Neural Machine Translation

Graham Neubig
2016-12-6

watashi wa CMU de kouen wo shiteimasu

I am giving a talk at CMU (end)

Estimate the Probability of Next Word

F = "watashi wa kouen wo shiteimasu"

$P(e_1\text{=I}\mid F) = 0.96$	$P(e_1\text{=talk}\mid F) = 0.03$ $P(e_1\text{=it}\mid F) = 0.01$	...	e_1= I
$P(e_2\text{=am}\mid F, e_1) = 0.9$	$P(e_2\text{=was}\mid F, e_1) = 0.09$	...	e_2= am
$P(e_3\text{=giving}\mid F, e_{1,2}) = 0.4$	$P(e_3\text{= talking}\mid F, e_{1,2}) = 0.3$ $P(e_3\text{=presenting}\mid F, e_{1,2}) = 0.03$	...	e_3= giving
$P(e_4\text{=a}\mid F, e_{1,3}) = 0.8$	$P(e_4\text{=my}\mid F, e_{1,3}) = 0.15$	...	e_4= a
$P(e_5\text{=talk}\mid F, e_{1,4}) = 0.4$ $P(e_5\text{=presentation}\mid F, e_{1,4}) = 0.3$	$P(e_5\text{=lecture}\mid F, e_{1,4}) = 0.15$ $P(e_5\text{=discourse}\mid F, e_{1,4}) = 0.1$	...	e_5= talk
$P(e_6\text{=(end)}\mid F, e_{1,5}) = 0.8$	$P(e_6\text{=now}\mid F, e_{1,5}) = 0.1$	...	e_6= (end)

In Other Words, Translation Can be Formulated As:

A Probability Model

$$P(E|F) = \prod_{i=1}^{I+1} P(e_i | F, e_1^{i-1})$$

A Translation Algorithm

$i = 0$
while e_i is not equal to "(end)":
  $i \leftarrow i+1$
  $e_i \leftarrow \text{argmax}_e\, P(e_i | F, e_{1,i-1})$

We learn the probabilities with neural networks!

Why is This Exciting?

- Amazing results:
  Within three years of invention, outperforming models developed over the past 15 years, and deployed in commercial systems
- Incredibly simple implementation:
  Traditional machine translation (e.g. 6k lines of Python)
  Neural machine translation (e.g. 280 lines of Python)
- Machine translation as machine learning:
  Easy to apply new machine techniques directly

Predicting Probabilities

Translation Model → Language Model

Translation Model Probability

$$P(E|F)=\prod_{i=1}^{I+1} P(e_i|F, e_1^{i-1})$$

↓ Forget the input *F*

Language Model Probability

$$P(E)=\prod_{i=1}^{I+1} P(e_i|e_1^{i-1})$$

Problem: How to predict next word $P(e_i|e_1^{i-1})$

Predicting by Counting

- Calculate word strings in corpus, take fraction

$$P(w_i | w_1 \ldots w_{i-1}) = \frac{c(w_1 \ldots w_i)}{c(w_1 \ldots w_{i-1})}$$

i live in pittsburgh . </s>
i am a graduate student . </s>
my home is in michigan . </s>

P(live | <s> i) = c(<s> i live)/c(<s> i) = 1 / 2 = 0.5

P(am | <s> i) = c(<s> i am)/c(<s> i) = 1 / 2 = 0.5

Problems With Counting

- Weak when counts are low:

Training:

i live in pittsburgh . </s>
i am a graduate student . </s>
my home is in michigan . </s>

Test:

<s> i live in michigan . </s>

⇩

P(michigan|<s> i live in) = 0/1 = 0

⇩

P(W=<s> i live in michigan . </s>) = 0

- **Solutions:** Restricting length, smoothing

Log-linear Language Model [Chen+ 00]

- Based on the previous words, give all words a score s

Previous words: “giving a”

$$
\begin{array}{l} \text{a} \\ \text{the} \\ \text{talk} \\ \text{gift} \\ \text{hat} \\ \ldots \end{array}
\quad
\boldsymbol{b} = \begin{pmatrix} 3.0 \\ 2.5 \\ -0.2 \\ 0.1 \\ 1.2 \\ \ldots \end{pmatrix}
\quad
\boldsymbol{w}_{1,a} = \begin{pmatrix} -6.0 \\ -5.1 \\ 0.2 \\ 0.1 \\ 0.6 \\ \ldots \end{pmatrix}
\quad
\boldsymbol{w}_{2,giving} = \begin{pmatrix} -0.2 \\ -0.3 \\ 1.0 \\ 2.0 \\ -1.2 \\ \ldots \end{pmatrix}
\quad
\boldsymbol{s} = \begin{pmatrix} -3.2 \\ -2.9 \\ 1.0 \\ 2.2 \\ 0.6 \\ \ldots \end{pmatrix}
$$

Words we're predicting

How likely are they?

How likely given the previous word is “a”?

How likely given two words before is “giving”?

Total score

Log-linear Language Model [Chen+ 00]

- Convert scores into probabilities by taking exponent and normalizing (called the softmax function)

$$P\left(e_i = x \mid e_{i-n+1}^{i-1}\right) = \frac{e^{s\left(e_i = x \mid e_{i-n+1}^{i-1}\right)}}{\sum_{\tilde{x}} e^{s\left(e_i = \tilde{x} \mid e_{i-n+1}^{i-1}\right)}}$$

$$\boldsymbol{p}\left(e_i \mid e_{i-n+1}^{i-1}\right) = \mathrm{softmax}\left(\boldsymbol{s}\left(e_i \mid e_{i-n+1}^{i-1}\right)\right)$$

$$\begin{array}{l} \text{a} \\ \text{the} \\ \text{talk} \\ \text{gift} \\ \text{hat} \\ \ldots \end{array} \quad \boldsymbol{s} = \begin{pmatrix} -3.2 \\ -2.9 \\ 1.0 \\ 2.2 \\ 0.6 \\ \ldots \end{pmatrix} \xrightarrow{\text{softmax}} \boldsymbol{p} = \begin{pmatrix} 0.002 \\ 0.003 \\ 0.329 \\ 0.444 \\ 0.090 \\ \ldots \end{pmatrix}$$

Learning Log Linear Models

- Often learn using Stochastic Gradient Descent (SGD)
- **Basic idea:** Given a training example, find the direction that we should move parameters w to improve probability of word e_i

$$\boxed{\delta = \frac{d}{d\,\mathbf{w}} p\left(e_i \middle| e_{i-n+1}^{i-1}\right)}$$

(gradient of the probability)

- Move the parameters in that direction

$$\boxed{\mathbf{w} \leftarrow \mathbf{w} + \alpha\,\delta}$$

Problem with Linear Models: Cannot Deal with Feature Combinations

farmers eat steak	→	**high**	cows eat steak	→	**low**
farmers eat hay	→	**low**	cows eat hay	→	**high**

- Cannot express by just adding features. What do we do?
  - Remember scores for each combination of words

$$\begin{array}{l} \text{steak} \\ \text{hay} \\ \dots \end{array} \quad \boldsymbol{w}_{2,1,\text{farmers},\text{eat}} = \begin{pmatrix} 2.0 \\ -2.1 \\ \dots \end{pmatrix} \quad \boldsymbol{w}_{2,1,\text{cows},\text{eat}} = \begin{pmatrix} -1.2 \\ 2.9 \\ \dots \end{pmatrix}$$

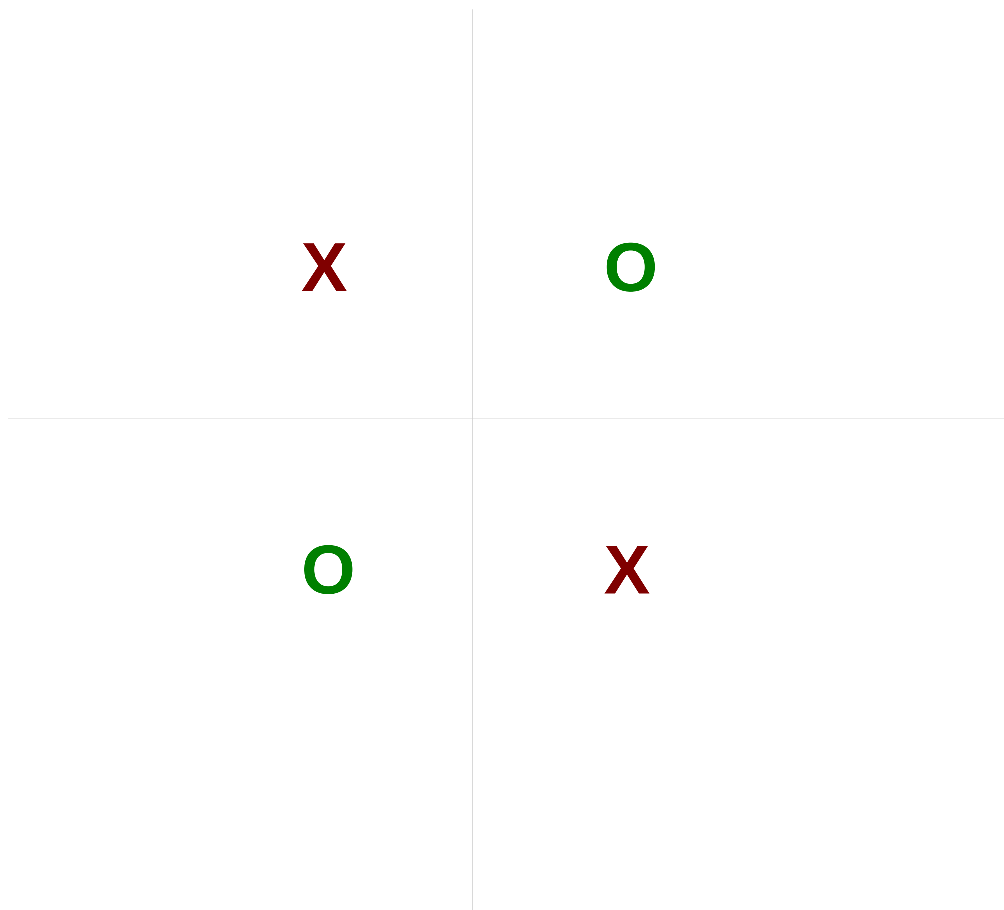
explosion in number of parameters, memory usage

- Neural nets!

