COMP 551 – Applied Machine Learning
Lecture 17: Deep Learning (cont'd)

Instructor: Joelle Pineau (*jpineau@cs.mcgill.ca*)

Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Major paradigms for deep learning

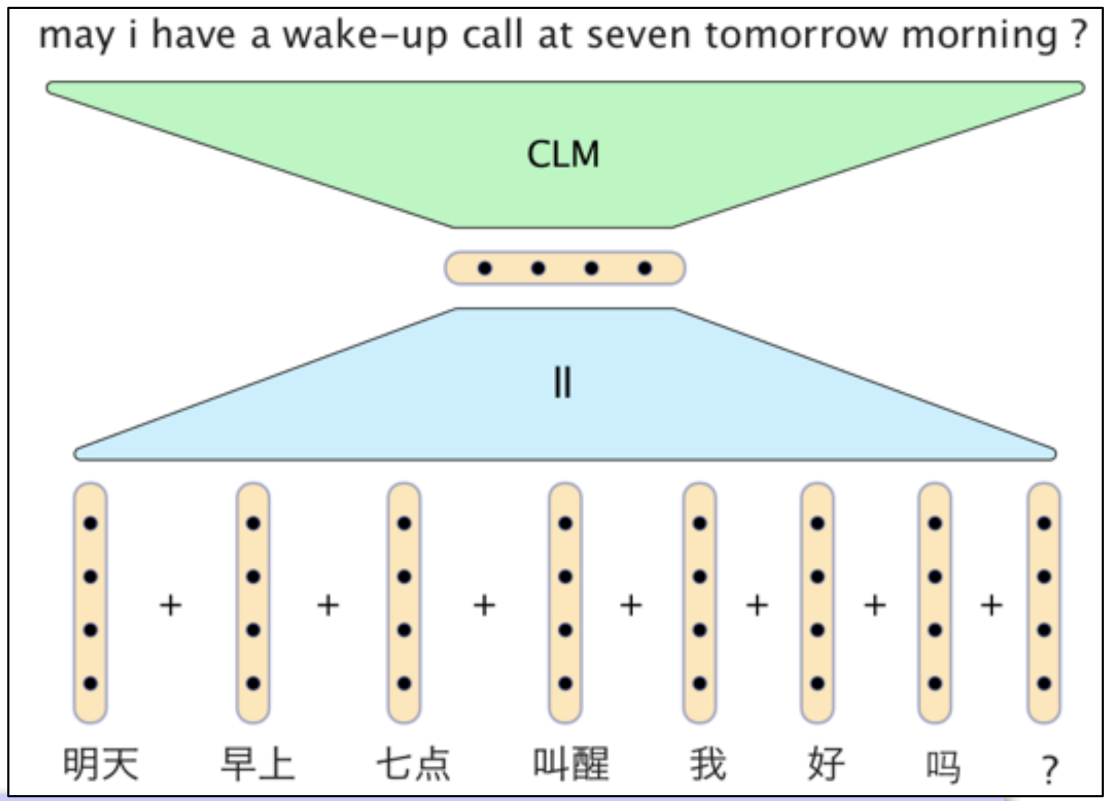
- **Deep neural networks**: The model should be interpreted as a computation graph.
  - Supervised training: Feedforward neural networks.
  - Unsupervised pre-training: Stacked autoencoders.

- Special architectures for different problem domains.
  - Computer vision => Convolutional neural nets.
  - Text and speech => Recurrent neural nets.

Neural models for sequences

- Several datasets contain sequences of data (e.g. time-series, text)
- Bag-of-words assumption looses the ordering information.

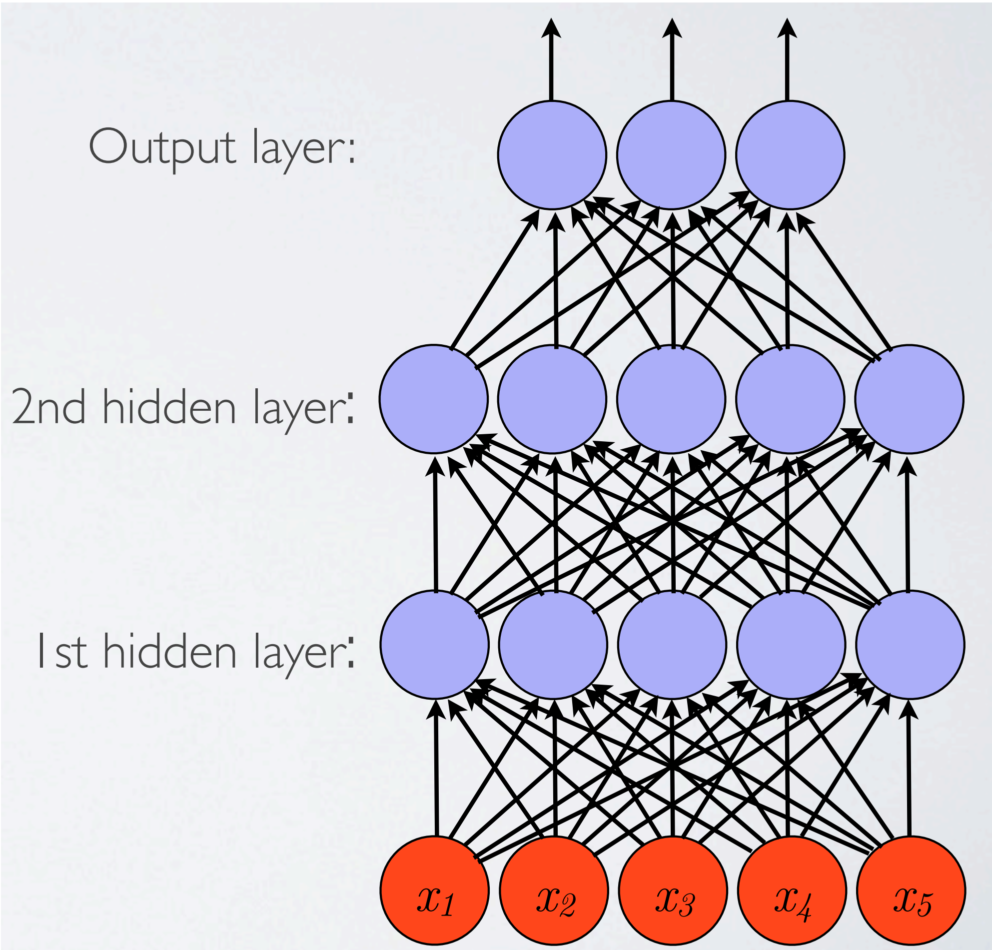
- E.g. Machine translation



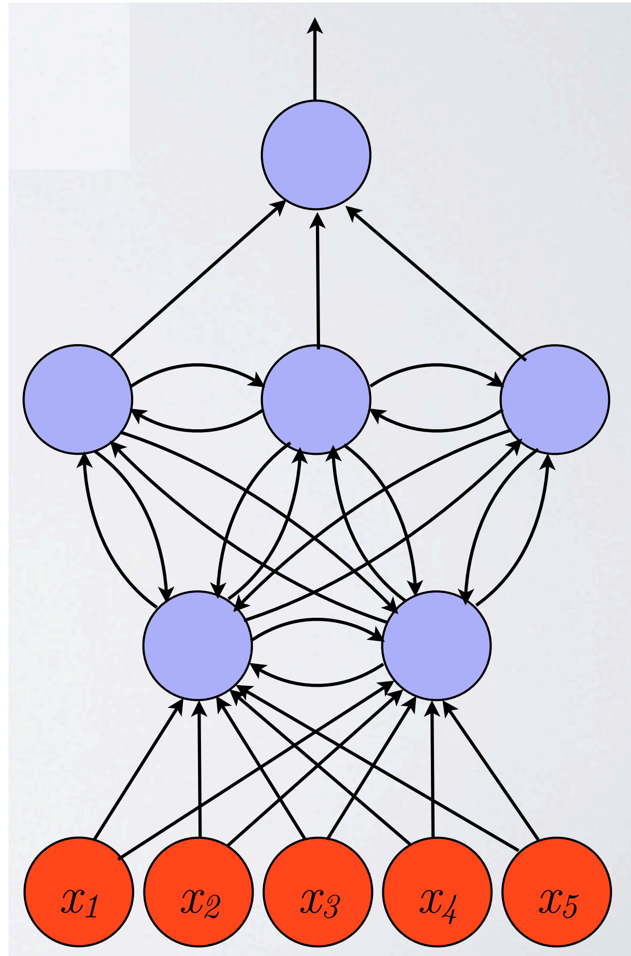
From Phil Blumson's slides:

Recurrent Neural Networks (RNNs)

Feed-forward neural net



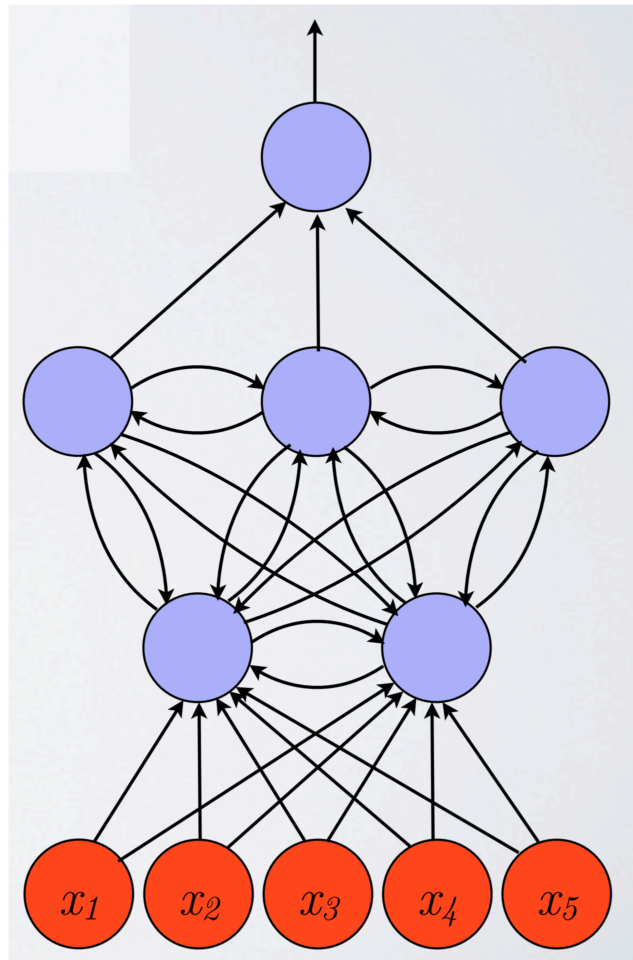
Add cycles in network



Recurrent neural networks (RNNs)

- RNNs can have arbitrary topology.
  - No fixed direction of information flow.
- Delays associated with connections.
  - Every directed cycle contains a delay.
- What can we represent with cycles?

