Математические методы анализа текстов

Introduction to machine translation

Потапенко Анна Александровна

14 ноября 2018 г.

Machine Translation

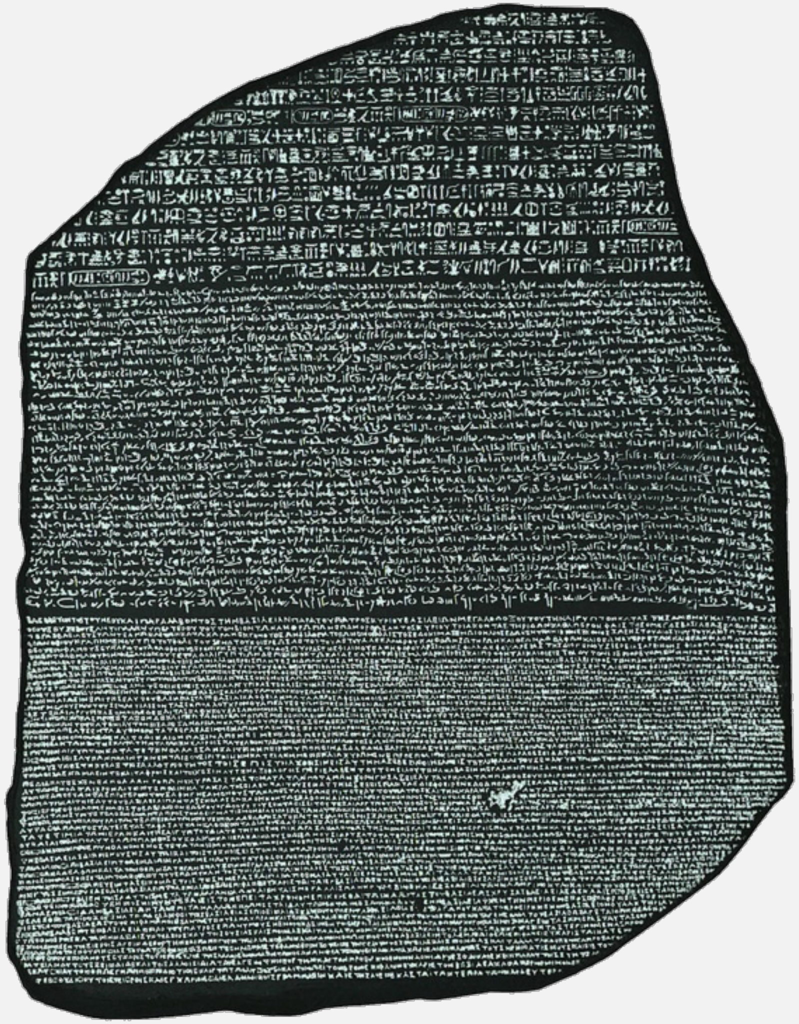
Parallel data

Parallel corpora:

- Europarl
- Movie subtitles
- Translated news, books
- Wikipedia (comparable)
- http://opus.lingfil.uu.se/

Lot’s of problems with data:

- Noisy
- Specific domain
- Rare language pairs
- Not aligned, not enough



Evaluation

- How to compare two arbitrary translations?
- Low agreement rate even between reviewers
- BLEU score – a popular automatic technique

Evaluation

- How to compare two arbitrary translations?
- Low agreement rate even between reviewers
- BLEU score – a popular automatic technique

Reference: ***E-mail was sent on Tuesday.***

System output: ***The letter was sent on Tuesday.***

Evaluation

- How to compare two arbitrary translations?
- Low agreement rate even between reviewers
- BLEU score – a popular automatic technique

Reference: ***E-mail was sent on Tuesday.***

System output: ***The letter was sent on Tuesday.***

1-grams: 4 / 6

Evaluation

- How to compare two arbitrary translations?
- Low agreement rate even between reviewers
- BLEU score – a popular automatic technique

Reference: ***E-mail was sent on Tuesday.***

System output: ***The letter was sent on Tuesday.***

1-grams: 4 / 6

2-grams: 3 / 5

Evaluation

- How to compare two arbitrary translations?
- Low agreement rate even between reviewers
- BLEU score – a popular automatic technique

Reference: ***E-mail was sent on Tuesday.***

System output: ***The letter was sent on Tuesday.***

1-grams: 4 / 6

2-grams: 3 / 5

3-grams: 2 / 4

4-grams: 1 / 3

Evaluation

- How to compare two arbitrary translations?
- Low agreement rate even between reviewers
- BLEU score – a popular automatic technique