Neural Networks

Problem: Can't learn Feature Combinations

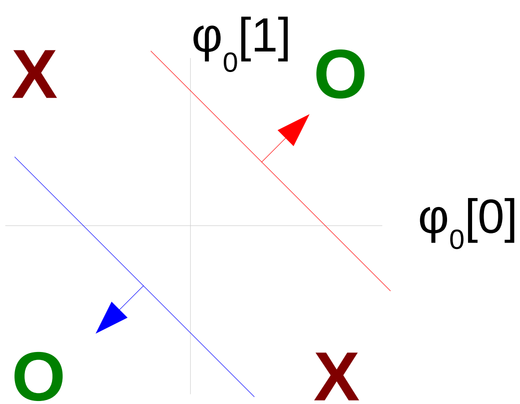
- Cannot achieve high accuracy on non-linear functions



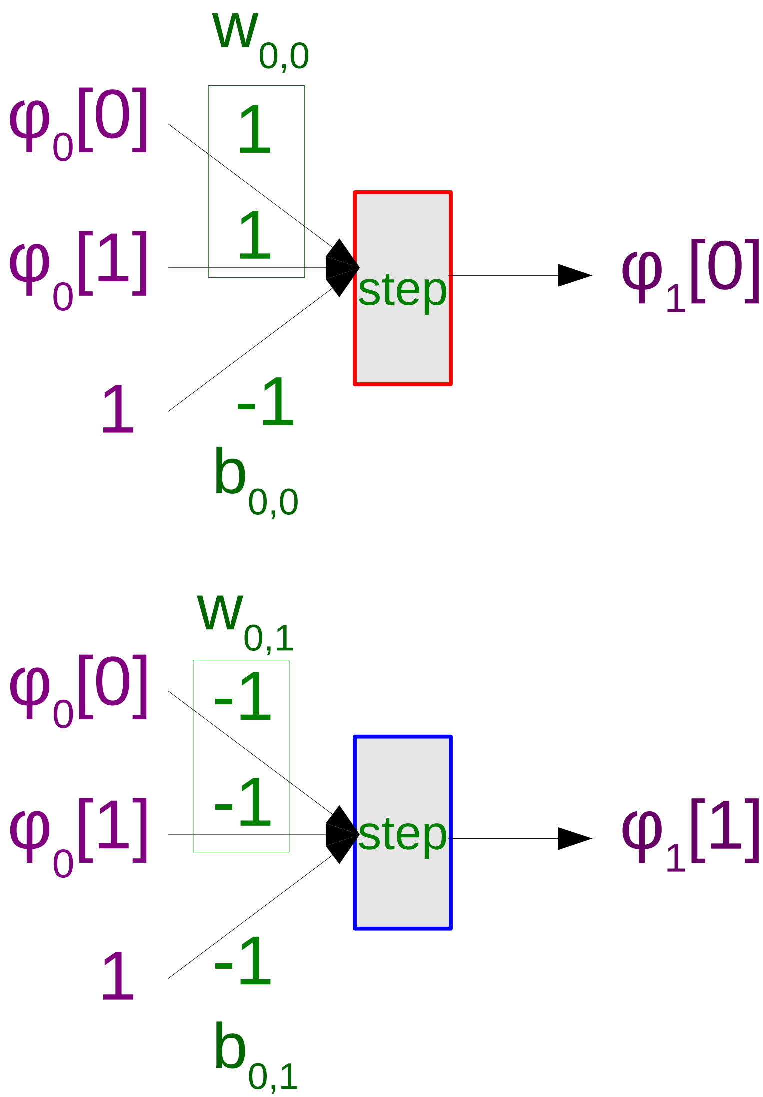
Solving Non-linear Classification

- Create two classifiers

$\varphi_0(x_1) = \{-1, 1\}$ $\varphi_0(x_2) = \{1, 1\}$

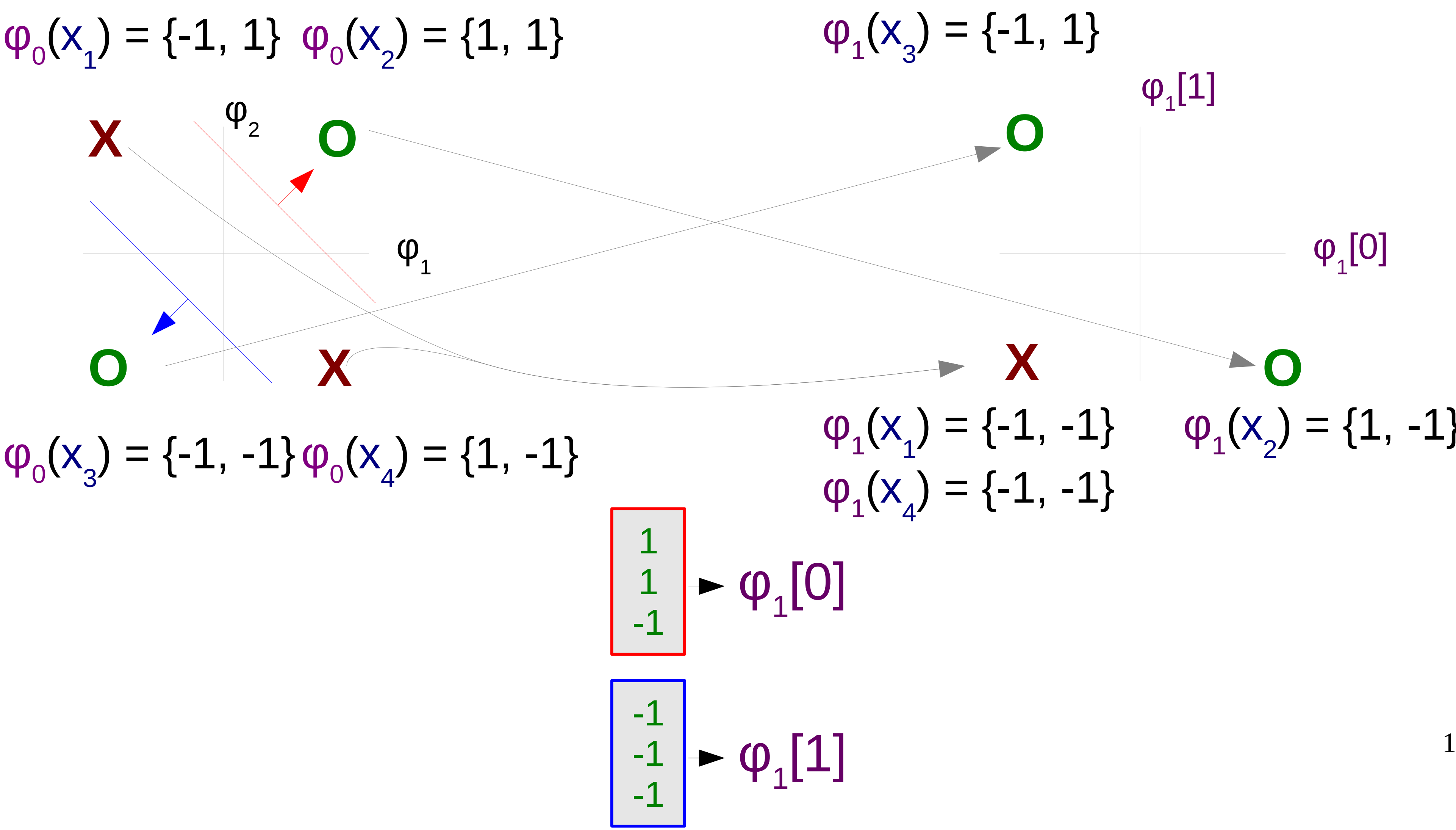


$\varphi_0(x_3) = \{-1, -1\}$ $\varphi_0(x_4) = \{1, -1\}$



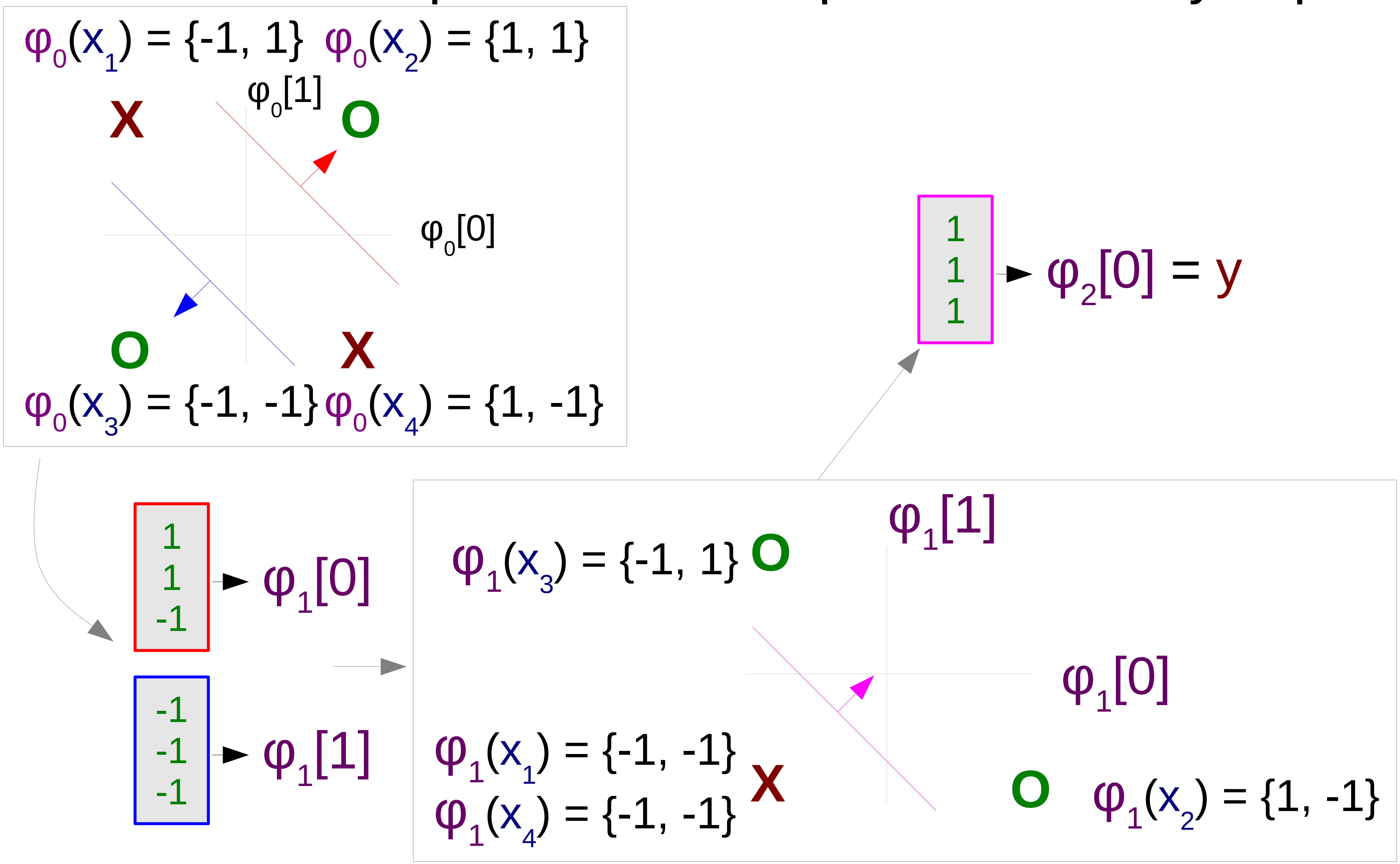
Example

- These classifiers map to a new space



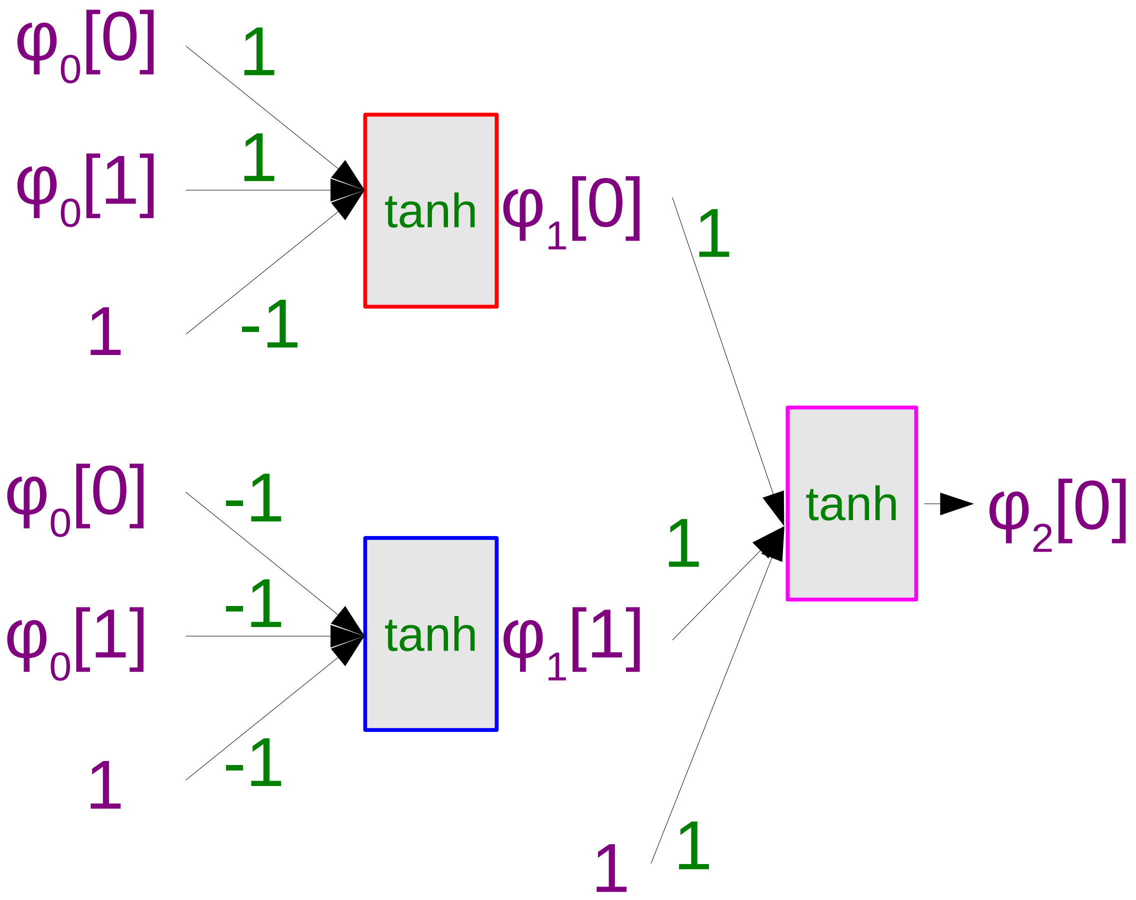
Example

- In the new space, the examples are linearly separable!



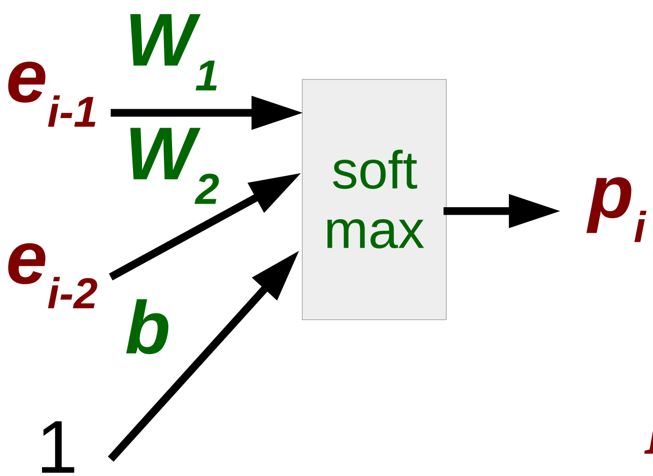
Example

- The final net



Language Modeling with Neural Nets

Overview of Log Linear Language Model



$$p_i = \text{softmax}\left(b + \sum_{k=1}^{n-1} W_k e_{i-k} \right)$$

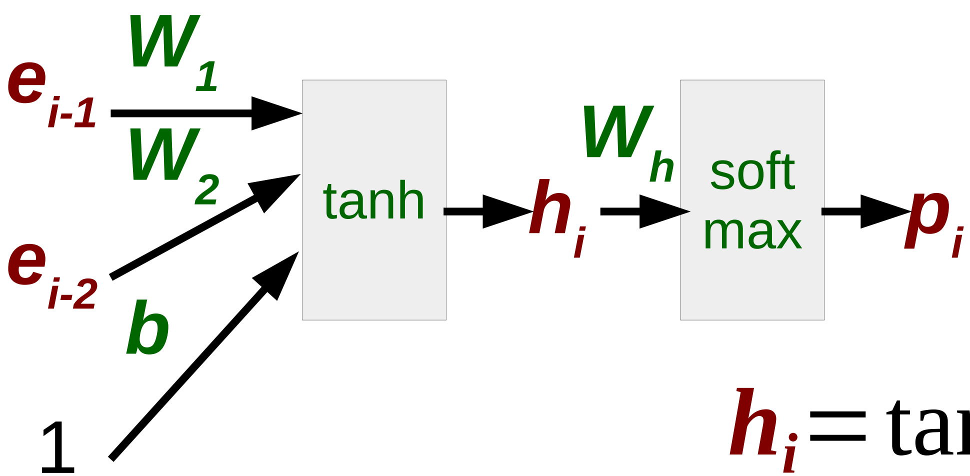
e_{i-1} and e_{i-2} are vectors where the element corresponding to the word is 1:

	a	the	talk	gift	giving	
e_{i-1} =	{1,	0,	0,	0,	0,	...}
e_{i-2} =	{0,	0,	0,	0,	1,	...}

W_1, W_2 are weight matrices b is a weight vector