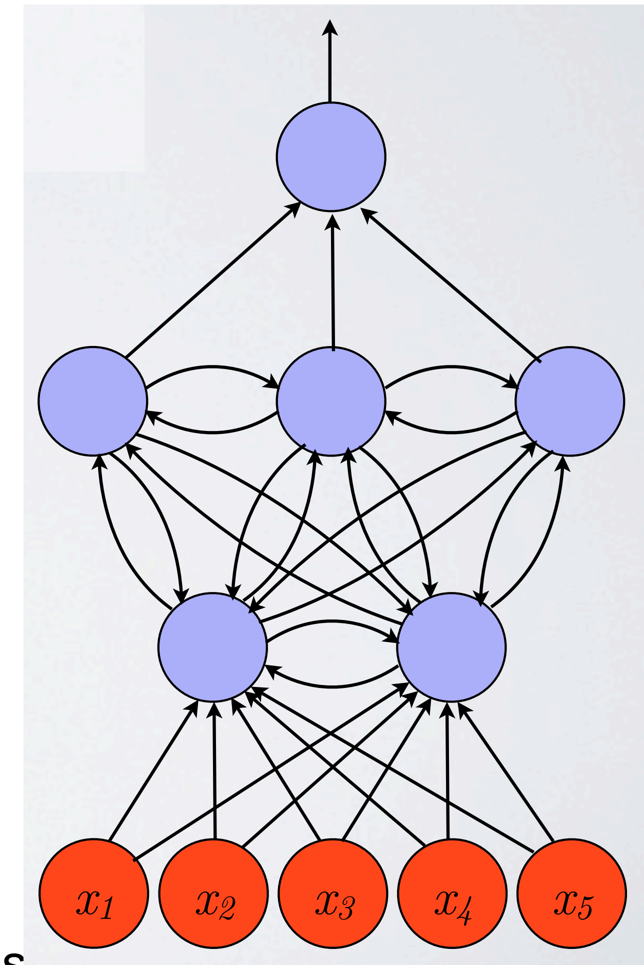
Add cycles in network



Recurrent neural networks (RNNs)

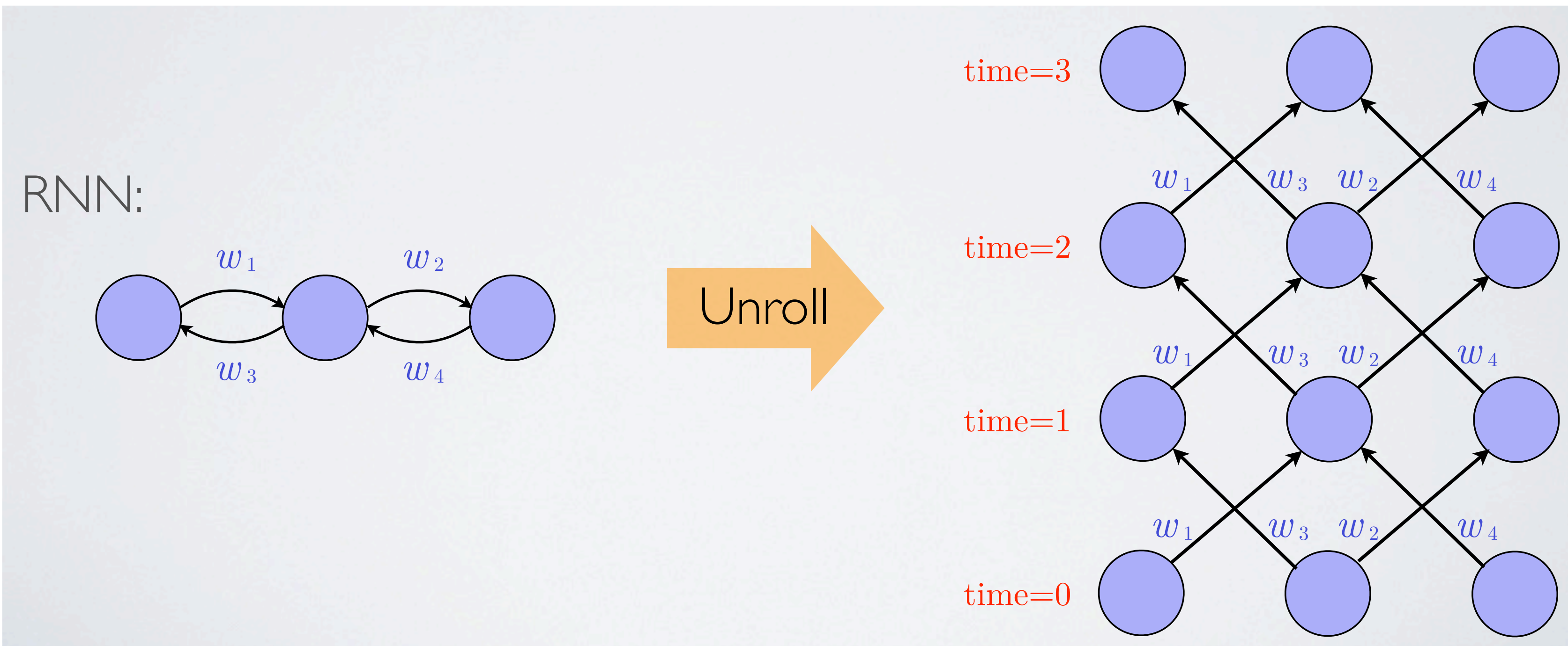
- RNNs can have arbitrary topology.
  - No fixed direction of information flow.
- Delays associated with connections.
  - Every directed cycle contains a delay.
- What can we represent with cycles?
  - Store an internal dynamic state.
  - Summarize/encode sequences, time-series.
  - Can capture oscillatory patterns.
  - Can ignore some portion of sequence.
  - Hard: Sequences with long dependencies.

Add cycles in network



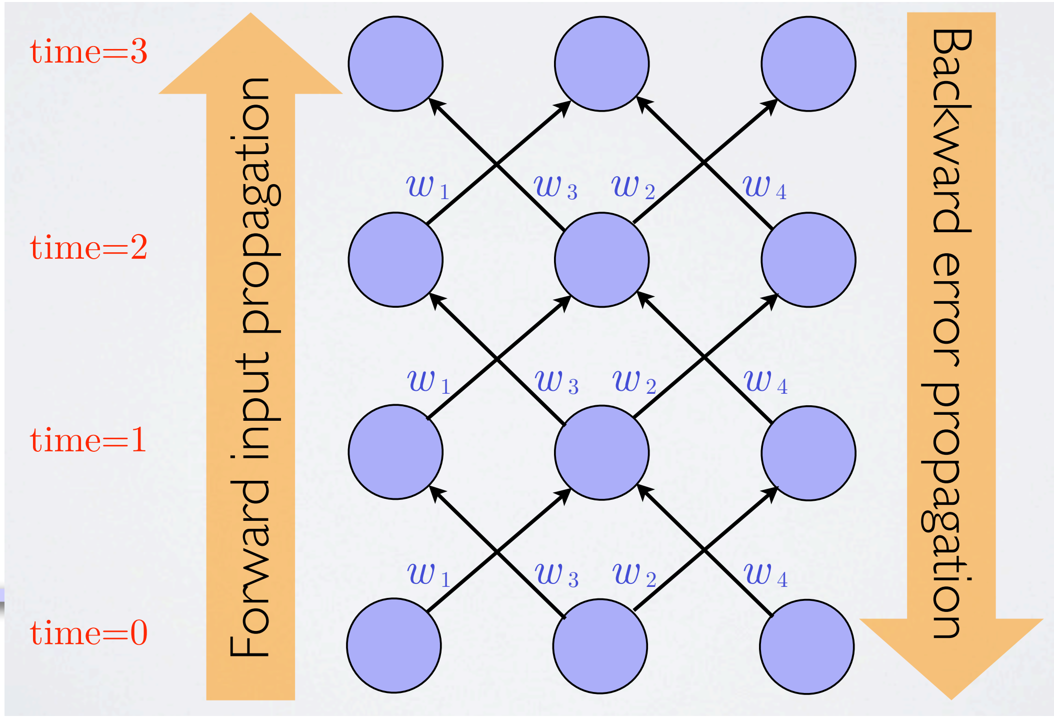
Recurrent Neural Networks (RNNs)

- Can unroll the RNN in time to get a standard feedforward NN.



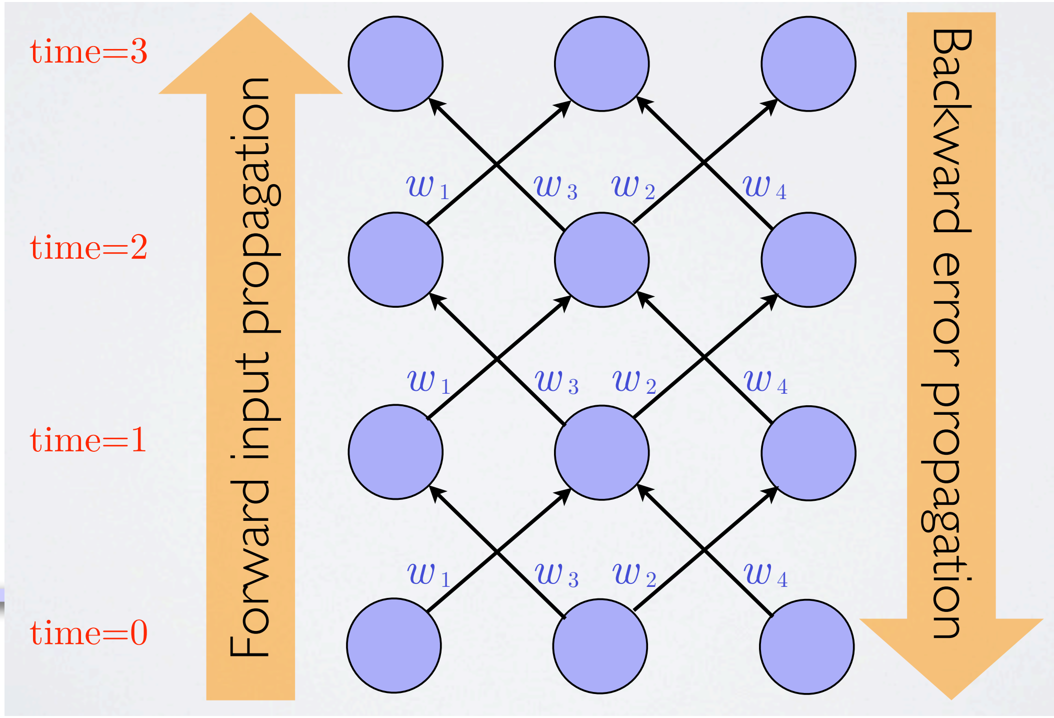
Training RNNs

- Backpropagate through time on the unrolled RNN, with constraint that corresponding weights are tied.