Reference: ***E-mail was sent on Tuesday.***

System output: ***The letter was sent on Tuesday.***

1-grams: 4 / 6

2-grams: 3 / 5

3-grams: 2 / 4

4-grams: 1 / 3

Brevity penalty : min(1, 6 / 5)

Evaluation

- How to compare two arbitrary translations?
- Low agreement rate even between reviewers
- BLEU score – a popular automatic technique

Reference: ***E-mail was sent on Tuesday.***

System output: ***The letter was sent on Tuesday.***

1-grams: 4 / 6

2-grams: 3 / 5

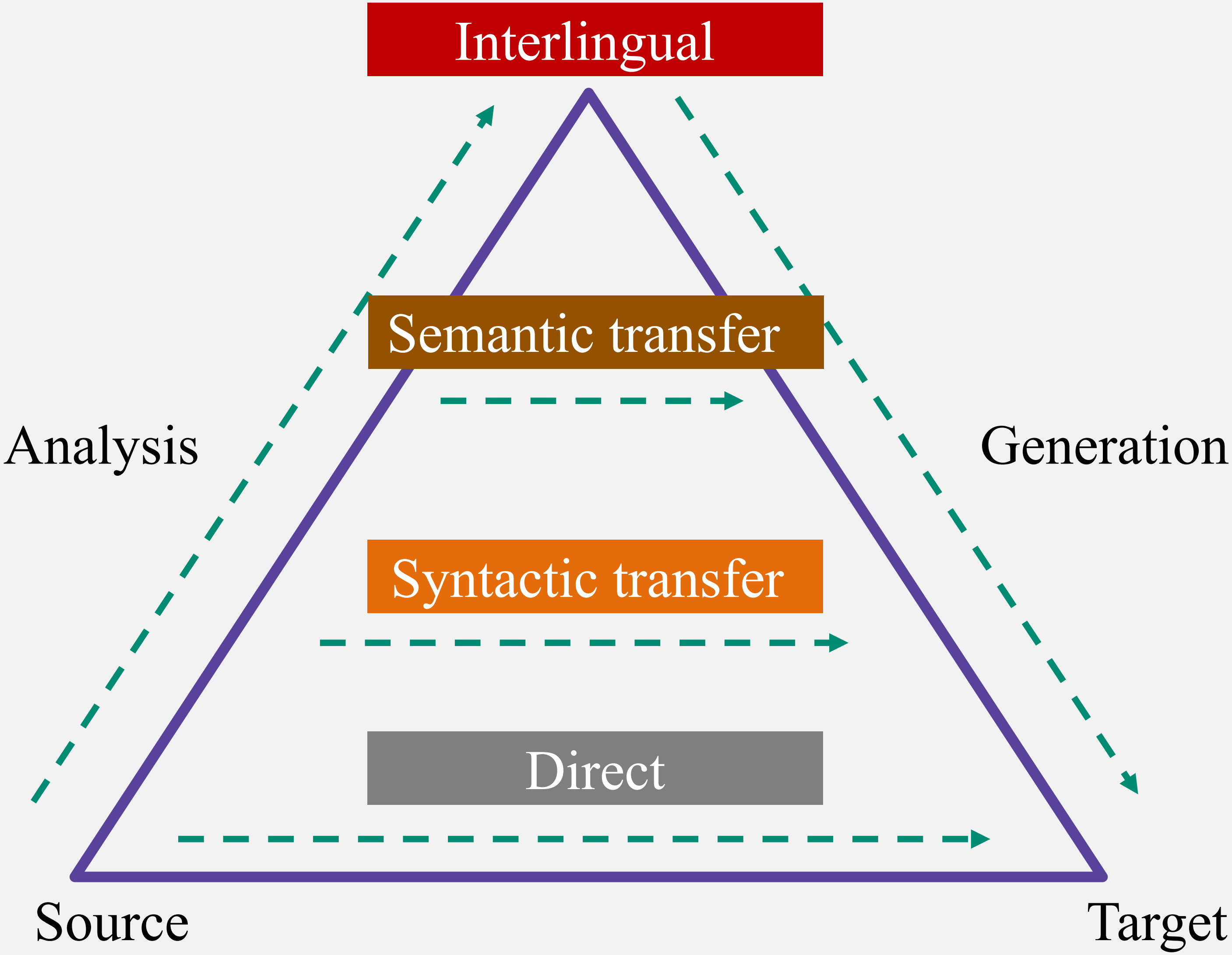
3-grams: 2 / 4

4-grams: 1 / 3

$$\text{BLEU} = 1 \cdot \sqrt[4]{\frac{4}{6} \cdot \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3}}$$