Neural Network Language Model

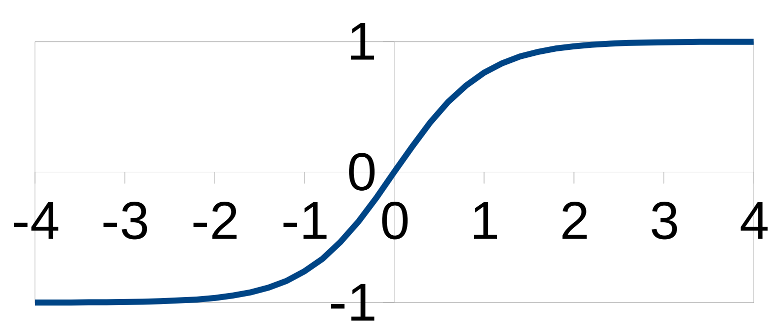
- Add a “hidden layer” that calculates representations



$$h_i = \tanh\left(b + \sum_{k=1}^{n-1} W_k e_{i-k}\right)$$

$$p_i = \mathrm{softmax}\left(W_h h_i\right)$$

tanh →



What Can We Do With Neural Nets?

- Learn shared “features” of the context
- Example: What do livestock eat?
  “ {cows, horses, sheep, goats} {eat, consume, ingest}”

$$\begin{array}{l} \text{eat} \\ \text{consume} \\ \text{ingest} \\ \text{cows} \\ \text{horses} \\ \text{sheep} \\ \text{goats} \\ \dots \end{array} \quad \boldsymbol{W_2}[1]=\begin{pmatrix} -1 \\ -1 \\ -1 \\ \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ \dots \end{pmatrix} \quad \boldsymbol{W_1}[1]=\begin{pmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ -1 \\ -1 \\ -1 \\ -1 \\ \dots \end{pmatrix} \quad \boldsymbol{b}[1]=-1$$

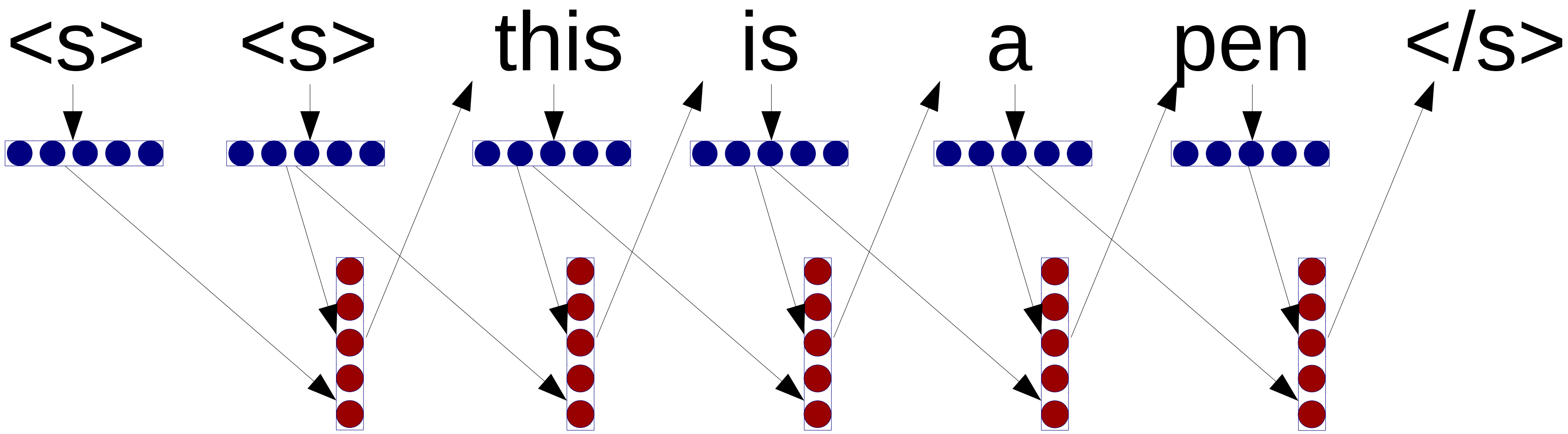
cows eat → tanh(1)

men eat → tanh(-1)

cows find → tanh(-1)

- If both are true, positive number, otherwise negative
- Simple features must remember all 4x3 combinations!

