, April 2015.
[23] Chen, Danqi and Manning, Christopher. A fast and accurate dependency parser using neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1082.