Brevity penalty : min(1, 6 / 5)

The mandatory slide



Henry S. Thompson, https://www.inf.ed.ac.uk/teaching/courses/anlp/lectures/30/

Roller-coaster of machine translation

1954 Georgetown IBM experiment Russian to English:

- Claimed that MT would be solved **within 3-5 years.**

1966 ALPAC report:

- Concluded that MT was **too expensive and ineffective.**

Two main paradigms

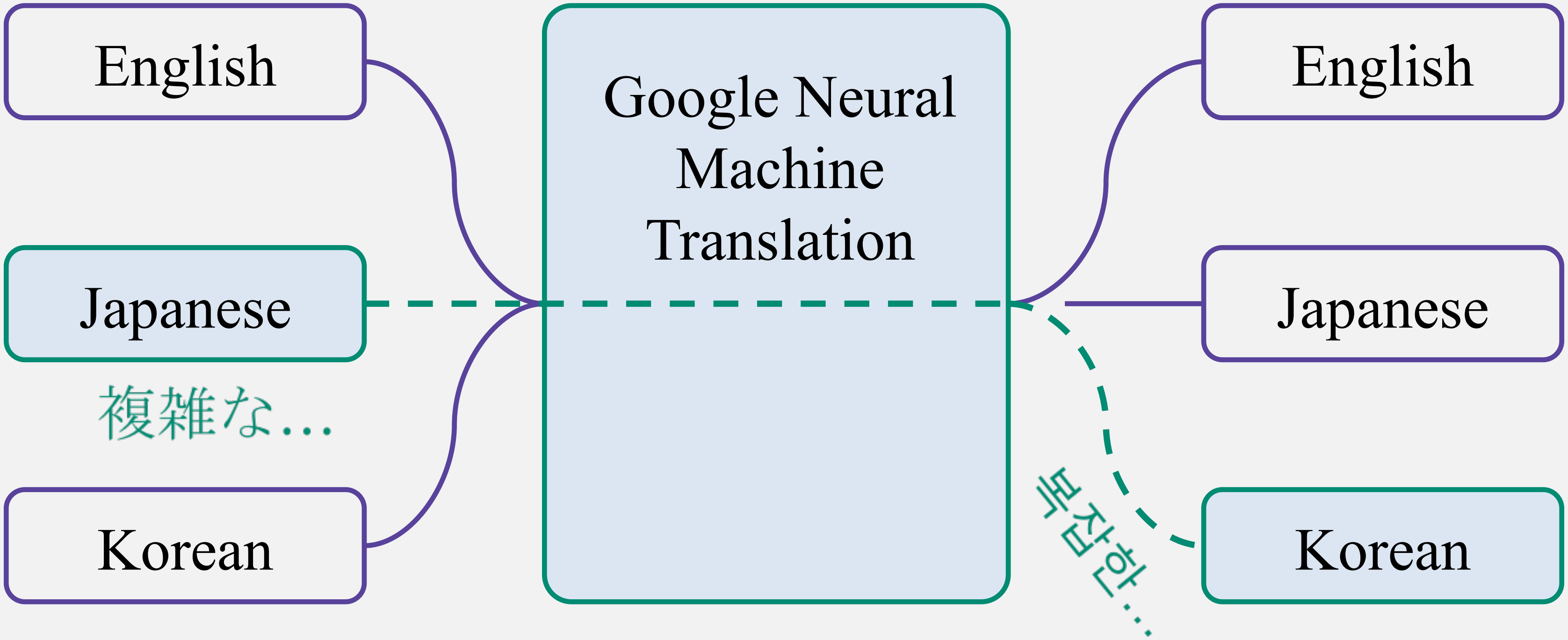
Statistical Machine Translation (SMT):

- 1988 – Word-based models (IBM models)
- 2003 – Phrase-based models (Philip Koehn)
- 2006 – Google Translate (and Moses, next year)

Neural Machine Translation (NMT):

- 2013 – First papers on pure NMT
- 2015 – NMT enters shared tasks (WMT, IWSLT)
- 2016 – Launched in production in companies