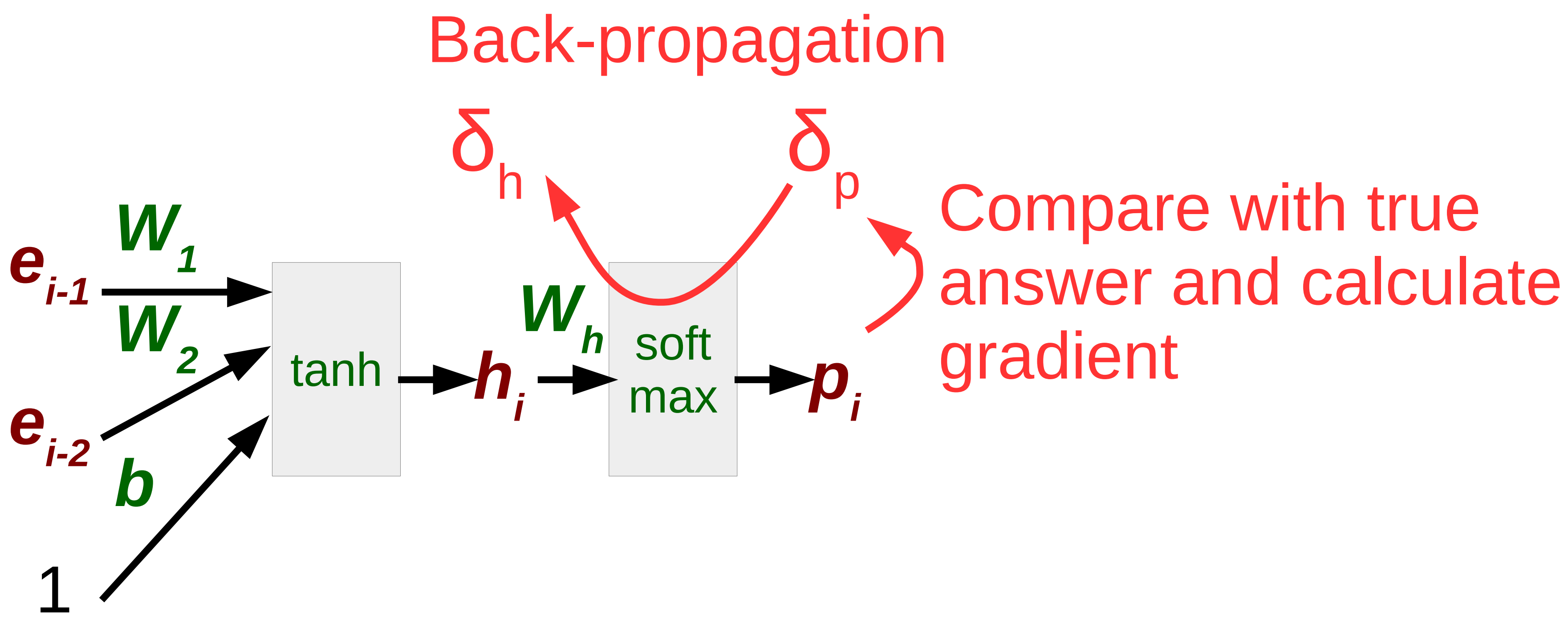
Neural Network Language Model [Nakamura+ 90, Bengio+ 06]



- Convert each word into word representation, considering word similarity
- Convert the context into low-dimensional hidden layer, considering contextual similarity

Learning Neural Networks: Back Propagation

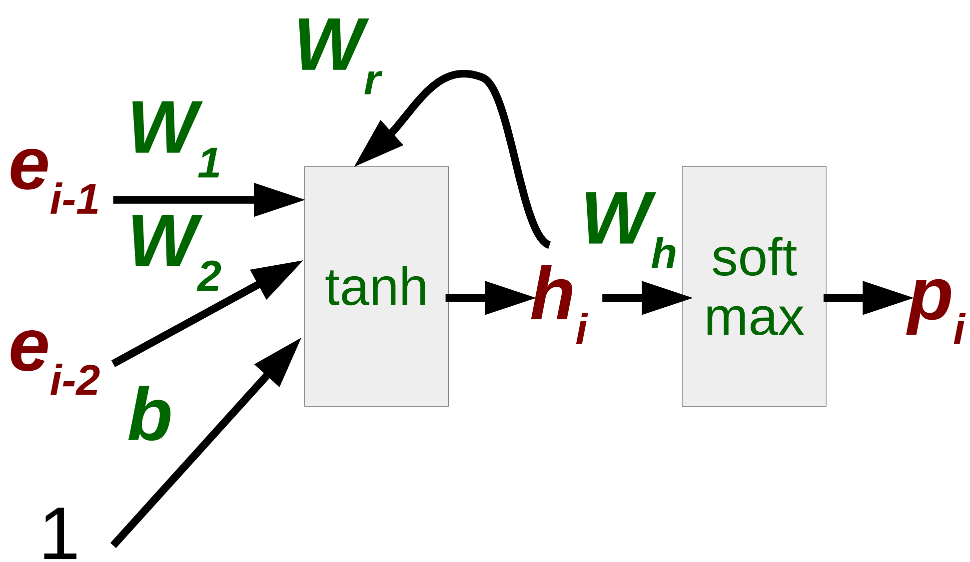
- Calculate the direction the last layer needs to go in
- Pass this information backwards through the network



Recurrent Neural Nets

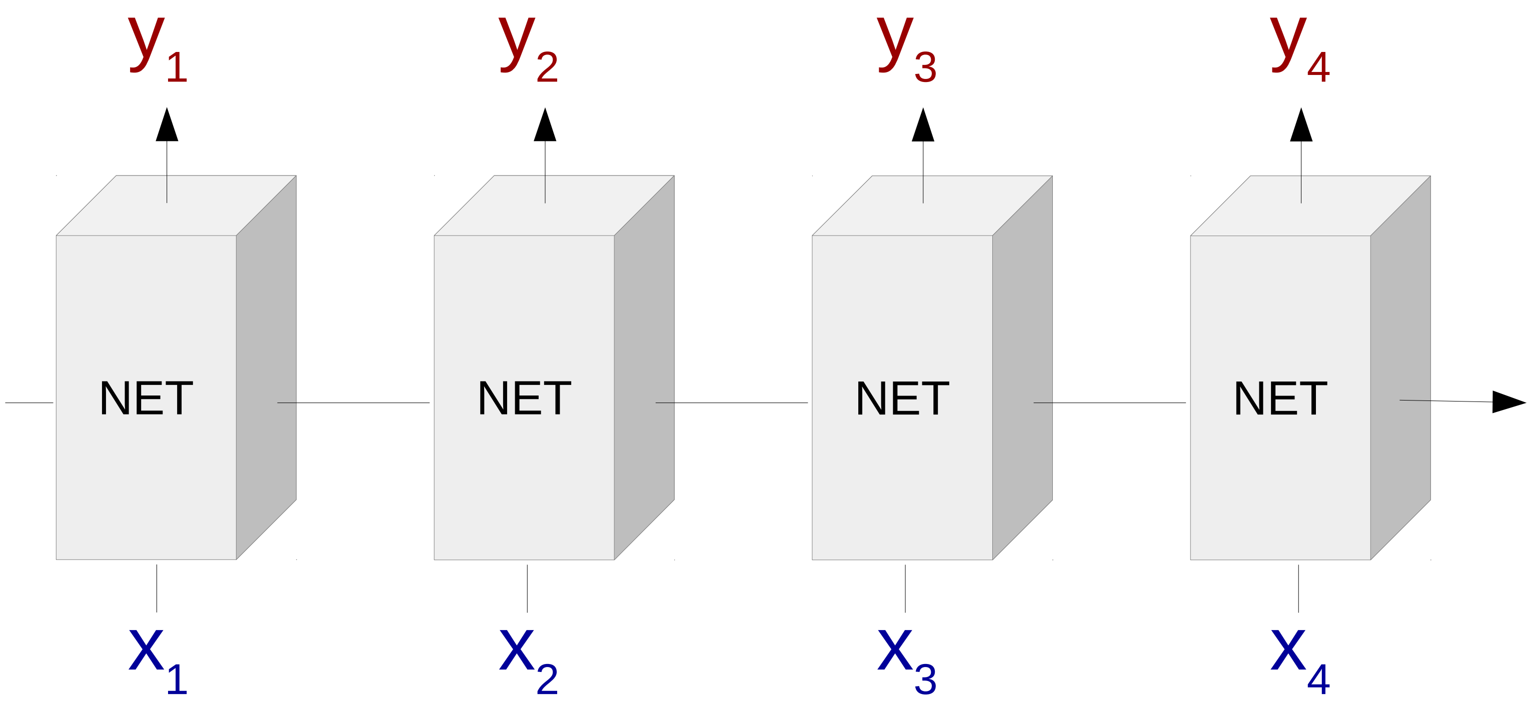
Recurrent Neural Nets (RNN)

- Pass output of the hidden layer from the last time step back to the hidden layer in the next time step

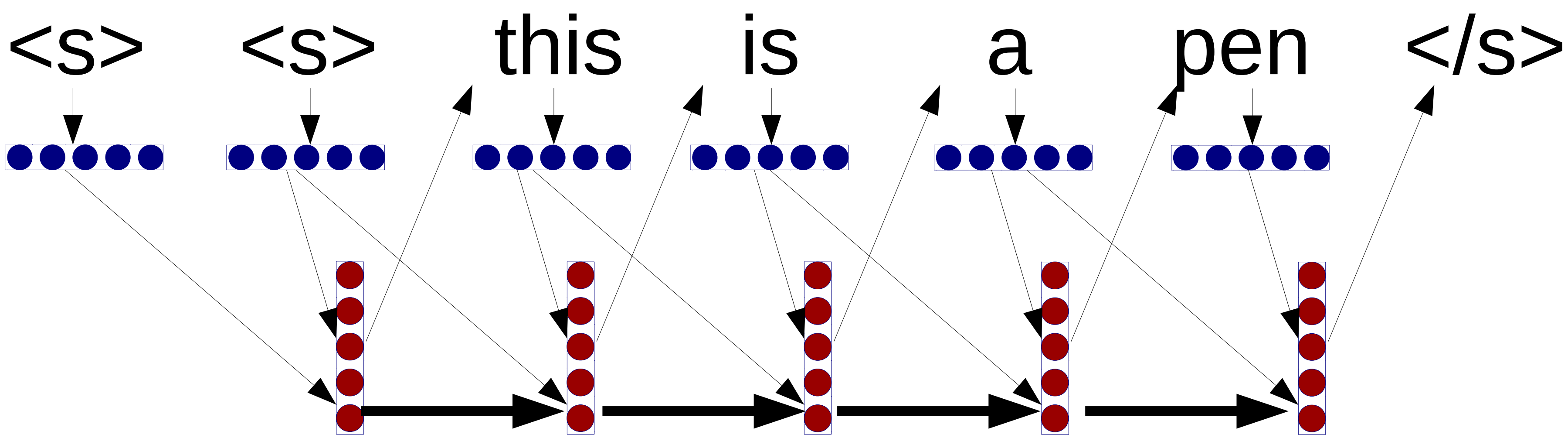


- Why?: Can remember long distance dependencies
- Example:
  - He doesn't have very much confidence in himself
  - She doesn't have very much confidence in herself