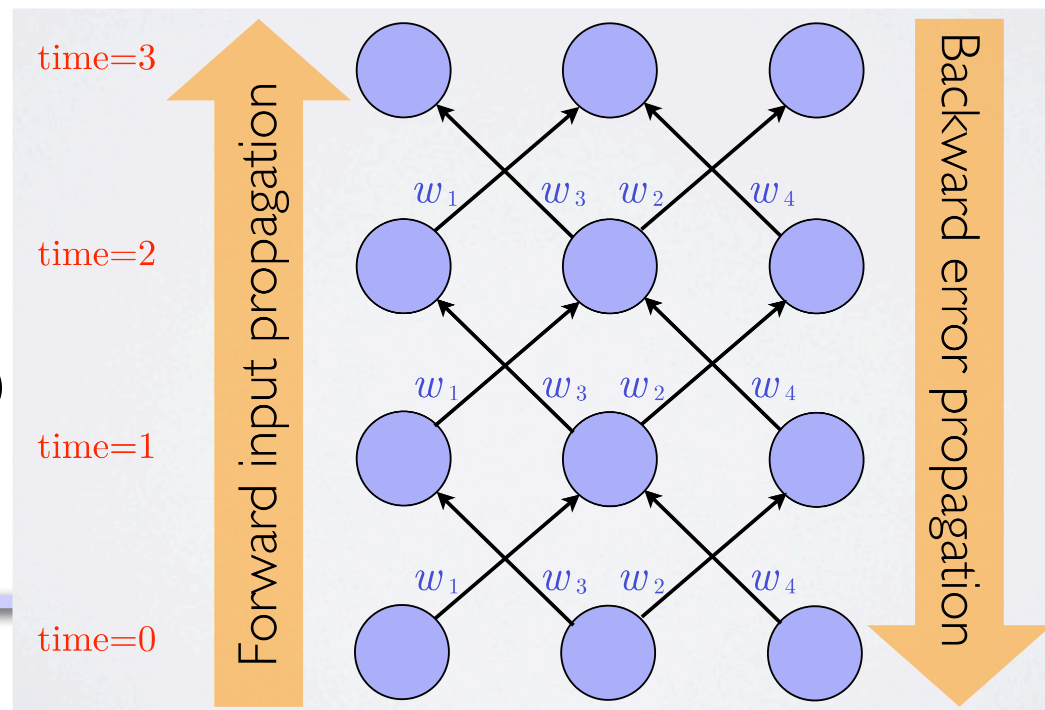
Training RNNs

- Backpropagate through time on the unrolled RNN, with constraint that corresponding weights are tied.
- Can specify the target in a few different ways:
  - Desired final activation of all units
  - Desired activations for all units for multiple time steps.
  - Desired activity of a subset of units.



Training RNNs

- Backpropagate through time on the unrolled RNN, with constraint that corresponding weights are tied.
- Can specify the target in a few different ways:
  - Desired final activation of all units
  - Desired activations for all units for multiple time steps.
  - Desired activity of a subset of units.



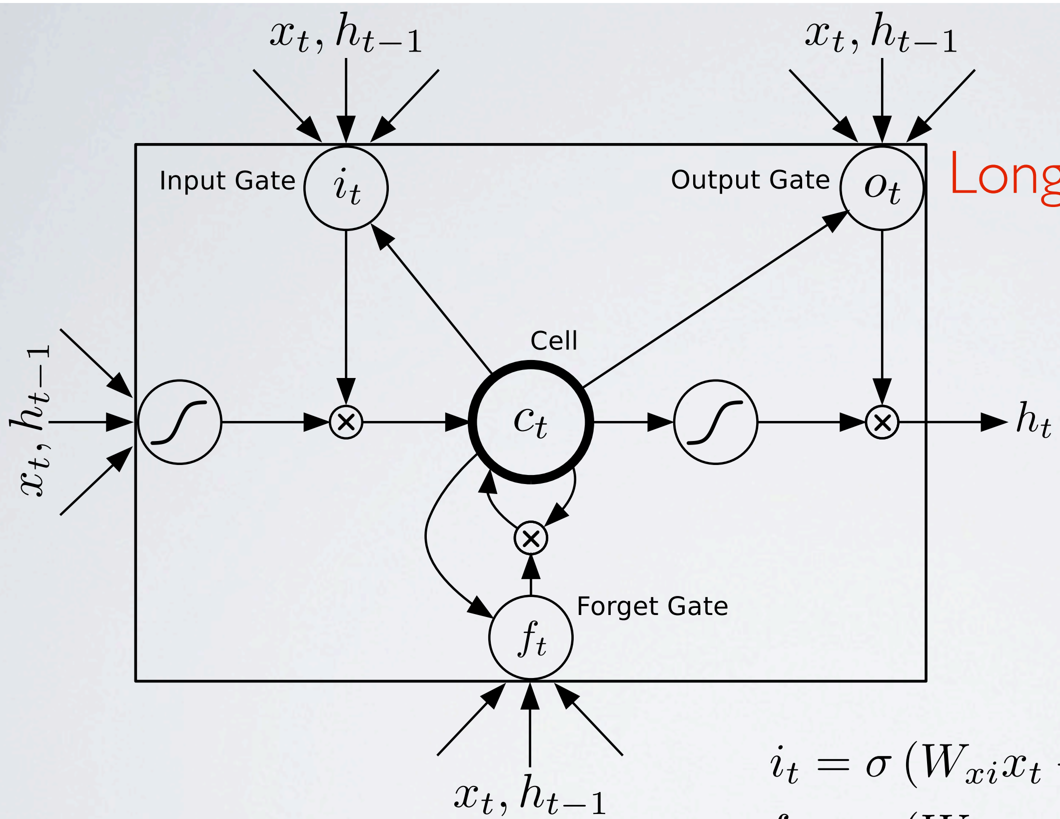
- Main challenge:

  Exploding/vanishing gradients

  (gradients shrink/grow quickly.)

  => Change the architecture.

Long short-term memory (LSTM) network



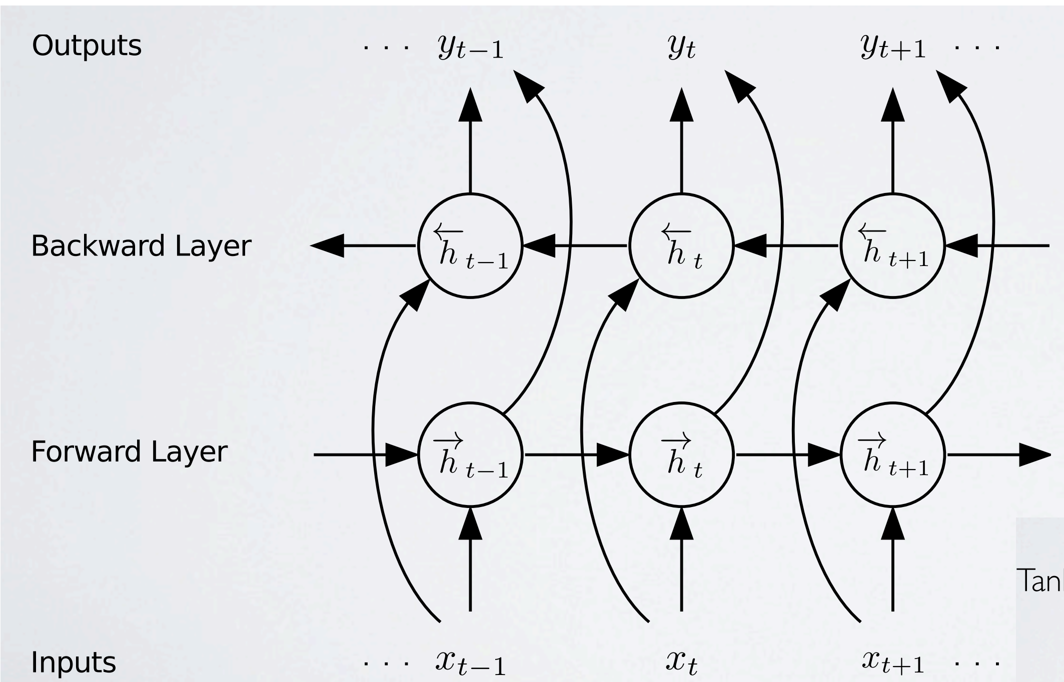
Long short-term memory cell

LSTM cell equations:

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right)$$
$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)$$
$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right)$$
$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right)$$
$$h_t = o_t \tanh(c_t)$$

LSTMs for speech recognition

Graves, Mohamed & Hinton (2013) used a bidirectional LSTM to incorporate both previous and future contextual information to predict a sequence of phonemes from the sequence of utterances.



$$\overrightarrow{h}_t = \mathcal{H}\left(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right)$$

$$\overleftarrow{h}_t = \mathcal{H}\left(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right)$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y$$

	Network	Weights	Epochs	PER
Tanh RNN ⇒	CTC-3L-500H-TANH	3.7M	107	37.6%
LSTMs	CTC-1L-250H	0.8M	82	23.9%
	CTC-1L-622H	3.8M	87	23.0%
	CTC-2L-250H	2.3M	55	21.0%
	CTC-3L-421H-UNI	3.8M	115	19.6%
	CTC-3L-250H	3.8M	124	18.6%
	CTC-5L-250H	6.8M	150	18.4%
	TRANS-3L-250H	4.3M	112	18.3%
	PRETRANS-3L-250H	**4.3M**	**144**	**17.7%**

Tasks for which LSTMs are best

- LSTM architecture has existed for many years (Hochreiter & Schmidhuber 1997).
- Several state-of-the-art results:
  - Cursive handwriting recognition (Graves & Schmidhuber, 2009)
  - Speech recognition (Graves, Mohamed & Hinton, 2013)
  - Machine translation (Sutskever, Vinyals & Le, 2014)
  - Question-answer (Weston et al., 2015)
  - Unstructured dialogue response generation (Serban et al., 2016)
- Main model for language understanding & generation tasks.