Zero-shot translation



https://research.googleblog.com/2016/11/zero-shot-translation-with-googles.html

Noisy channel:
said in English, received in French

The main equation

- **Given:** French (foreign) sentence *f*,
- **Find:** English translation *e*:

$$e^* = \operatorname*{argmax}_{e \in E} p(e|f)$$

1993 Brown et al., “The mathematics of statistical machine translation”

The main equation

- **Given:** French (foreign) sentence *f*,
- **Find:** English translation *e*:

$$e^* = \operatorname*{argmax}_{e \in E} p(e|f) = \operatorname*{argmax}_{e \in E} \frac{p(f|e)p(e)}{p(f)} =$$

1993 Brown et al., “The mathematics of statistical machine translation”

The main equation

- **Given:** French (foreign) sentence *f*,
- **Find:** English translation *e*:

$$e^* = \operatorname*{argmax}_{e \in E} p(e|f) = \operatorname*{argmax}_{e \in E} \frac{p(f|e)p(e)}{p(f)} =$$

$$= \operatorname*{argmax}_{e \in E} p(e)p(f|e)$$

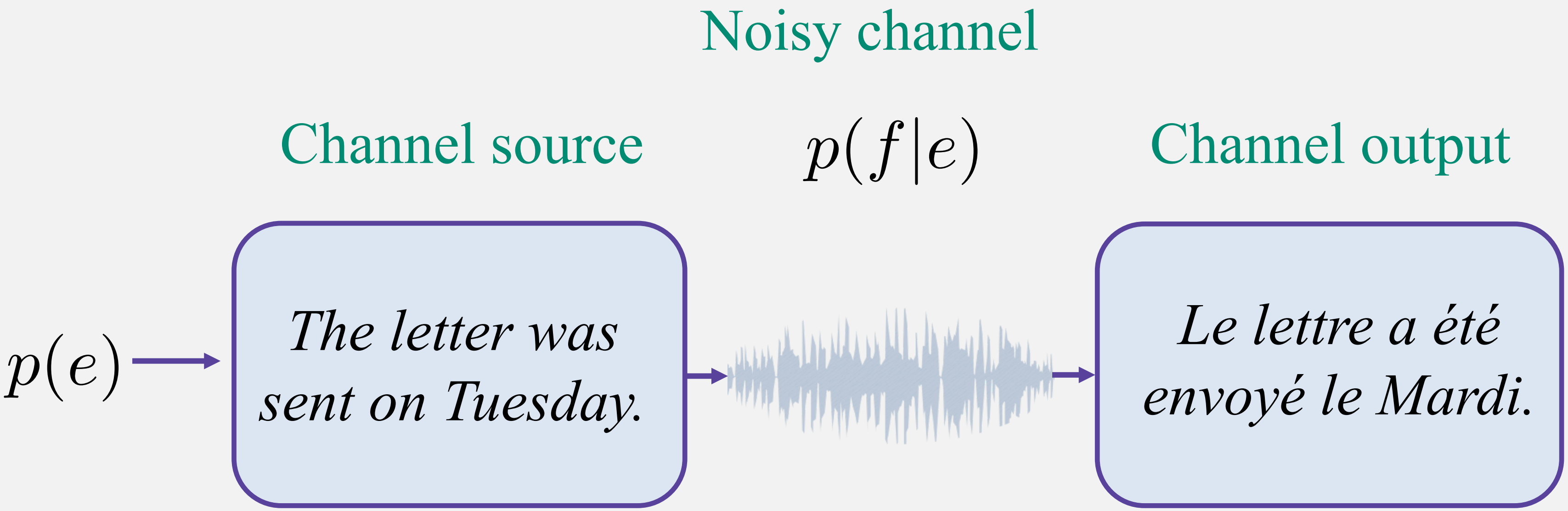
1993 Brown et al., “The mathematics of statistical machine translation”

Why is it easier to deal with?

$$e^* = \operatorname*{argmax}_{e \in E} \underbrace{p(e)}_{\text{Language model}} \underbrace{p(f|e)}_{\text{Translation model}}$$

- $p(e)$ models the *fluency* of the translation
- $p(f|e)$ models the *adequacy* of the translation
- argmax is the search problem implemented by a *decoder*