RNNs as Sequence Models

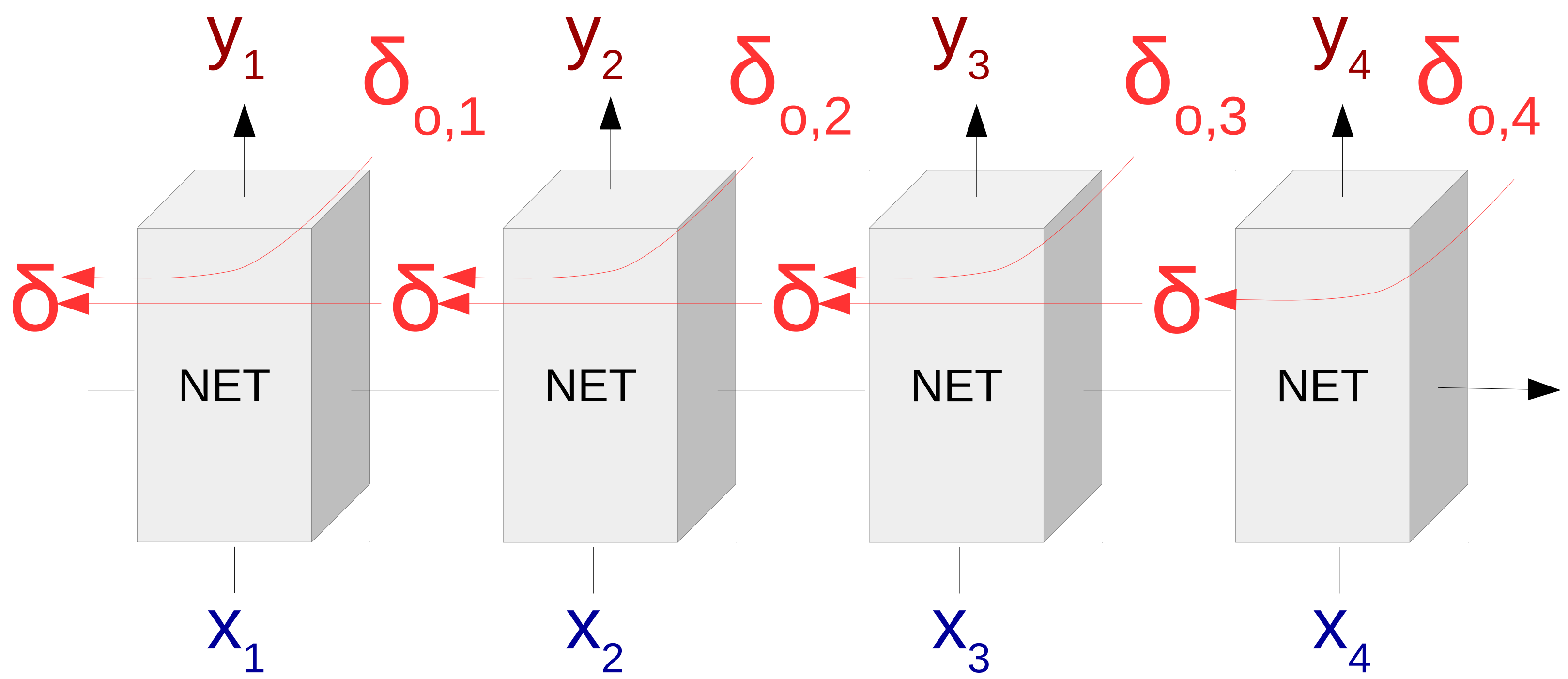


Recurrent Neural Network Language Model [Mikolov+ 10]



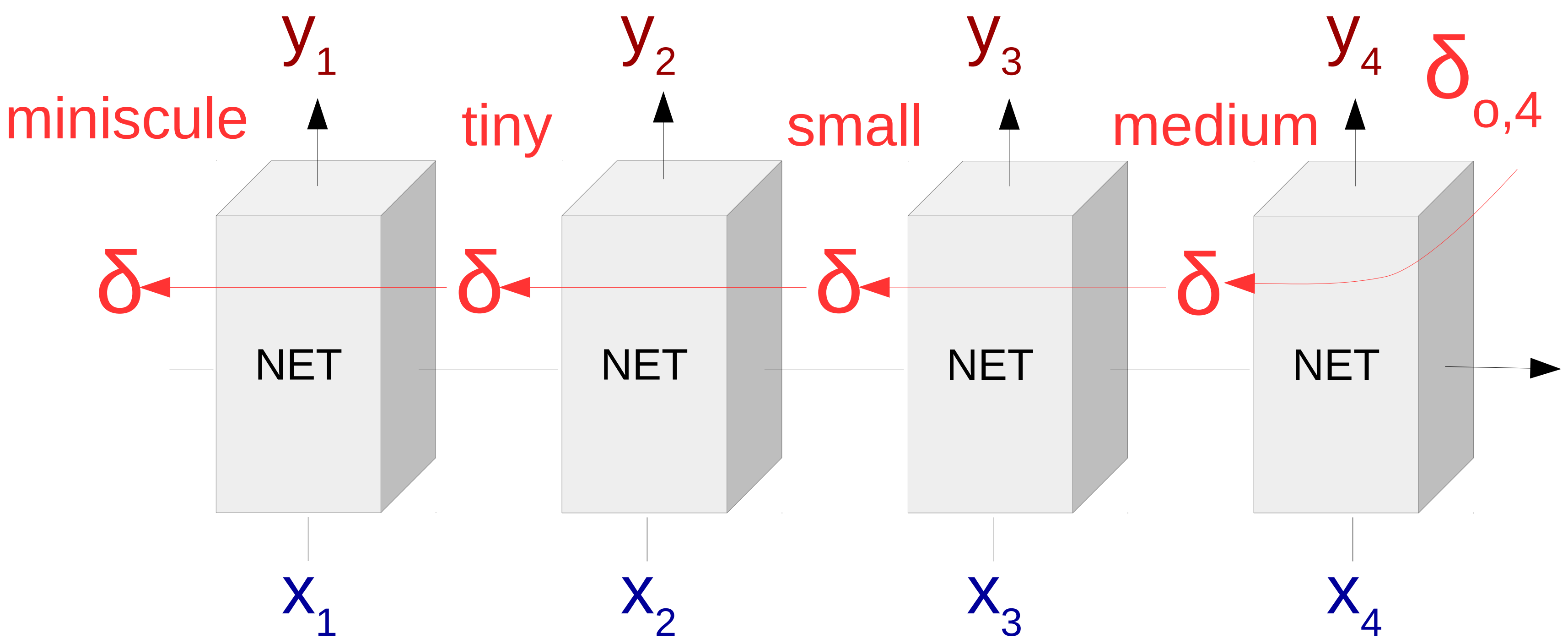
- Greatly improves accuracy of machine translation, speech recognition, etc.

Calculating Gradients for RNNs



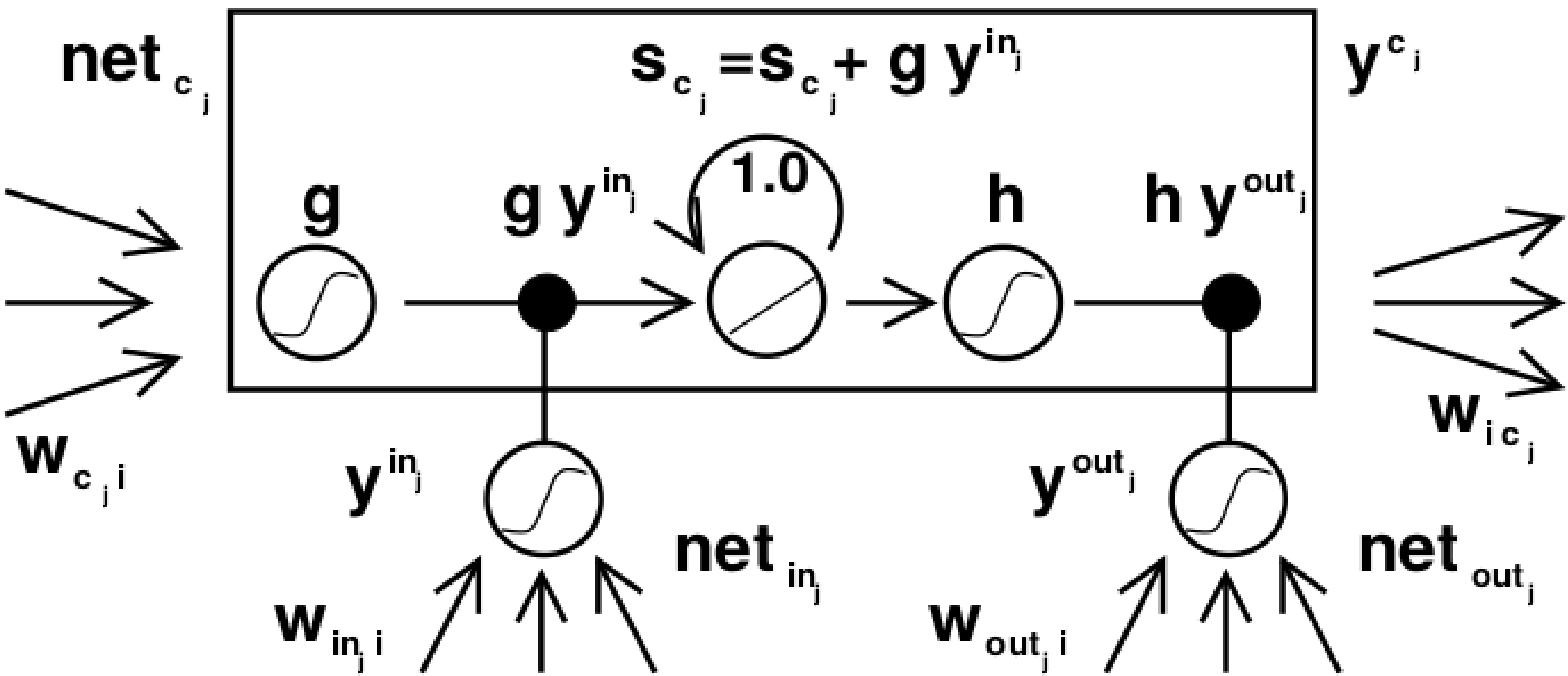
- First, calculate values forward
- Propagate errors backward