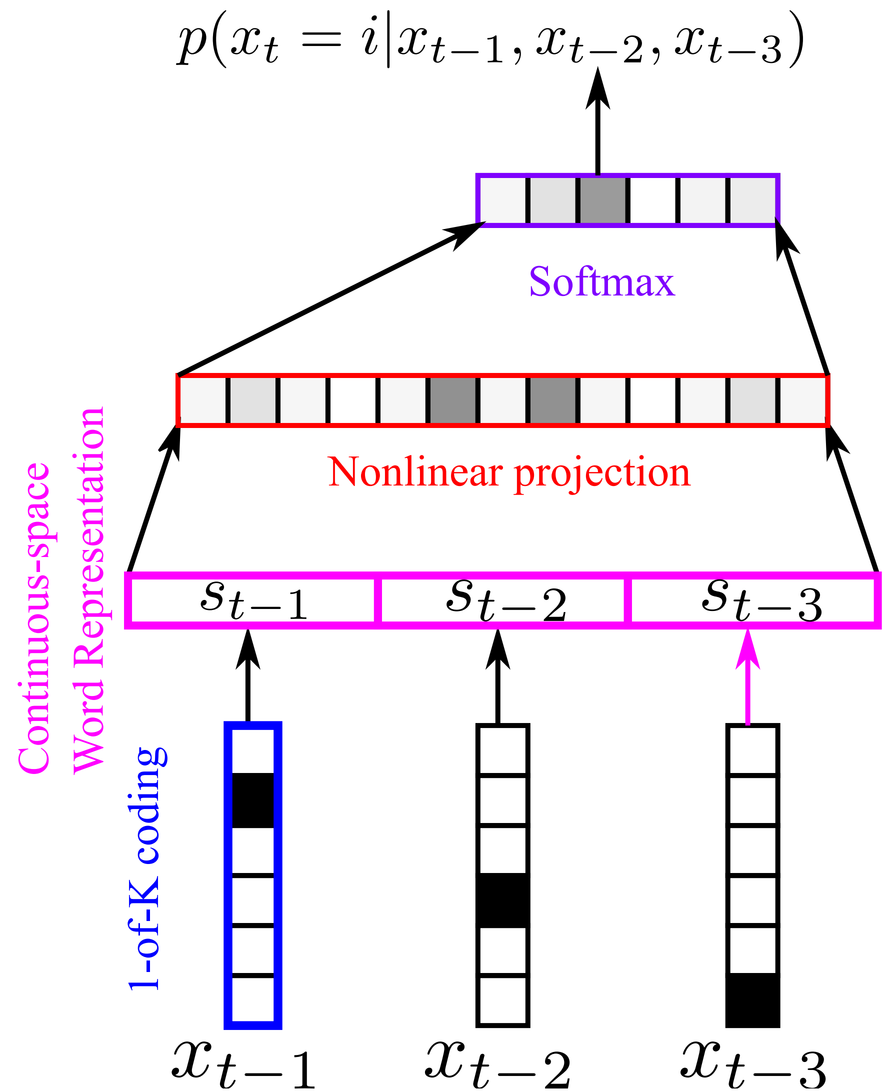
Neural Language Modelling

- Given sequence of words:
  $x_1, x_2, \ldots, x_{t-1}, x_t$

- Neural Language Modelling
  $p(x_t \mid x_{t-n},\ldots,x_{t-1}) = f_x(x_{t-n},\ldots,x_{t-1})$

Continuous space word representation

$$s_{t'} = W^\top x_{t'}, \text{ where } W \in \mathbb{R}^{|V| \times d}$$



Neural Language Modelling

- Given sequence of words:

  $x_1, x_2, \ldots, x_{t-1}, x_t$

- Neural Language Modelling

  $p(x_t \mid x_{t-n},\ldots,x_{t-1}) = f_x(x_{t-n},\ldots,x_{t-1})$

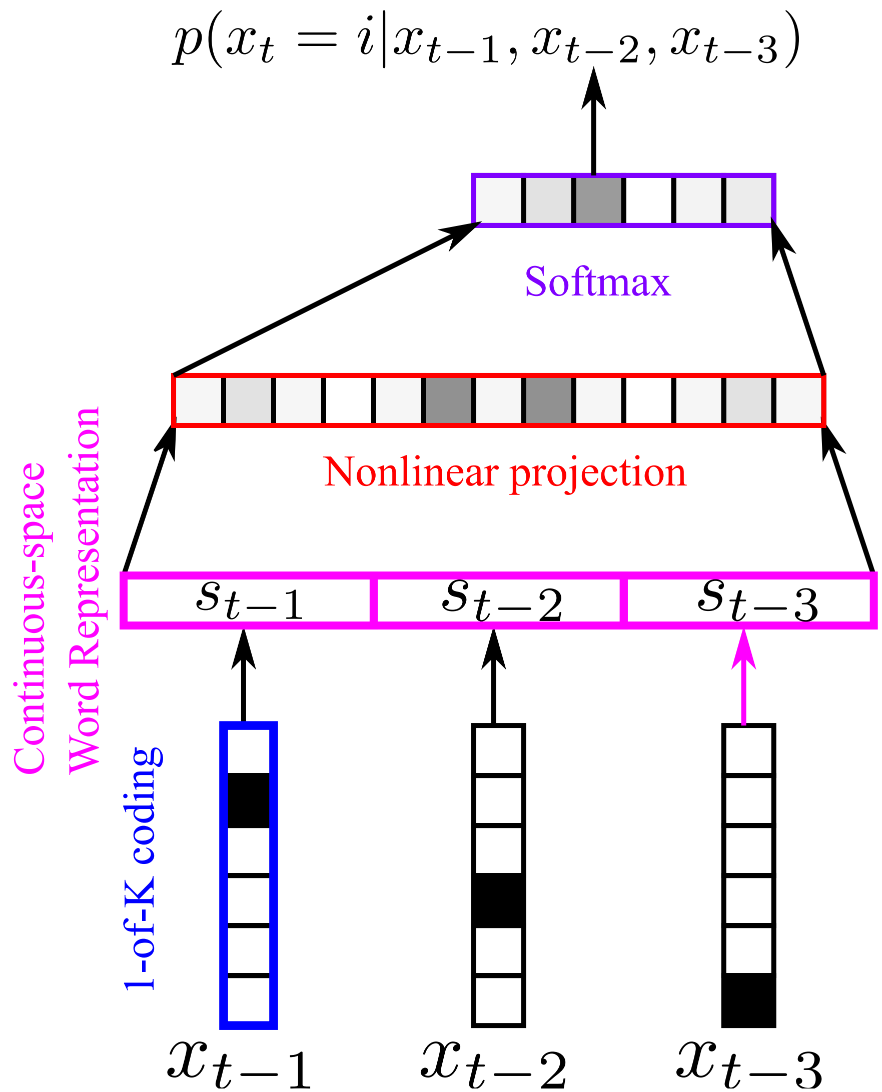
Continuous space word representation

$$s_{t'} = W^\top x_{t'}, \text{ where } W \in \mathbb{R}^{|V|\times d}$$

Nonlinear hidden layer

$$h = \tanh(U^\top [s_{t-1}; s_{t-2}; \cdots ; s_{t-n}] + b)$$

$$\text{, where } U \in \mathbb{R}^{nd\times d'} \text{ and } b \in \mathbb{R}^{d'}$$



Neural Language Modelling

- Given sequence of words:
  $x_1, x_2, \ldots, x_{t-1}, x_t$

- Neural Language Modelling
  $p(x_t \mid x_{t-n},\ldots,x_{t-1}) = f_x(x_{t-n},\ldots,x_{t-1})$

Continuous space word representation

$$s_{t'} = W^\top x_{t'}, \text{ where } W \in \mathbb{R}^{|V|\times d}$$

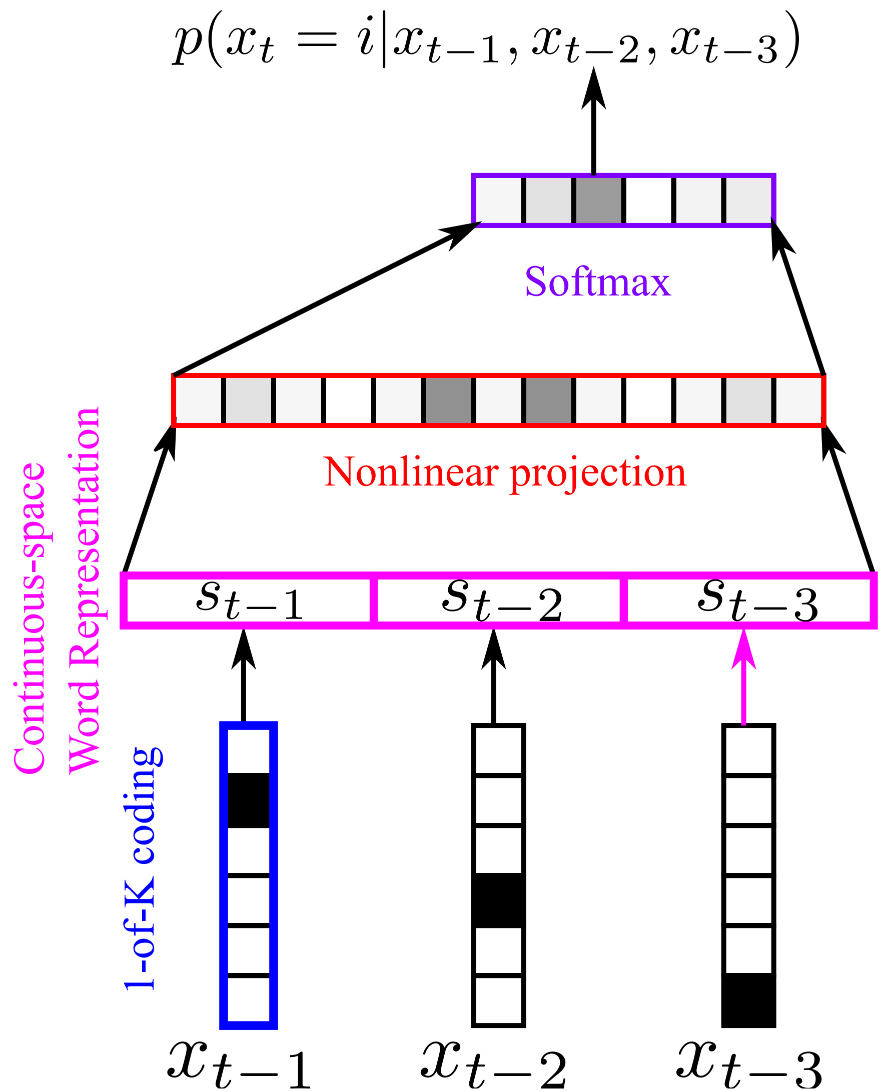
Nonlinear hidden layer

$$h = \tanh(U^\top [s_{t-1}; s_{t-2}; \cdots; s_{t-n}] + b)$$

$$\text{, where } U \in \mathbb{R}^{nd\times d'} \text{ and } b \in \mathbb{R}^{d'}$$