Noisy Chanel



Noisy channel

Channel source

$p(f|e)$

Channel output

$p(e)$

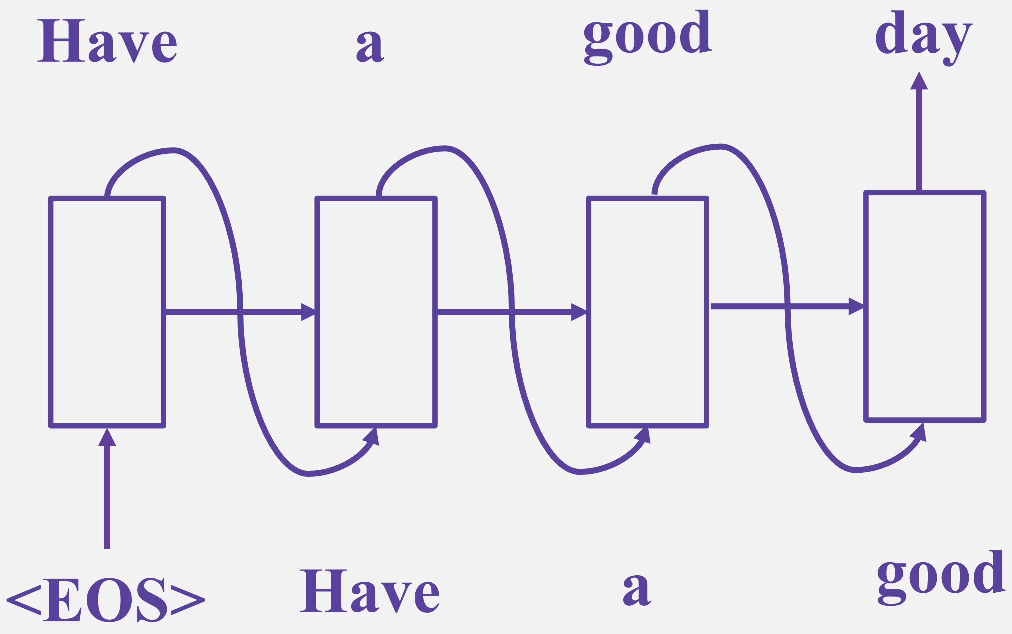
The letter was sent on Tuesday.

Le lettre a été envoyé le Mardi.

Language model: p(e)

$$p(\mathbf{e}) = p(e_1)p(e_2|e_1)\ldots p(e_k|e_1\ldots e_{k-1})$$

N-gram models or neural networks:



Translation model: p(f|e)