The Vanishing Gradient Problem

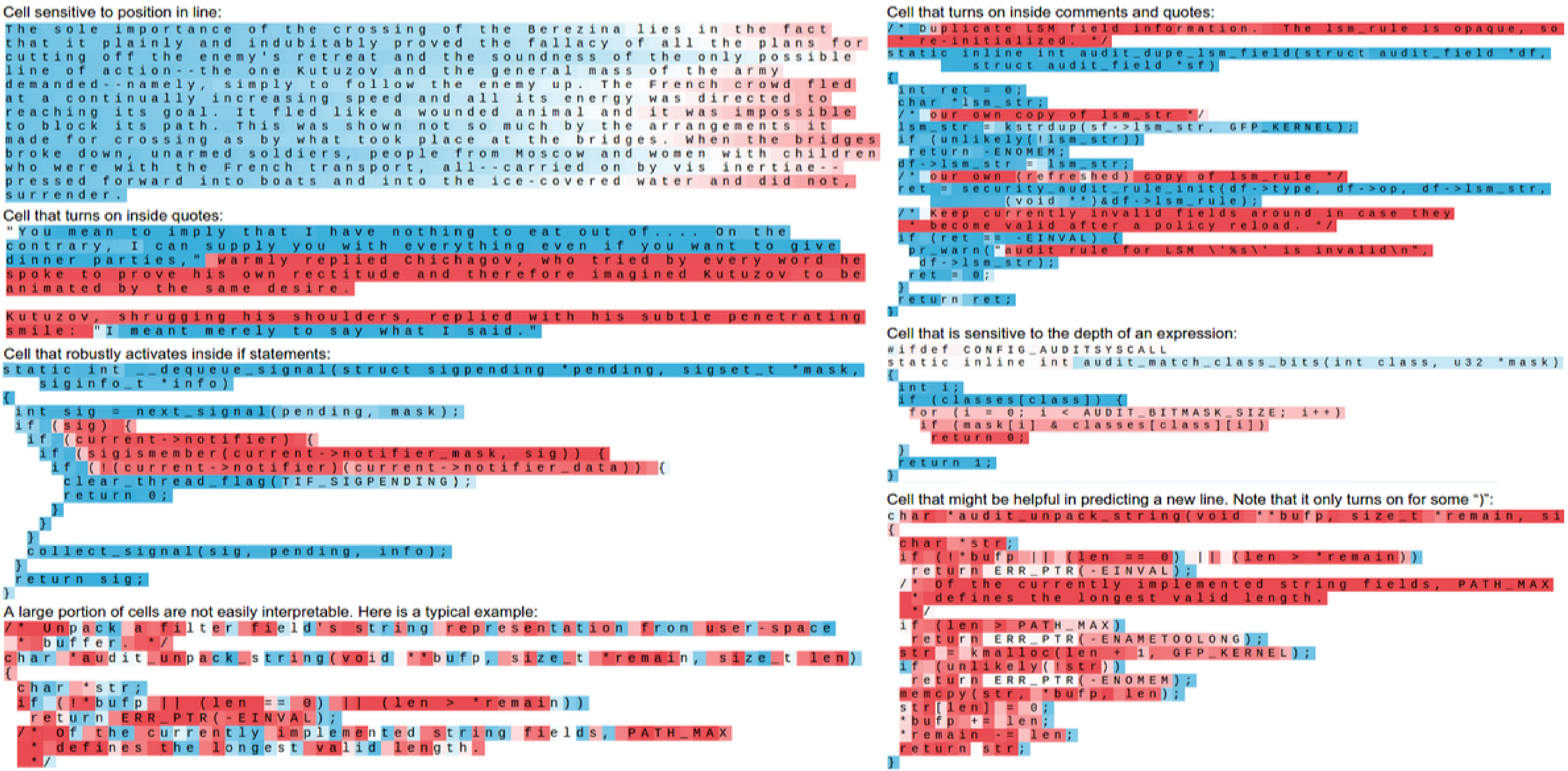


Long Short-term Memory [Hochreiter+ 97]

- Based on a linear function that preserves previous values
- Gating structure to control information flow



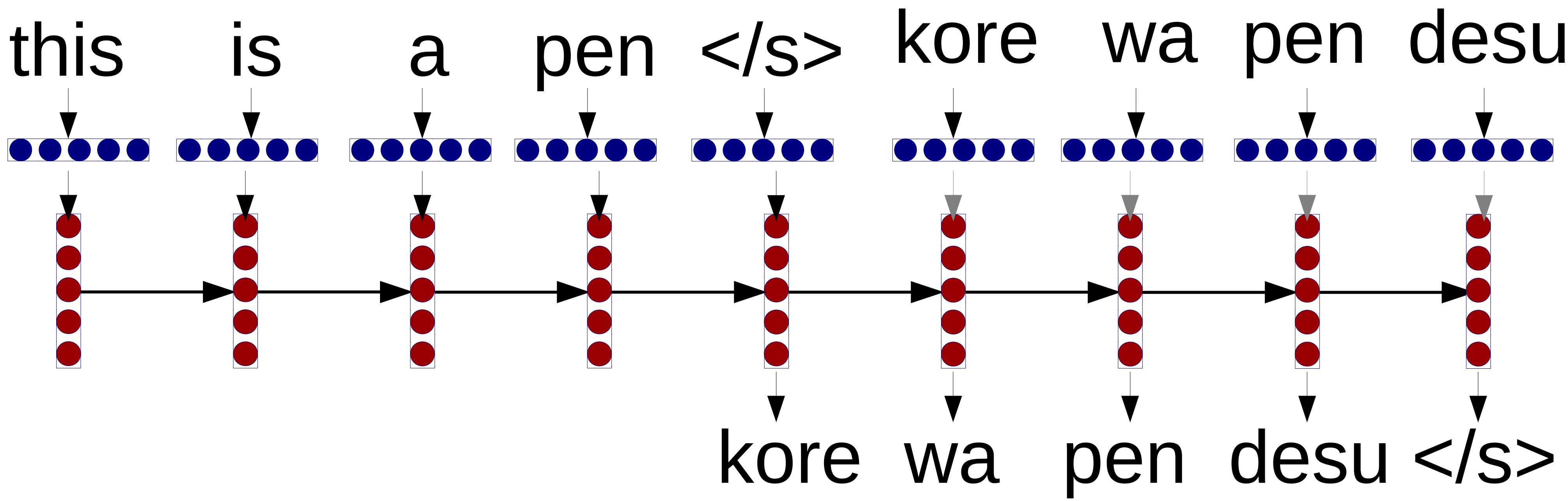
What Can a Recurrent Network Learn? [Karpathy+ 15]



Encoder-Decoder Translation Model

[Kalchbrenner+ 13, Sutskever+ 14]

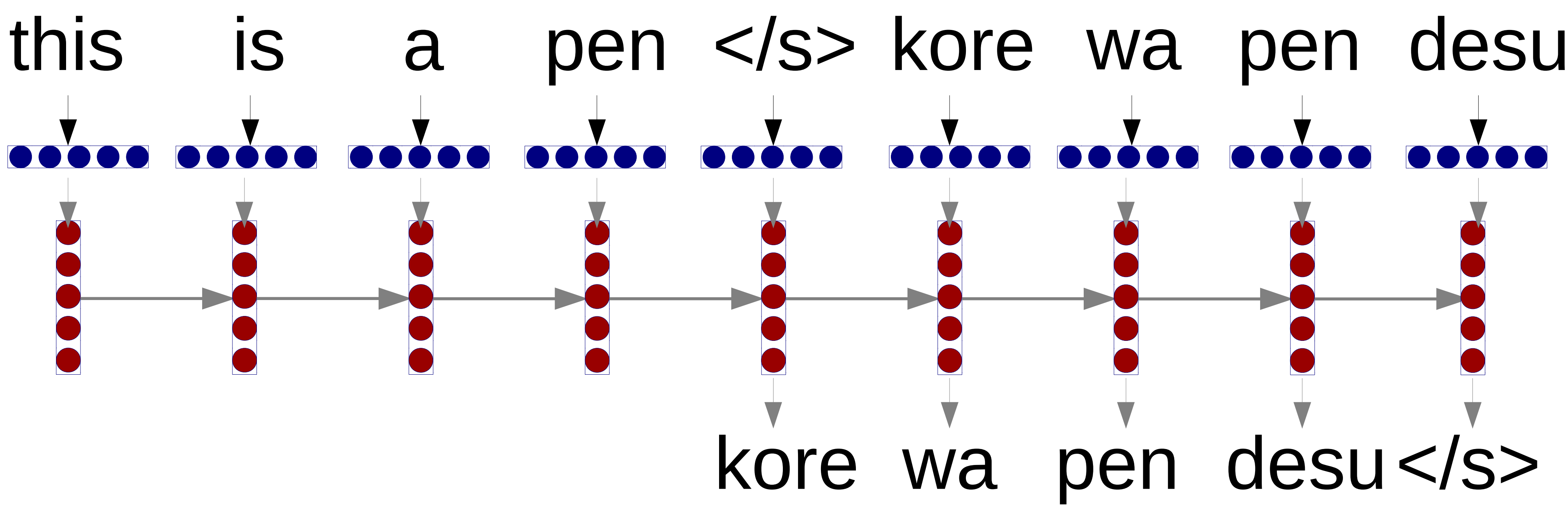
Recurrent NN Encoder-Decoder Model [Sutskever+ 14]



- In other words, exactly like RNN language model, but first “reads” the input sentence

$$P(e_1^I | f_1^J) = \prod_{i=1}^{I+1} P(e_i | f_1^J, e_1^{i-1})$$

Example of Generation



Read the input

Write the output

$$\operatorname{argmax}_{e_i} P\left(e_i \middle| f_1^J, e_1^{i-1}\right)$$

So, How Well Does It Work?

Method	BLEU
Phrase-based Baseline	33.30
Encoder-Decoder	26.17
Encoder-Decoder w/ Tricks	34.81

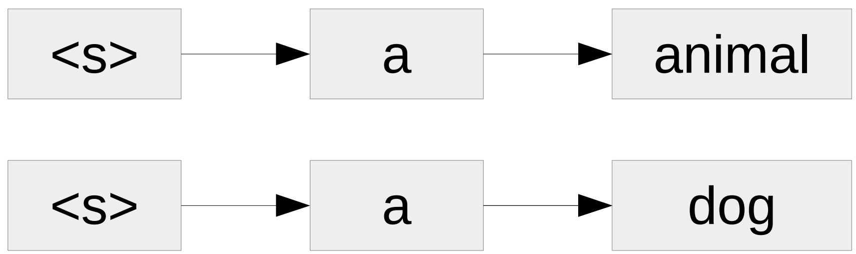
[Sutskever et al. 2014]

Answer: competitive with strong phrase-based traditional systems! (With a little work...)

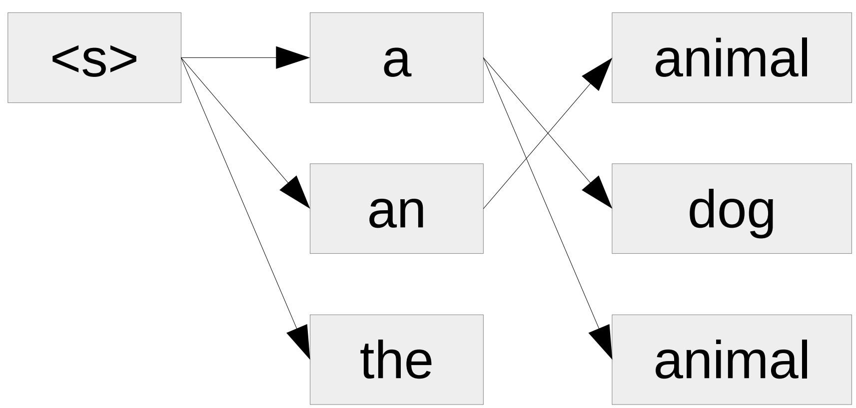
Trick 1: Beam Search

- **Greedy search:** select one-best at every time step
  - **Problem:** locally optimal decisions not globally optimal

Ref:
“an animal”



- **Beam search:** maintain several hypotheses every step



Trick 2: Ensembling

- Average two models together (in regular or log space)

	Model 1	**Model 2**		**Model 1+2**
a the talk gift hat …	$\boldsymbol{p_1} = \begin{pmatrix} 0.002 \\ 0.003 \\ 0.329 \\ \mathbf{0.444} \\ 0.090 \\ \dots \end{pmatrix}$	$\boldsymbol{p_2} = \begin{pmatrix} 0.030 \\ 0.021 \\ 0.410 \\ 0.065 \\ \mathbf{0.440} \\ \dots \end{pmatrix}$	$\longrightarrow$	$\boldsymbol{p_e} = \begin{pmatrix} 0.016 \\ 0.012 \\ \mathbf{0.370} \\ 0.260 \\ 0.265 \\ \dots \end{pmatrix}$

- Why does this work?
  - Errors tend to be **uncorrelated**
  - Errors tend to be **less confident**

Small Example on Japanese-English

- Trained on 116k short, conversational sentences

	BLEU	RIBES
Moses PBMT	38.6	80.3
Encoder-Decoder	39.0	82.9

Does it Stand Up to Manual Inspection?