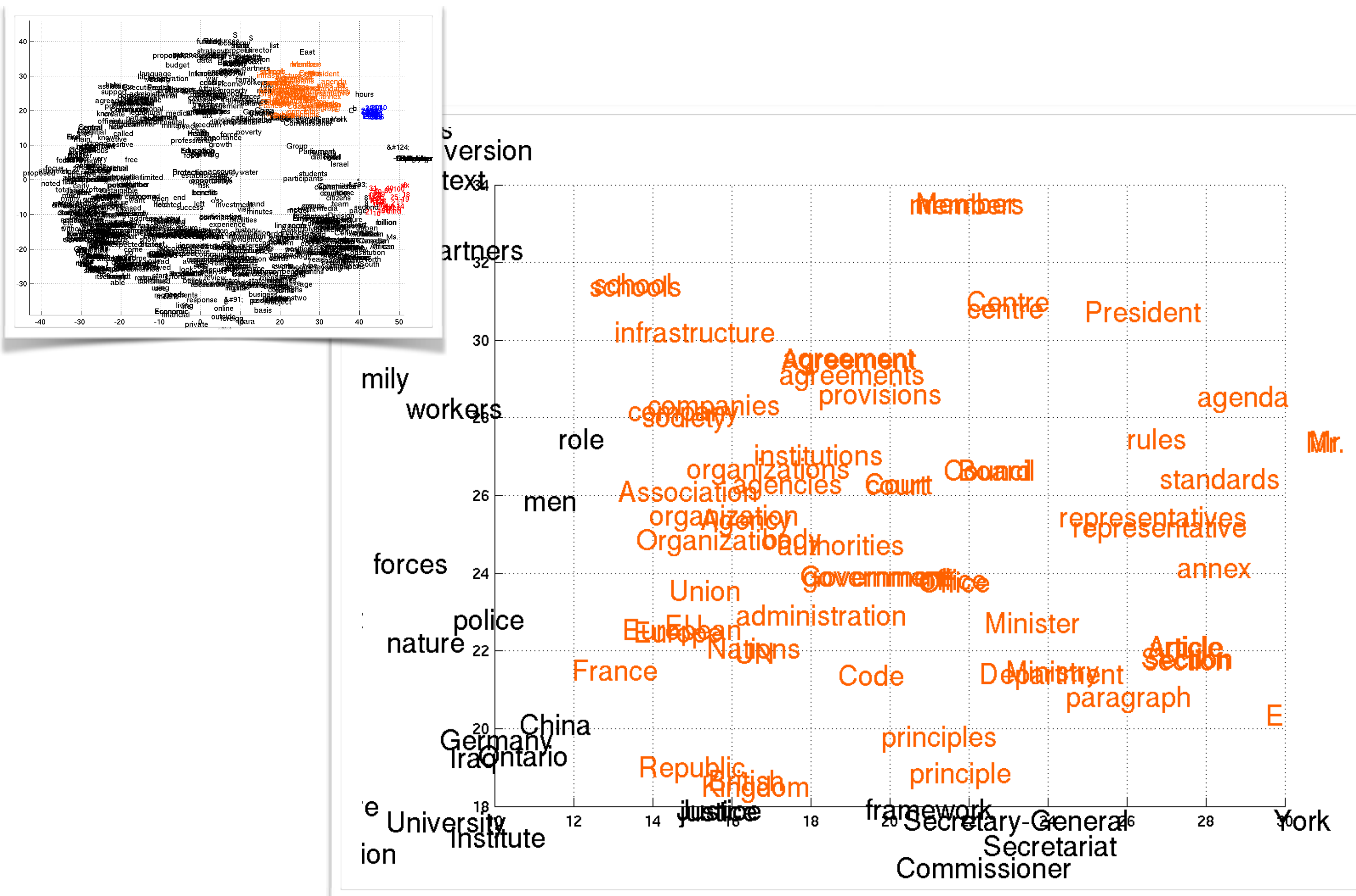
Softmax normalization

$$p(x_t = i \mid x_{t-n},\ldots,x_{t-1}) = \frac{\exp(y_i)}{\sum_{j=1}^{|V|}\exp(y_j)}$$



Neural Language Modelling

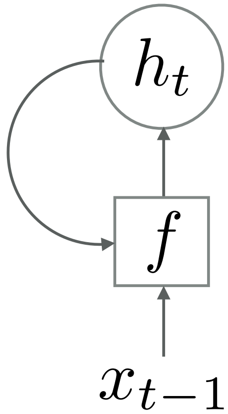
- Continuous space representation - Embeddings



Language modelling from recursion

- Directly model the conditional probabilities.

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$



- Recursive Construction:

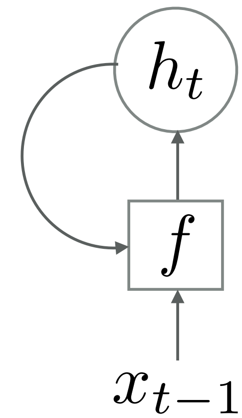
Initial Condition: $h_0 = 0$

Recursion: $h_t = f(x_{t-1}, h_{t-1})$

Language modelling from recursion

- Directly model the conditional probabilities.

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$



- Recursive Construction:

Initial Condition: $h_0 = 0$

Recursion: $h_t = f(x_{t-1}, h_{t-1})$

Example: $p(\text{eating}|\text{the, cat, is})$

(1) Initialization: $h_0 = 0$

(2) Recursion

(1) $h_1 = f(h_0, \text{the})$

(2) $h_2 = f(h_1, \text{cat})$

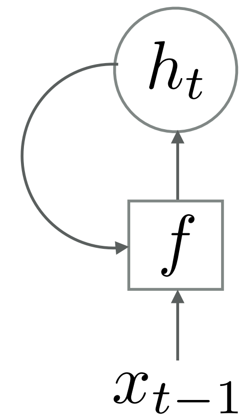
(3) $h_3 = f(h_2, \text{is})$

(3) Readout: $p(\text{eating}|\text{the, cat, is}) = g(h_3)$

Language modelling from recursion

- Directly model the conditional probabilities.

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$



- Recursive Construction:

Initial Condition: $h_0 = 0$

Recursion: $h_t = f(x_{t-1}, h_{t-1})$

Example: $p(\text{eating}|\text{the, cat, is})$

(1) Initialization: $h_0 = 0$

(2) Recursion

(1) $h_1 = f(h_0, \text{the})$

(2) $h_2 = f(h_1, \text{cat})$

(3) $h_3 = f(h_2, \text{is})$

(3) Readout: $p(\text{eating}|\text{the, cat, is}) = g(h_3)$

We call h_t an internal hidden state,

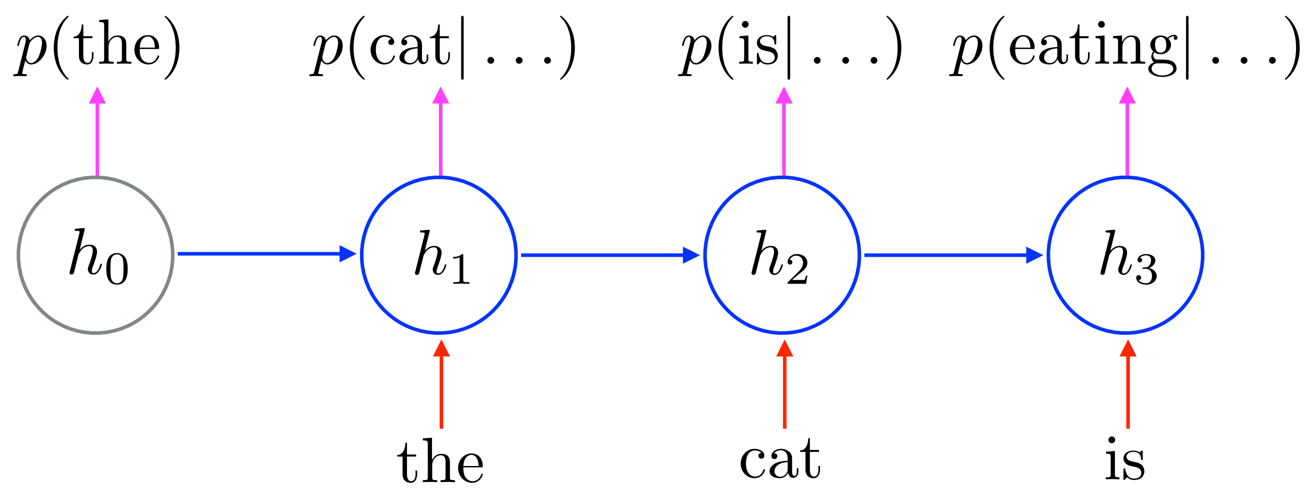
or **memory**, which summarizes history from x_1 up to x_{t-1}.

Recurrent neural language model

Transition Function $h_t = f(h_{t-1}, x_{t-1})$

Output/Readout Function $p(x_t = w | x_1, \ldots, x_{t-1}) = g_w(h_t)$

Example: $p(\text{the}, \text{cat}, \text{is}, \text{eating})$



Training an RNN language model

- Loss function:

Log-Probability of a sentence $(x_1, x_2, \ldots, x_T)$

$$\log p(x_1, x_2, \ldots, x_T) = \sum_{t=1}^{T} \log p(x_t \mid x_1, \ldots, x_{t-1})$$

Training an RNN language model

- Loss function:

Log-Probability of a sentence $(x_1, x_2, \ldots, x_T)$

$$\log p(x_1, x_2, \ldots, x_T) = \sum_{t=1}^{T} \log p(x_t \mid x_1, \ldots, x_{t-1})$$