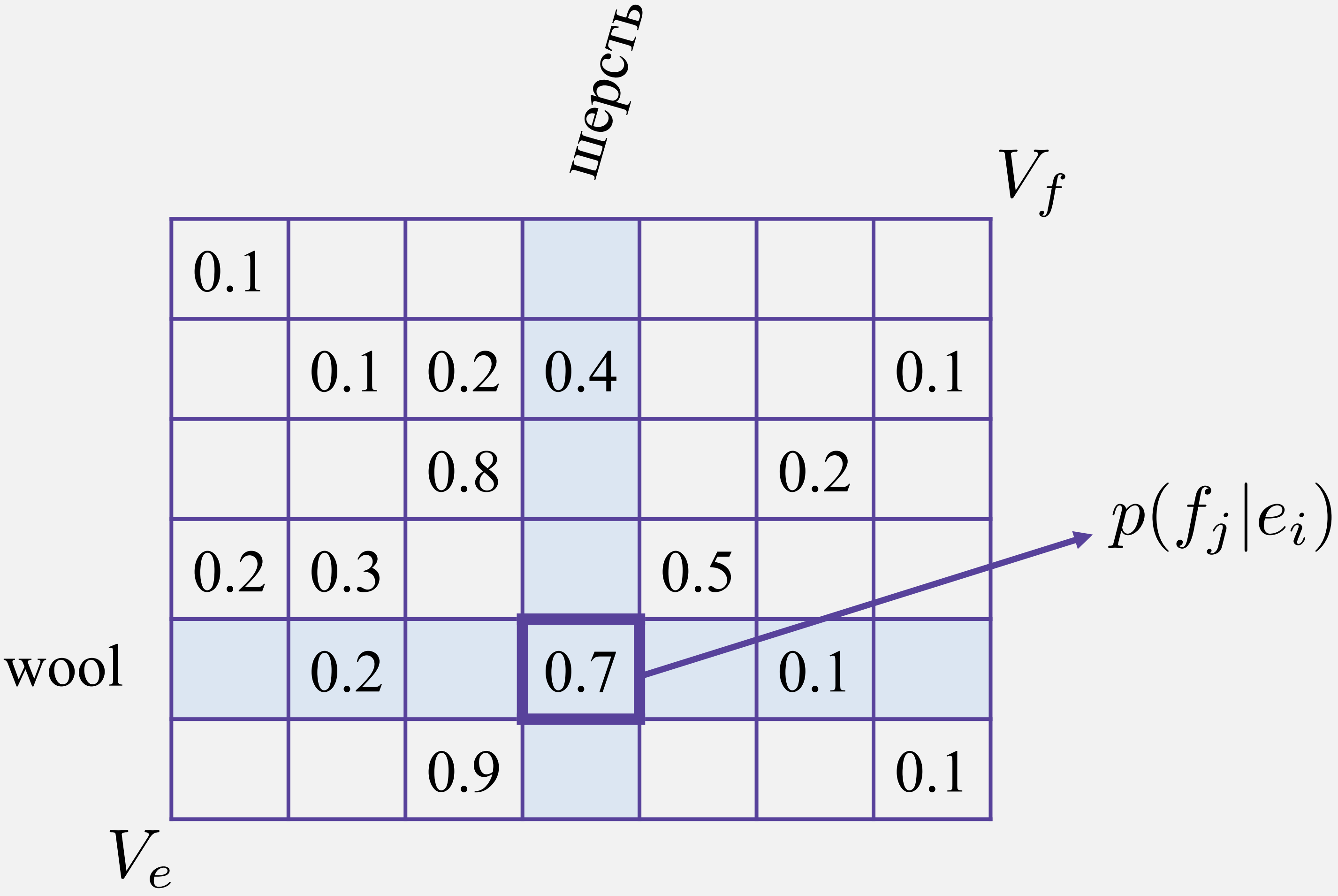
$$p(f|e) = p(f_1, f_2, \ldots f_J | e_1, e_2, \ldots e_I)$$

f (Foreign): Крику много, а шерсти мало.

e (English): Great cry and little wool.

Translation model: p(f|e)

We could learn translation probabilities for separate words:



				шерсть			
	0.1						
		0.1	0.2	0.4			0.1
			0.8			0.2	
	0.2	0.3			0.5		
wool		0.2		0.7		0.1	
			0.9				0.1

Translation model: p(f|e)

But how to build the probability for the whole sentences?

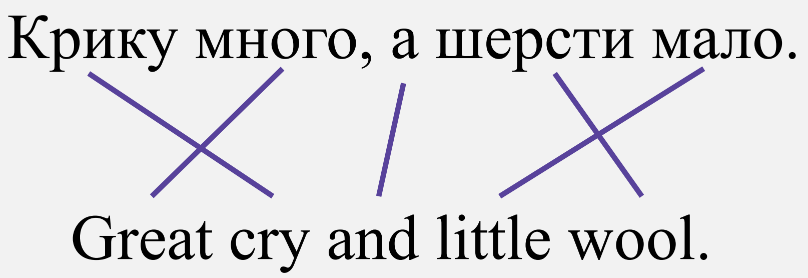
$$p(f|e) = \begin{array}{l}\text{Some Magic}\\ \text{Factorization}\end{array} \left[\; p(f_j|e_i) \;\right]$$

Translation model: p(f|e)

But how to build the probability for the whole sentences?

$$p(f|e) = \text{Some Magic Factorization} \left[p(f_j|e_i) \right]$$

Reorderings:

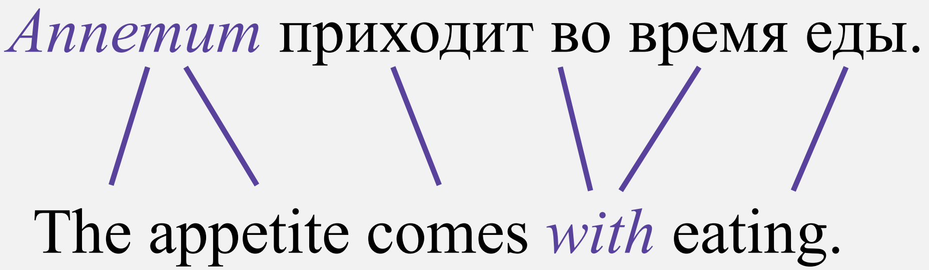


Крику много, а шерсти мало.