Answer: Yes, to some extent

Input: バスタブからお湯があふれてしまいました。
True: the hot water overflowed from the bathtub .
PBMT: the hot water up the bathtub .
EncDec: the bathtub has overflowed .

Input: コーヒーのクリーム入りをください。
True: i 'll have some coffee with cream , please .
PBMT: cream of coffee , please .
EncDec: i 'd like some coffee with cream .

But, There are Problems.

Giving up:

Input: ギブス を し な けれ ば な り ま せ ん 。
True: you 'll have to have a cast .
PBMT: i have a ギブス .
EncDec: you have to have a chance .

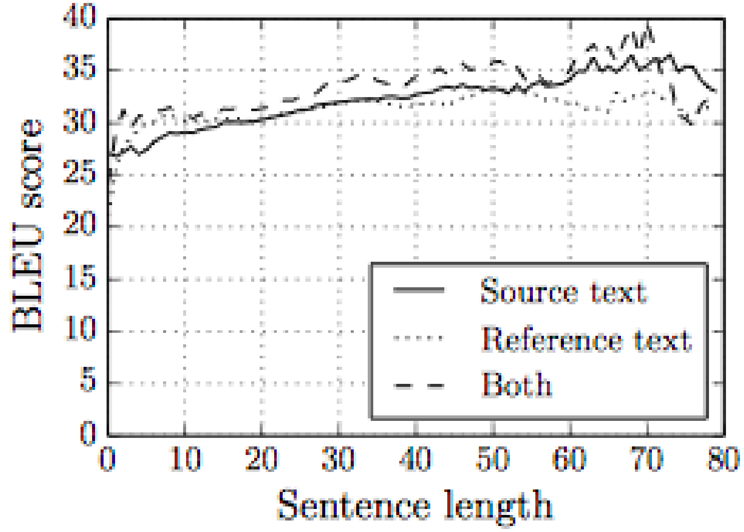
Repeating:

Input: どのファンデーションが私の肌の色に近いですか。
True: which foundation comes close to my natural skin color ?
PBMT: which foundation near my natural skin color ?
EncDec: which foundation is my favorite foundation with a foundation ?

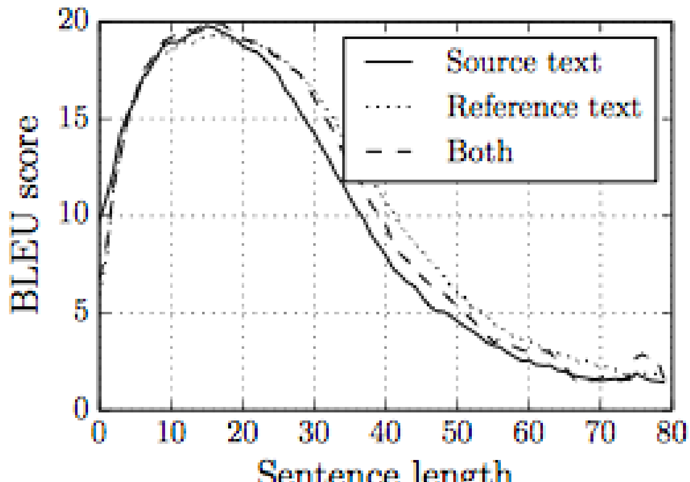
Attentional Models

Problem: Encoder-Decoder Models have Trouble with Longer Sentences

PBMT



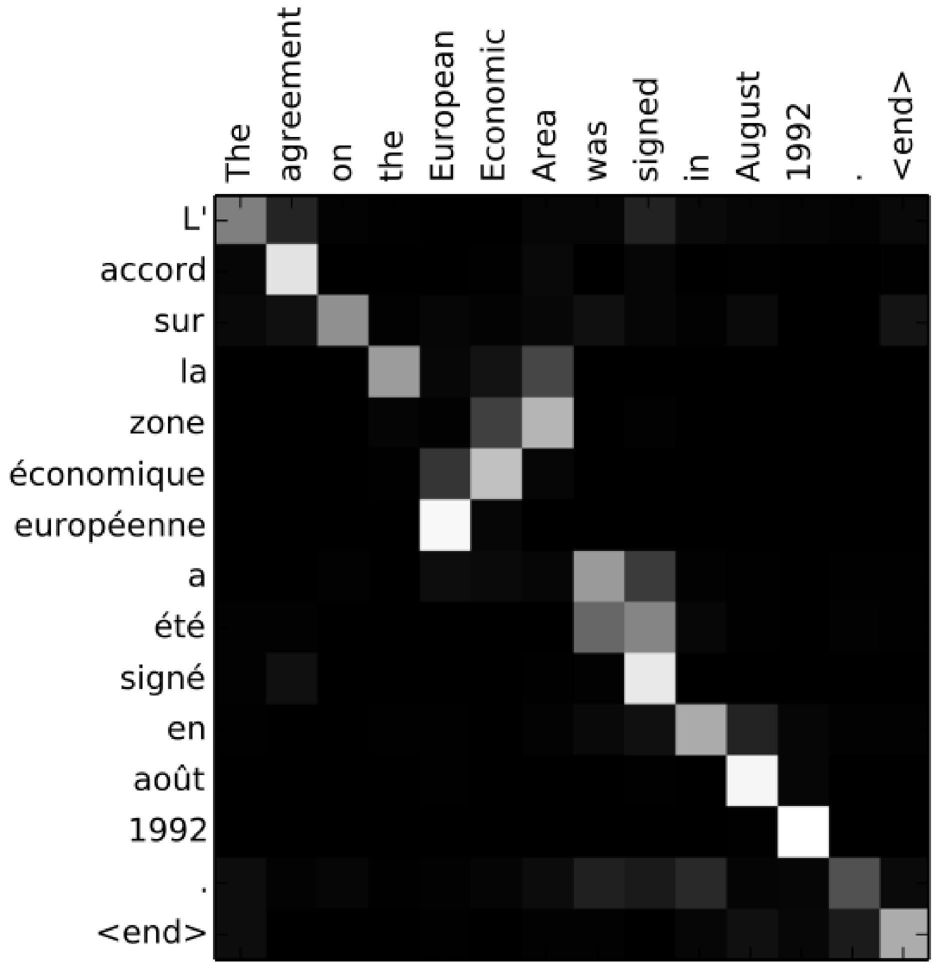
RNN



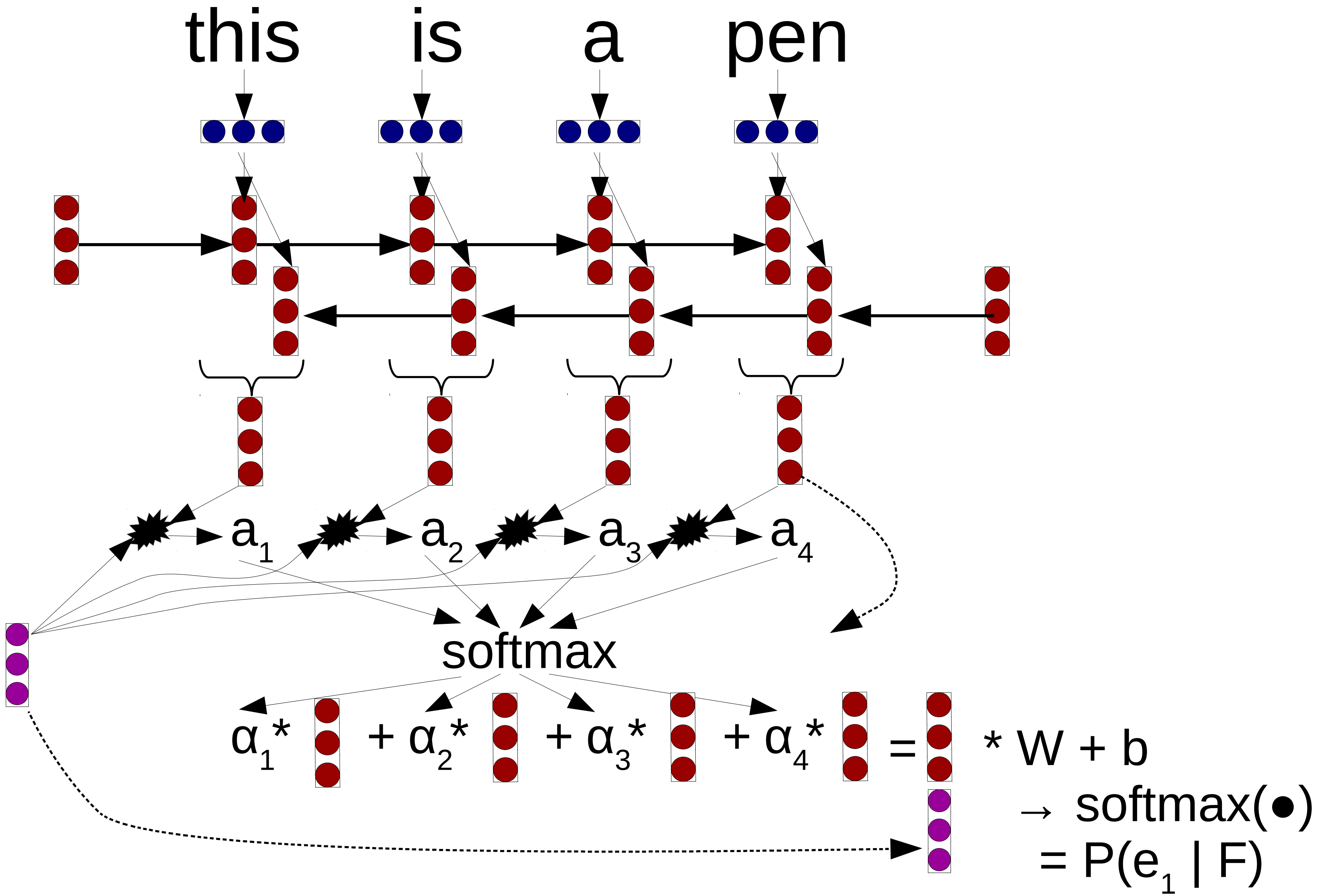
[Pouget-Abadie+ 2014]

Attentional Nets [Bahdanau+ 15]

- While translating, decide which word to “focus” on



Looking Carefully at (One Step of) Attention



Exciting Results!