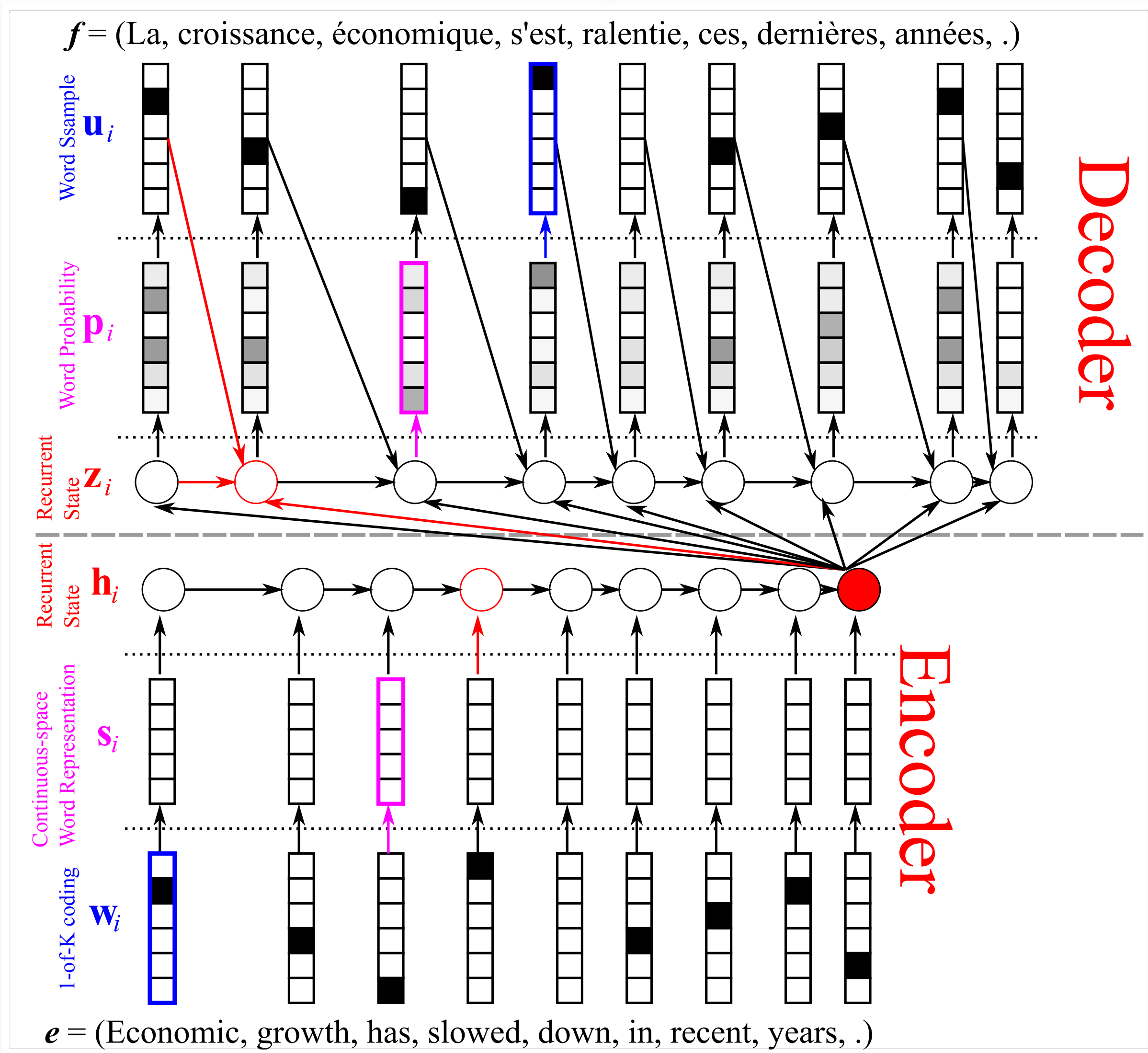
- Train an RNN LM to maximize the log-prob's of training sentences.

Given a training set of N sentences: $\{(x_1^1, \ldots, x_{T_1}^1), \ldots, (x_1^N, \ldots, x_{T_N}^N)\}$

$$\text{maximize}_\Theta \frac{1}{N} \sum_{n=1}^{N} \log p(x_1^n, \ldots, x_{T_n}^n)$$

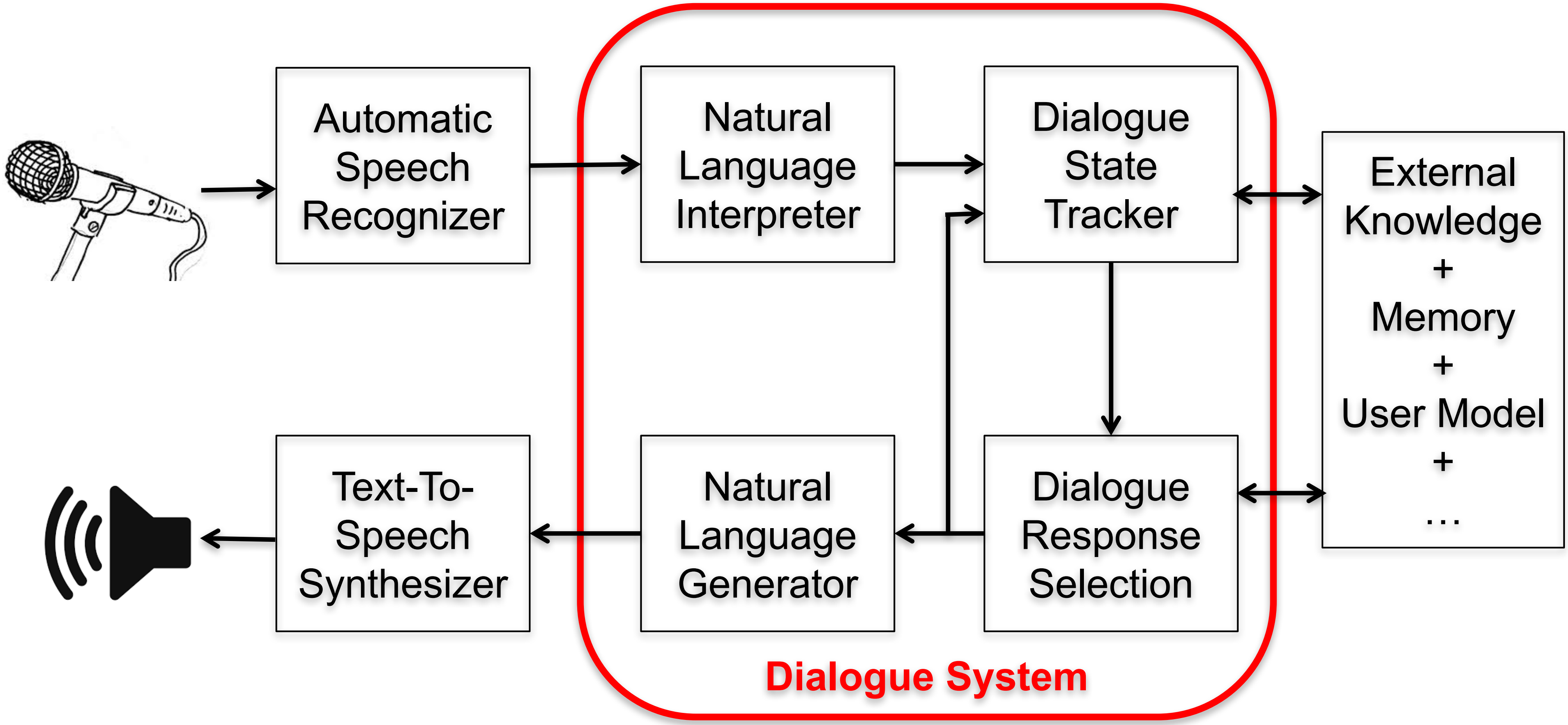
$$\Longleftrightarrow \ \text{minimize}_\Theta J(\Theta) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p(x_t^n | x_1^n \ldots, x_{t-1}^n)$$

Neural Machine Translation



(Chrisman, 1991;
Forcada&Ñeco, 1997;
Castaño&Casacuberta, 1997;
Kalchbrenner&Blunsom, 2013;
Sutskever et al., 2014;
Cho et al., 2014)

Dialogue management



Dialogue datasets

Dataset	Type	Task	# Dialogues	# Utterances	Description
Switchboard [2]	Human-human spoken	Various	2,400	—	Telephone conversations on pre-specified topics
DSTC1 [9]	Human-computer spoken	State tracking	15,000	210,000	Bus ride information system
DSTC2 [4]	Human-computer spoken	State tracking	3,000	24,000	Restaurant booking system
DSTC3 [3]	Human-computer spoken	State tracking	2,265	15,000	Tourist information system
DSTC4 [5]	Human-human spoken	State tracking	35	—	21 hours of tourist info exchange over Skype
Twitter Corpus [6]	Human-human micro-blog	Next utterance generation	1,300,000	3,000,000	Post/ replies extracted from Twitter
Twitter Triple Corpus [8]	Human-human micro-blog	Next utterance generation	29,000,000	87,000,000	A-B-A triples from Twitter replies
Sina Weibo [7]	Human-human micro-blog	Next utterance generation	4,435,959	8,871,918	Post/ reply pairs extracted from Weibo

Ubuntu chat corpus

Initial chat room log:

Time	User	Utterance
03:44	Old	I dont run graphical ubuntu, I run ubuntu server.
03:45	kuja	Taru: Haha sucker.
03:45	Taru	Kuja: ?
03:45	bur[n]er	Old: you can use "ps ax" and "kill (PID#)"
03:45	kuja	Taru: Anyways, you made the changes right?
03:45	Taru	Kuja: Yes.
03:45	LiveCD	or killall speedlink
03:45	kuja	Taru: Then from the terminal type: sudo apt-get update
03:46	_pm	if i install the beta version, how can i update it when the final version comes out?
03:46	Taru	Kuja: I did.

Ubuntu chat corpus

Initial chat room log:

Time	User	Utterance
03:44	Old	I dont run graphical ubuntu, I run ubuntu server.
03:45	kuja	Taru: Haha sucker.
03:45	Taru	Kuja: ?
03:45	bur[n]er	Old: you can use "ps ax" and "kill (PID#)"
03:45	kuja	Taru: Anyways, you made the changes right?
03:45	Taru	Kuja: Yes.
03:45	LiveCD	or killall speedlink
03:45	kuja	Taru: Then from the terminal type: sudo apt-get update
03:46	_pm	if i install the beta version, how can i update it when the final version comes out?
03:46	Taru	Kuja: I did.

Disentangled into 2-way conversation:

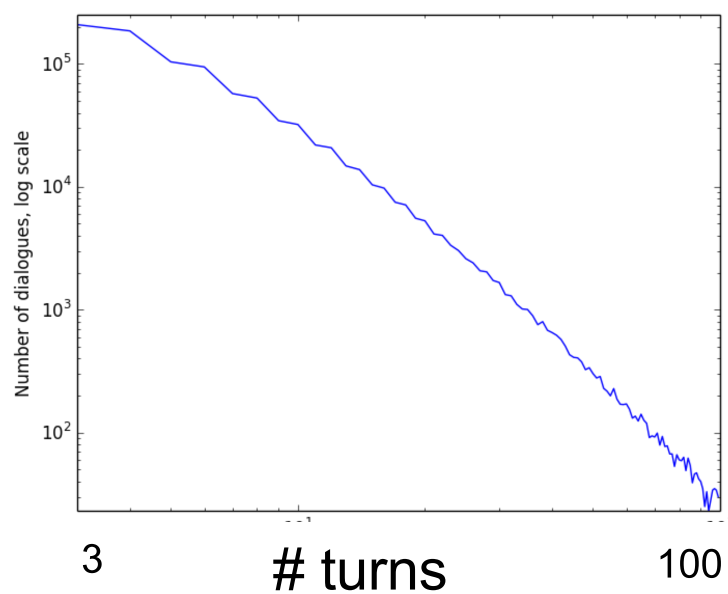
Sender	Recipient	Utterance
Old		I dont run graphical ubuntu, I run ubuntu server.
bur[n]er	Old	you can use "ps ax" and "kill (PID#)"
kuja	Taru	Haha sucker.
Taru	Kuja	?
kuja	Taru	Anyways, you made the changes right?
Taru	Kuja	Yes.
kuja	Taru	Then from the terminal type: sudo apt-get update
Taru	Kuja	I did.

Ubuntu dialogue corpus

Key properties:

# dialogues (human-human)	930,000
# utterances (in total)	7,100,000
# words (in total)	100,000,000
Min. # turns per dialogue	3
Avg. # turns per dialogue	7.71
Avg. # words per utterance	10.34
Median conversation length (min)	6

Histogram of number of turns per dialogue:



Task 1: Next utterance classification

Context:

….