Great cry and little wool.

Word Alignments

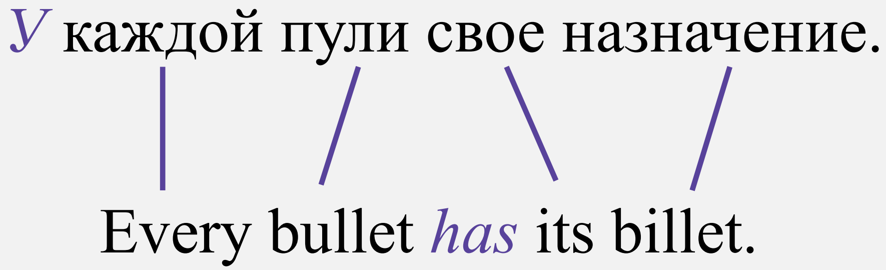
One-to-many and many-to-one:



Аппетит приходит во время еды.

The appetite comes *with* eating.

Words can disappear or appear from nowhere:



У каждой пули свое назначение.

Every bullet *has* its billet.

Word Alignment Models

Word Alignments

“As English not all languages words in the same order put. Hmmmmmm.» - Yoda

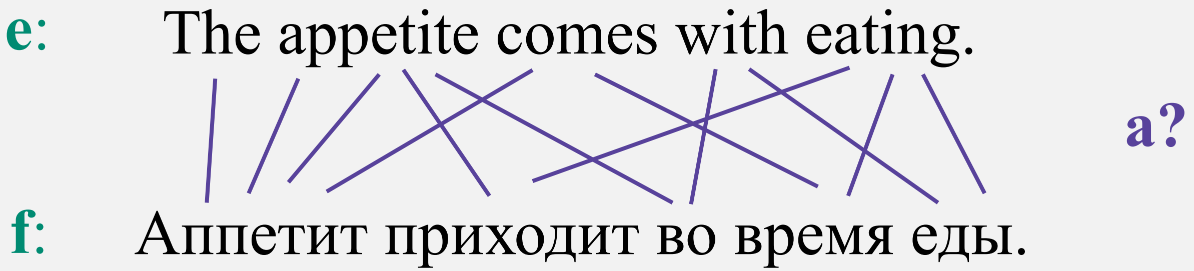
Word alignment task

Given a corpus of **(e, f)** sentence pairs:

- English, source: $e = (e_1, e_2, \ldots e_I)$
- Foreign, target: $f = (f_1, f_2, \ldots f_J)$

Predict:

- Alignments **a** between **e** and **f**:



e: The appetite comes with eating.

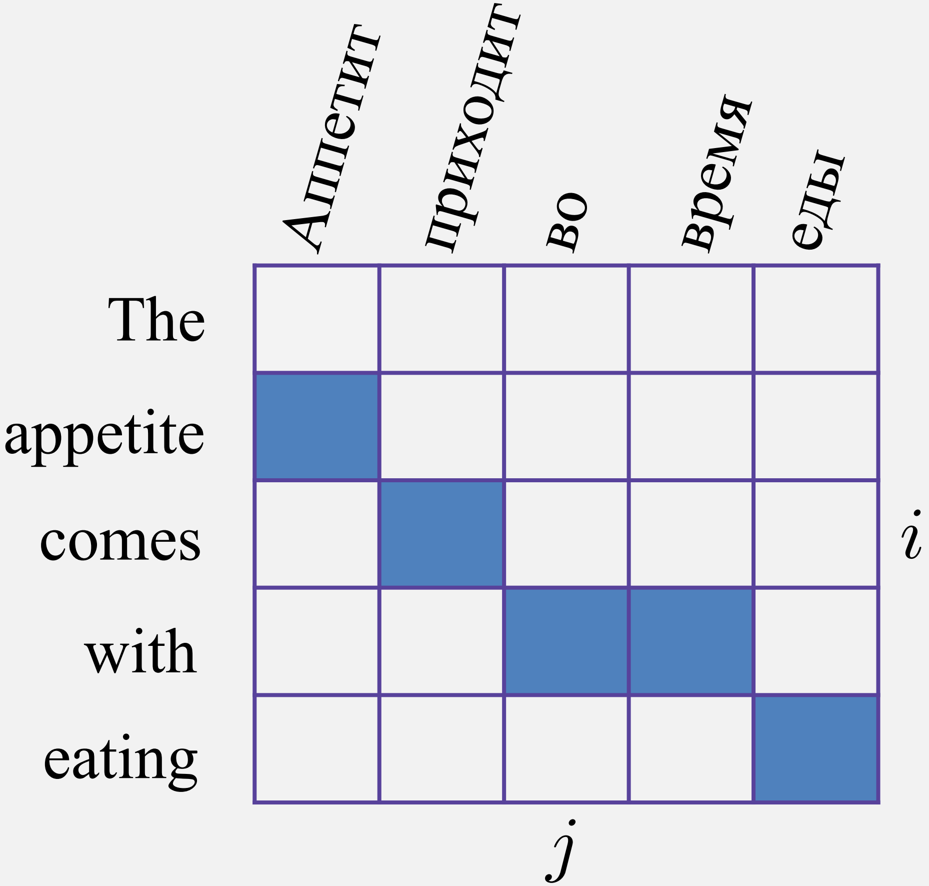
f: Аппетит приходит во время еды.