- IWSLT 2015:
  Best results on de-en
- WMT 2016:
  Best results on most language pairs
- WAT 2016:
  Best results on most language pairs
  (NAIST/CMU model 1st on ja-en)

What Has Gotten Better? Largely Grammar [Bentivogli+ 16]

Auxiliary-main verb construction [aux:V]:

	SRC	in this experiment , individuals **were shown** hundreds of hours of YouTube videos	
	HPB	in diesem Experiment , Individuen **gezeigt wurden** Hunderte von Stunden YouTube-Videos	✗
(a)	PE	in diesem Experiment **wurden** Individuen Hunderte von Stunden Youtube-Videos **gezeigt**	
	NMT	in diesem Experiment **wurden** Individuen hunderte Stunden YouTube Videos **gezeigt**	✓
	PE	in diesem Experiment **wurden** Individuen hunderte Stunden YouTube Videos **gezeigt**	

Verb in subordinate (adjunct) clause [neb:V]:

	SRC	... when coaches and managers and owners **look** at this information streaming ...	
	PBSY	... wenn Trainer und Manager und Eigentümer **betrachten** diese Information Streaming ...	✗
(b)	PE	... wenn Trainer und Manager und Eigentümer dieses Informations-Streaming **betrachten** ...	
	NMT	... wenn Trainer und Manager und Besitzer sich diese Informationen **anschauen** ...	✓
	PE	... wenn Trainer und Manager und Besitzer sich diese Informationen **anschauen** ...	

Prepositional phrase [pp:PREP det:ART pn:N] acting as temporal adjunct:

	SRC	so like many of us , I 've lived in a few closets **in my life**	
	SPB	so wie viele von uns , ich habe in ein paar Schränke **in meinem Leben** gelebt	✗
(c)	PE	so habe ich wie viele von uns **während meines Lebens** in einigen Verstecken gelebt	
	NMT	wie viele von uns habe ich in ein paar Schränke **in meinem Leben** gelebt	✗
	PE	wie viele von uns habe ich **in meinem Leben** in ein paar Schränken gelebt	

Negation particle [adv:PTKNEG]:

	SRC	but I eventually came to the conclusion that that just did **not** work for systematic reasons	
	HPB	aber ich kam schlielich zu dem Schluss , dass nur aus systematischen Gründen **nicht** funktionieren	✓
(d)	PE	aber ich kam schlielich zu dem Schluss , dass es einfach aus systematischen Gründen **nicht** funktioniert	
	NMT	aber letztendlich kam ich zu dem Schluss , dass das einfach **nicht** aus systematischen Gründen funktionierte	✗
	PE	ich musste aber einsehen , dass das aus systematischen Gründen **nicht** funktioniert	

Other Things to Think About

What is Our Training Criterion?

- We train our models for likelihood
- Evaluate our models based on quality of the generated sentences (BLEU)
- How do we directly optimize for translation quality?
    - Reinforcement learning [Ranzato+16]
    - Minimum risk training [Shen+16]
    - Beam search optimization [Wiseman+16]

How Do We Handle Rare Words?

- Neural MT has trouble with large vocabularies
  - **Speed:** Takes time to do a big softmax
  - **Accuracy:** Fail on less common training examples
- Solutions:
  - Sampling-based training methods [Mnih+12]
  - Translate using subword units [Sennrich+16]
  - Translate using characters [Chung+16]
  - Incorporate translation lexicons [Arthur+16]

Can We Train Multi-lingual Models?

- Multi-lingual data abounds, and we would like to use it
- Methods:
  - Train individual encoders/decoders for each language, but share training [Firat+16]
  - Train a single encoder/decoder for all languages [Johnson+16]
  - Transfer models from one language to another [Zoph+16]

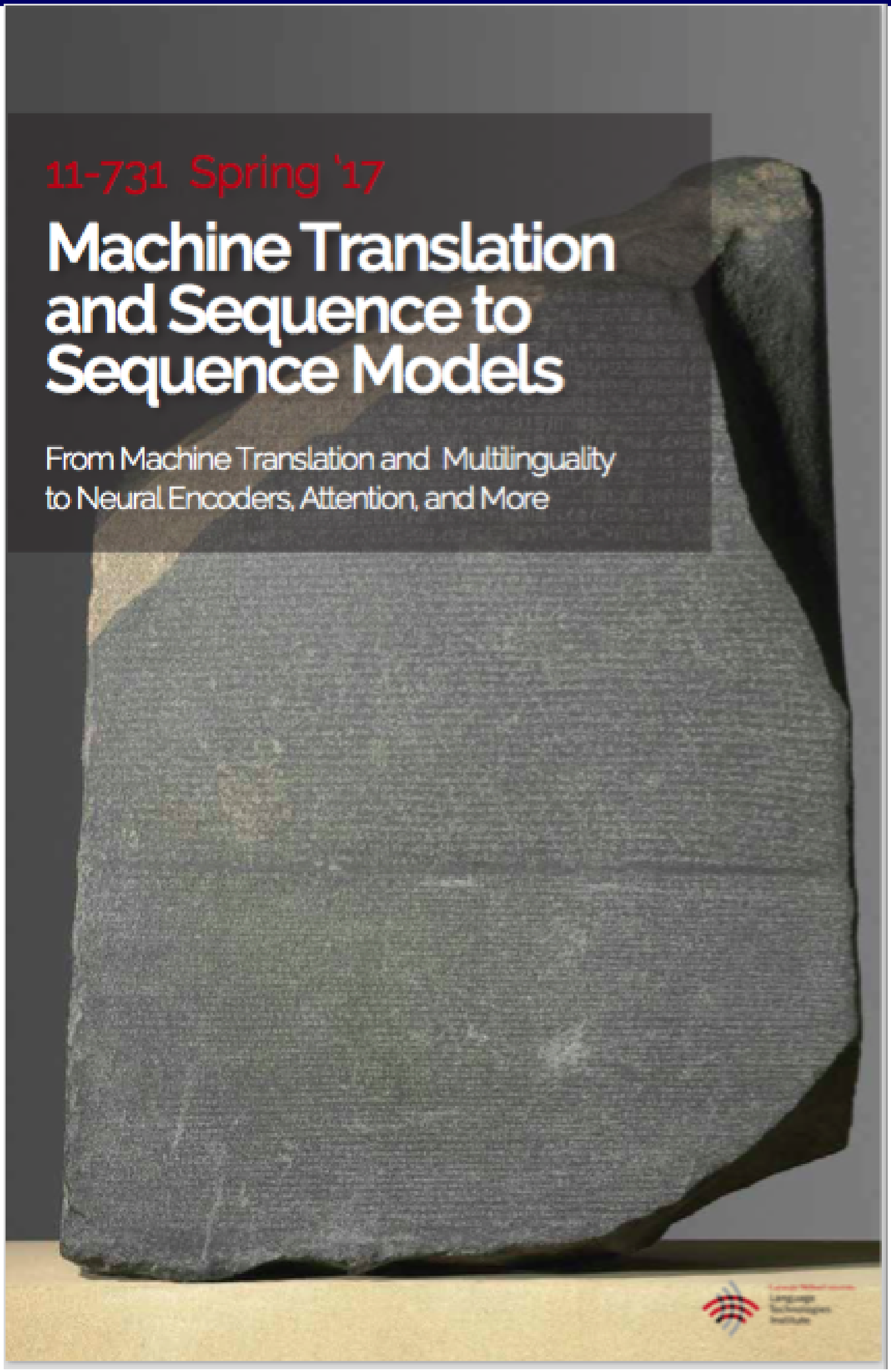
What Else Can We Do?

- Conversation [Sordoni+ 15, Vinyals+ 15]
  Input: utterance, Output: next utterance
- Executing programs [Zaremba+ 14]
  Input: program, Output: computation result
- And many others!

Conclusion/Tutorials/Papers

Conclusion

- Neural MT is exciting!



Tutorials

- My Neural MT Tips Tutorial: https://github.com/neubig/nmt-tips
- Kyunghyun Cho's DL4MT Tutorial: http://github.com/nyu-dl/dl4mt-tutorial
- Thang Luong, Kyunghyun Cho, and Chris Manning's Tutorial at ACL 2016: https://sites.google.com/site/acl16nmt/
- Rico Sennrich's Tutorial at AMTA 2016: http://statmt.org/mtma16/uploads/mtma16-neural.pdf

References

- P. Arthur, G. Neubig, and S. Nakamura. Incorporating discrete translation lexicons into neural machine translation. In Proc. EMNLP, 2016.
- D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Proc. ICLR, 2015.
- Y. Bengio, H. Schwenk, J.-S. Sen´ecal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In Innovations in Machine Learning, volume 194, pages 137–186. 2006.
- S. F. Chen and R. Rosenfeld. A survey of smoothing techniques for me models. Speech and Audio Processing, IEEE Transactions on, 8(1):37–50, Jan 2000.
- J. Chung, K. Cho, and Y. Bengio. A character-level decoder without explicit segmentation for neural machine translation. arXiv preprint arXiv:1603.06147,
- 2016.
- O. Firat, K. Cho, and Y. Bengio. Multi-way, multilingual neural machine translation with a shared attention mechanism. In Proc. NAACL, pages 866–875, 2016.
- S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.
- S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. In Proc. ACL, pages 1–10, 2015.
- M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Vi´egas, M. Wattenberg, G. Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. arXiv preprint arXiv:1611.04558, 2016.
- N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In Proc. EMNLP, pages 1700–1709, Seattle, Washington, USA, 2013. Association for Computational Linguistics.
- M.-T. Luong, I. Sutskever, Q. Le, O. Vinyals, and W. Zaremba. Addressing the rare word problem in neural machine translation. In Proc. ACL, pages 11–19, 2015.
- T. Luong, R. Socher, and C. Manning. Better word representations with recursive neural networks for morphology. In Proc. CoNLL, pages 104–113, 2013.

References

- T. Mikolov, M. Karafi´at, L. Burget, J. Cernocky`, and S. Khudanpur. Recurrent neural network based language model. In Proc. InterSpeech, pages 1045–1048,
- 2010.
- A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. arXiv preprint arXiv:1206.6426, 2012.
- M. Nakamura, K. Maruyama, T. Kawabata, and K. Shikano. Neural network approach to word category prediction for English texts. In Proc. COLING, 1990.
- M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. Proc. ICLR, 2016.
- R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In Proc. ACL, pages 1715–1725, 2016.
- S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu. Minimum risk training for neural machine translation. In Proc. ACL, pages 1683–1692, 2016.
- R. Socher, C. C. Lin, C. Manning, and A. Y. Ng. Parsing natural scenes and natural language with recursive neural networks. pages 129–136, 2011.
- I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Proc. NIPS, pages 3104–3112, 2014.
- A. Vaswani, Y. Zhao, V. Fossum, and D. Chiang. Decoding with large-scale neural language models improves translation. In Proc. EMNLP, pages 1387–1392, 2013.
- S. Wiseman and A. M. Rush. Sequence-to-sequence learning as beam-search optimization. In Proc. EMNLP, pages 1296–1306, 2016.
- B. Zoph, D. Yuret, J. May, and K. Knight. Transfer learning for low-resource neural machine translation. In Proc. EMNLP, pages 1568–1575, 2016.