“any apache hax around ? I just deleted all of _path_ - which package provides it?”

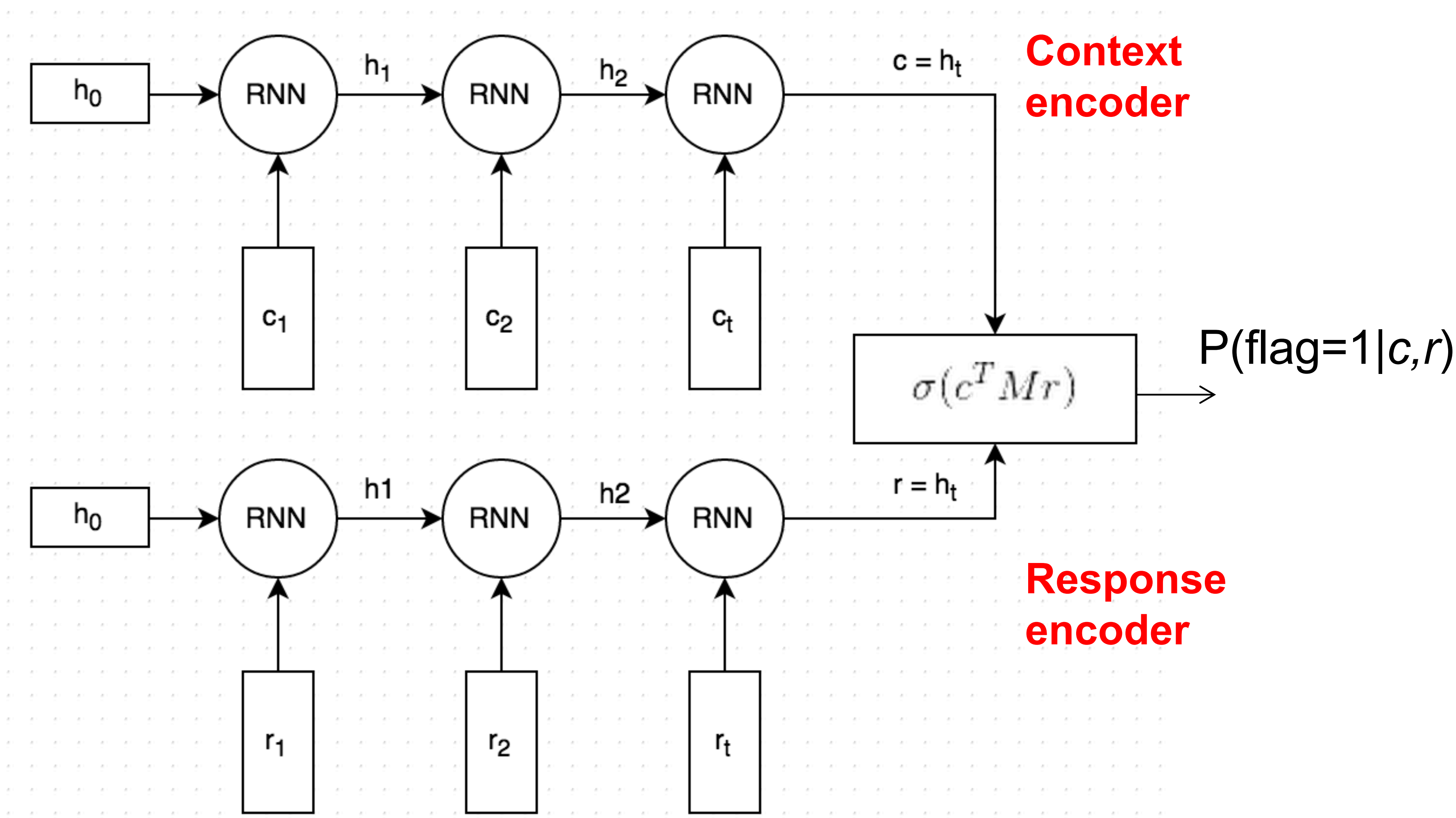
“reconfiguring apache do n’t solve it?”

Response 1: “does n’t seem to, no”

Response 2: “you can log in but not transfer files?”

The Dual Encoder model

[*Lowe, Pow, Serban, Pineau, SIGdial 2015*]



Results: Dual Encoder model on Ubuntu dataset

Method	TF-IDF	RNN
1 in 2 R@1	65.9%	87.8%
1 in 10 R@1	41.0%	60.4%
1 in 10 R@2	54.5%	74.5%
1 in 10 R@5	70.8%	92.6%

TF-IDF : Term frequency – inverse document frequency

TF*(t,d)* = frequency of a word *t* in a document *d*

IDF(t,D) = measure of how much information the word t provides across corpus of documents *D*

TF-IDF(*t,d,D*) = TF(*t,d*) x IDF(*t,D*)

User study

Context:
“Hello. anybody could help? __EOS__”
“You need to say what your problem is, first.”

Response: “the text of some of my applications' menu are not well displayed”
Response: “do you know if cs:s runs good on it?”
Response: “he wants emerald theme...”
Response: “i dont have a cd-rom drive.”
Response: “But wont the number be part? eg., sda4 is always '4'?”

User study

Context:
"Hello. anybody could help? __EOS__"
"You need to say what your problem is, first."

Response: "the text of some of my applications' menu are not well displayed"
Response: "do you know if cs:s runs good on it?"
Response: "he wants emerald theme..."
Response: "i dont have a cd-rom drive."
Response: "But wont the number be part? eg., sda4 is always '4'?"

	Number of Users	Ubuntu Corpus R@1	Ubuntu Corpus R@2
AMT non-experts	135	$52.9 \pm 2.7\%$	$69.4 \pm 2.5\%$
AMT experts	10	$52.0 \pm 9.8\%$	$63.0 \pm 9.5\%$
Lab experts	8	$83.8 \pm 8.1\%$	$87.8 \pm 7.2\%$
ANN model	machine	66.2%	83.7%

Task 2: Large corpus next-utterance retrieval

- **Search full dataset** for a good response: 1 in 10^6 R@10
  - Pre-compute the response encoding for all candidate utterances.
- Output ranked list of responses based on P(flag=1|*c*,*r*) = $\sigma(c^T Mr)$.

Query("why is my laptop losing battery so fast") **Dual Encoder model**

Top 10 likely responses in order

[[0.99915196]] i wonder if it ' s a heat issue. or it ' s draining the battery so fast that your laptop will shutdown

[[0.99909478]] didnt know that there is a page for apm , thanks :d. well , apm is not quite what i needed . my battery is going low too fast - although it should work at least __number__ hours (up to __number__) , it is **unknown** empty at ~ 1:40 . it is a toshiba m50 satellite and i think that i have to **unknown** something to spare some energy . the notebook an the accu are __number__ hours old ...

[[0.9989985]] sorry rodd !. how long does it stay on without being plugged in ?. and how old is battery roughly ?

[[0.99867463]] any ideas as to why nothing changes ?. yes to all ?. ok , here ' s what i ' ve got __url__. i followed this guide : __url__ to install the **unknown** i do n't mind restarting , i can check the bios and see what the temp is according to it. brb. nothing changed , cpu temp according to bios is the __organization__ temp in sensors and __organization__ temp is the __organization__ cpu temp. nothing changed , cpu temp according to bios is the __organization__ temp in sensors and __organization__ temp is the __organization__ cpu tem

[[0.99856425]] i will seriously give you , free of charge , a __number__ ghz athlonxp on an a7v8x with roughly __number__ gb ram. why do you people have such horrid hardware ?

[[0.99848473]] i have this other computer , mobo is a asus **unknown** and no network card ive tried in it will work , i have a cheap network hub that is ok , this comp is in it , i got another old one going on it , but it refuses to use it. ive tried about 10-12 different network adaptors and short of trying to put in a **unknown** system for it im out of ideas. so far infact , i only have a intel adaptor on a older asus based comp and the __number__ 3com card in this computer going , most of the other ones i tried were infact , identical models to the 3com in this computer , and i tested them to work fine at school ...

[[0.99823273]] blast ... forgot about the __organization__ settings , have n't checked them ... will reboot & have a look @ bios . thanks !. homebuilt - __person__ a7n8x-e mobo , 1gb ddr , __number__ ghz amd xp-m cpu

Query("why is my laptop losing battery so fast", "tfidf") Tf-idf match on query

[1] come again ?. you might want to check __url__

[2] ibm thinkpad t22 ?

[3] __gpe__ to know :)

[4] i tried there but there isnt my problem

[5] i guess is another problem .

[6] __gpe__ , np . thanks for your time :)

[7] try livecd , most likely it is hardware issue

[8] this shows my how much time is left . but i would like to see the actual discharge rate

[9] __gpe__ prob not. your __organization__ probably limits charging above a certain % too (why it says __number__ minutes vs say __number__)

[10] that is correct. fast user switching seems to work better for me (it uses the __organization__ package for doing it . it is probably a newer version in __gpe__)

Measuring response retrieval quality

- **BLEU score** from Machine Translation analyzes co-occurrence of n-grams in 2 sentences.

Score computed between true response and generated response.

Dual Encoder model	17.08 *(high variance)*
Tf-idf	5.81
Random response	0.20

Generative modeling of responses

<speaker A> *How are you, Tom?* </s>

<speaker B> *I'm good, thanks* <pause> *did you get my message yesterday?* </s>

<speaker B nods>

<speaker B> *Yes, it was interesting.* </s>

<speaker C turns head around>

<speaker C> *what message?* </s>

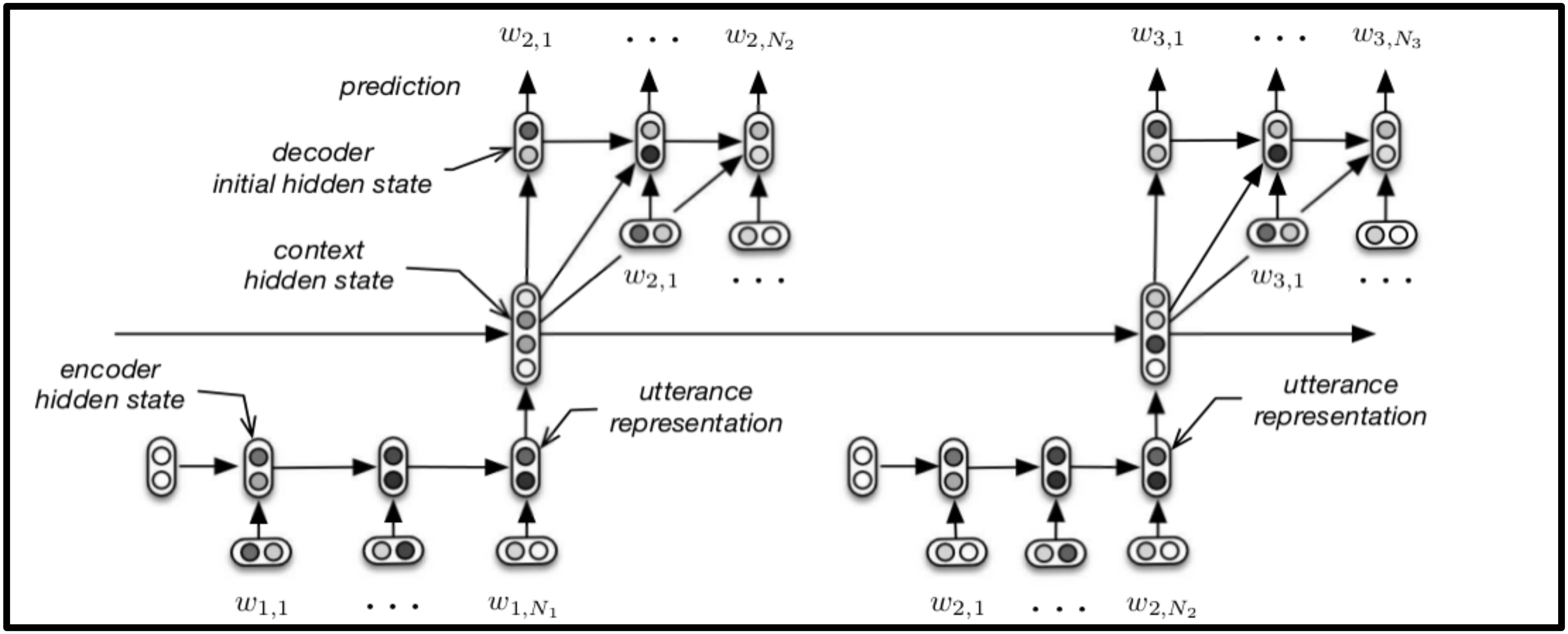
...

Task 3: Natural language response generation

[*Serban, A. Sordoni, Y. Bengio, A. Courville, J. Pineau, AAAI 2015*]

Hierarchical Encoder-Decoder

– Encode each utterance + Encode the conversation
– Decode response into natural language



Results

Model	Perplexity	Perplexity@U_3	Error-Rate	Error-Rate@U_3
Backoff N-Gram	64.89	65.05	-	-
Modified Kneser-Ney	60.11	54.75	-	-
Absolute Discounting N-Gram	56.98	57.06	-	-
Witten-Bell Discounting N-Gram	53.30	53.34	-	-
RNN	35.63 ± 0.16	35.30 ± 0.22	$66.34\% \pm 0.06$	$66.32\% \pm 0.08$
DCGM-I	36.10 ± 0.17	36.14 ± 0.26	$66.44\% \pm 0.06$	$66.57\% \pm 0.10$
HRED	36.59 ± 0.19	36.26 ± 0.29	$66.32\% \pm 0.06$	$66.32\% \pm 0.11$
HRED + Word2Vec	33.95 ± 0.16	33.62 ± 0.25	$66.06\% \pm 0.06$	$66.05\% \pm 0.09$
RNN + SubTle	27.09 ± 0.13	26.67 ± 0.19	$64.10\% \pm 0.06$	$64.07\% \pm 0.10$
HRED + SubTle	27.14 ± 0.12	26.60 ± 0.19	$64.10\% \pm 0.06$	$64.03\% \pm 0.10$
HRED-Bi. + SubTle	$\mathbf{26.81 \pm 0.11}$	$\mathbf{26.31 \pm 0.19}$	$\mathbf{63.93\% \pm 0.06}$	$\mathbf{63.91\% \pm 0.09}$

Results

Model	Perplexity	Perplexity@U_3	Error-Rate	Error-Rate@U_3
Backoff N-Gram	64.89	65.05	-	-
Modified Kneser-Ney	60.11	54.75	-	-
Absolute Discounting N-Gram	56.98	57.06	-	-
Witten-Bell Discounting N-Gram	53.30	53.34	-	-
RNN	35.63 ± 0.16	35.30 ± 0.22	$66.34\% \pm 0.06$	$66.32\% \pm 0.08$
DCGM-I	36.10 ± 0.17	36.14 ± 0.26	$66.44\% \pm 0.06$	$66.57\% \pm 0.10$
HRED	36.59 ± 0.19	36.26 ± 0.29	$66.32\% \pm 0.06$	$66.32\% \pm 0.11$
HRED + Word2Vec	33.95 ± 0.16	33.62 ± 0.25	$66.06\% \pm 0.06$	$66.05\% \pm 0.09$
RNN + SubTle	27.09 ± 0.13	26.67 ± 0.19	$64.10\% \pm 0.06$	$64.07\% \pm 0.10$
HRED + SubTle	27.14 ± 0.12	26.60 ± 0.19	$64.10\% \pm 0.06$	$64.03\% \pm 0.10$
HRED-Bi. + SubTle	$\mathbf{26.81 \pm 0.11}$	$\mathbf{26.31 \pm 0.19}$	$\mathbf{63.93\% \pm 0.06}$	$\mathbf{63.91\% \pm 0.09}$

Conclusion?

- Neural models are better than n-gram models.
- HRED is better than RNNs (handles longer dialogues)
- Incorporating Word2Vec and SubTle improves performance.

Evaluation metrics

✅ **Perplexity, word error rate**

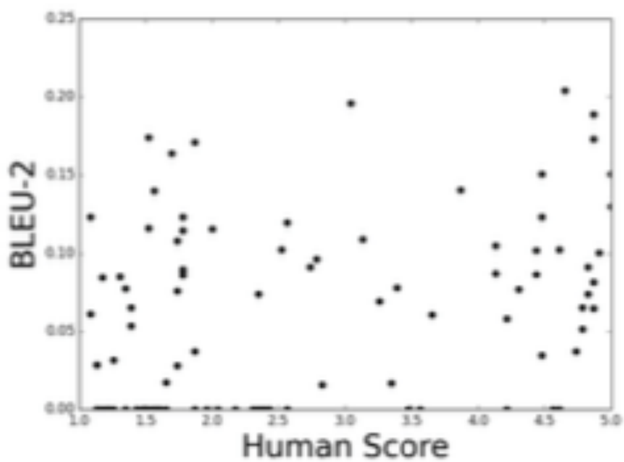
Word overlap metrics: Count number of overlapping word subsets between generated and reference response.

- From machine translation: **BLEU**, METEOR
- From text summarization: ROUGE

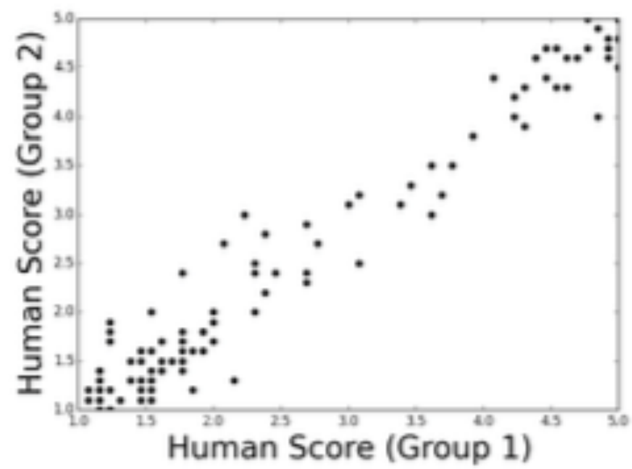
Correlation with human judgment

[*Liu, Lowe, Serban, Noseworthy, Charlin, Pineau, EMNLP 2016*]

BLEU-2



Between humans



Task design

Context:
Hello. anybody could help? __EOS__
You need to say what your problem is, first.

Response 1: the text of some of my applications' menu are not well displayed (ubuntu 8.10) .
Response 2: do you know if cs:s runs good on it?
Response 3: he wants emerald theme...
Response 4: i dont have a cd-rom drive.
Response 5: But wont the number be part? eg., sda4 is always '4'?

Space of acceptable next utterances is large!

It's hard to pick a good loss function!

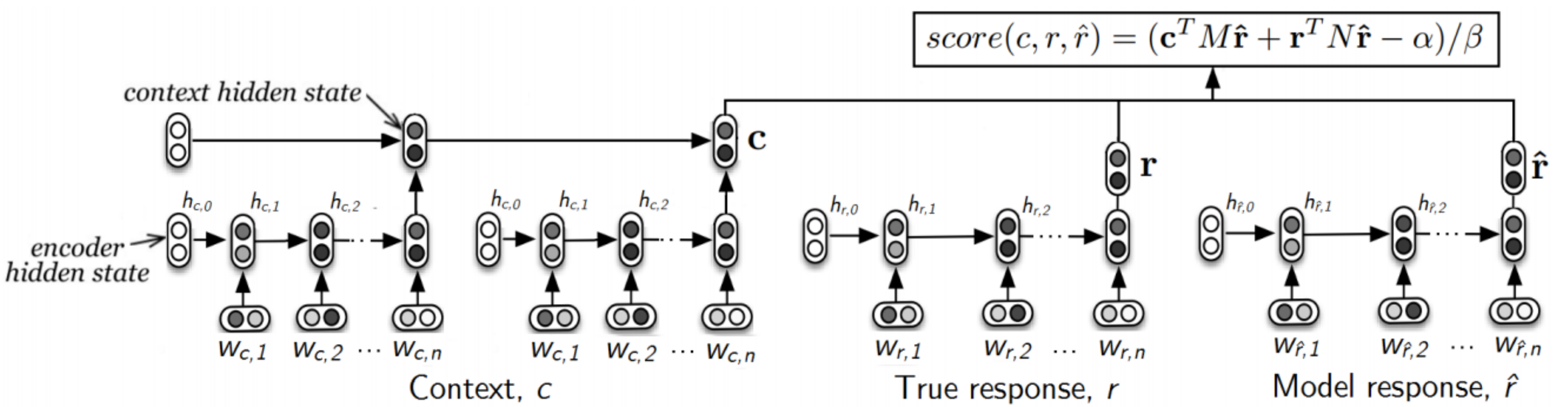
Automatic Dialogue Evaluation Model (ADEM)

- Given context, model response, and reference response,

  ADEM tries to predict the human score for that response.

$$\mathcal{L} = \sum_{i=1:K} [score(c_i, r_i, \hat{r}_i) - human\_score_i]^2 + \gamma||\theta||_1$$

- Minimize:



What you should know

- Types of deep learning architectures:
  - Stacked autoencoders
  - Convolutional neural networks
  - **Recurrent neural networks**

- Examples of successful applications.

- From more on Deep Learning, see invited talks at DLSS'16:

  https://sites.google.com/site/deeplearningsummerschool2016/speakers

  (Some material from today's lecture taken from Kyunghyun Cho's talk.)