a?

Word alignment matrix

	Аппетит	приходит	во	время	еды
The	■				
appetite	■				
comes		■			
with			■	■	
eating					■

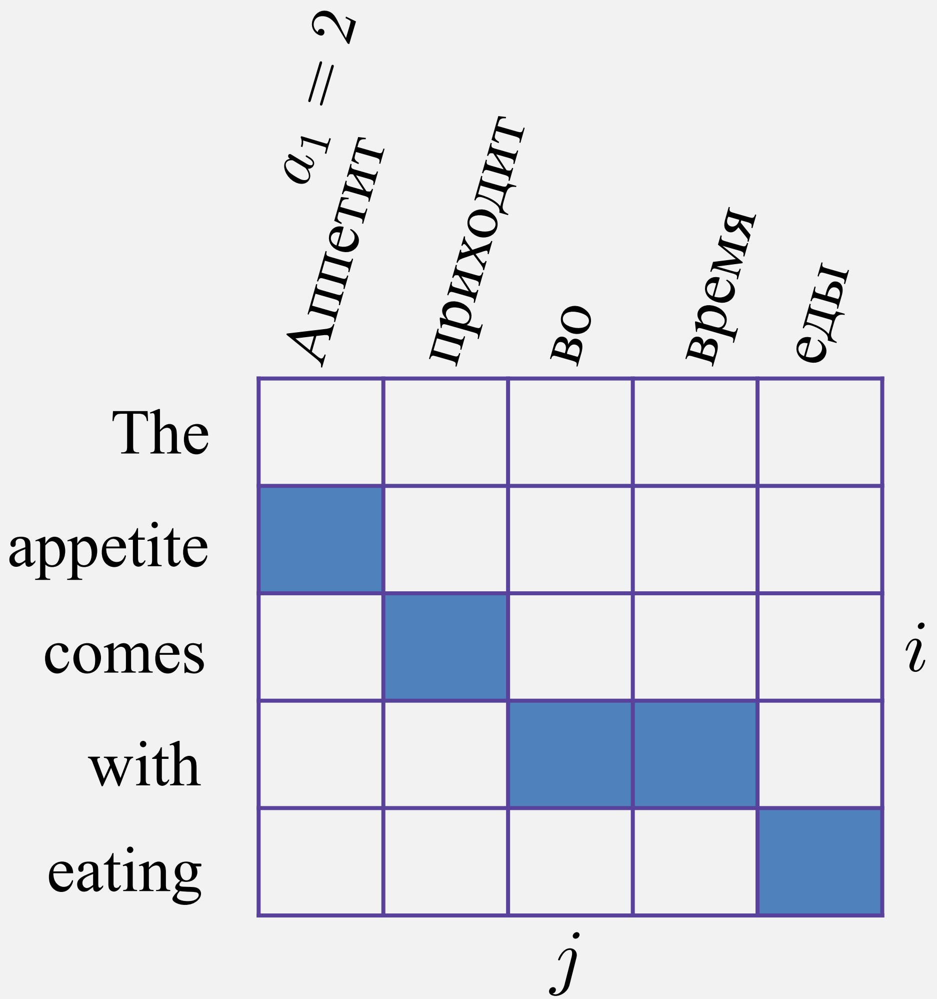
Word alignment matrix



	Аппетит	приходит	во	время	еды
The					
appetite	■				
comes		■			
with			■	■	
eating					■

Each target word is allowed to have only one source!

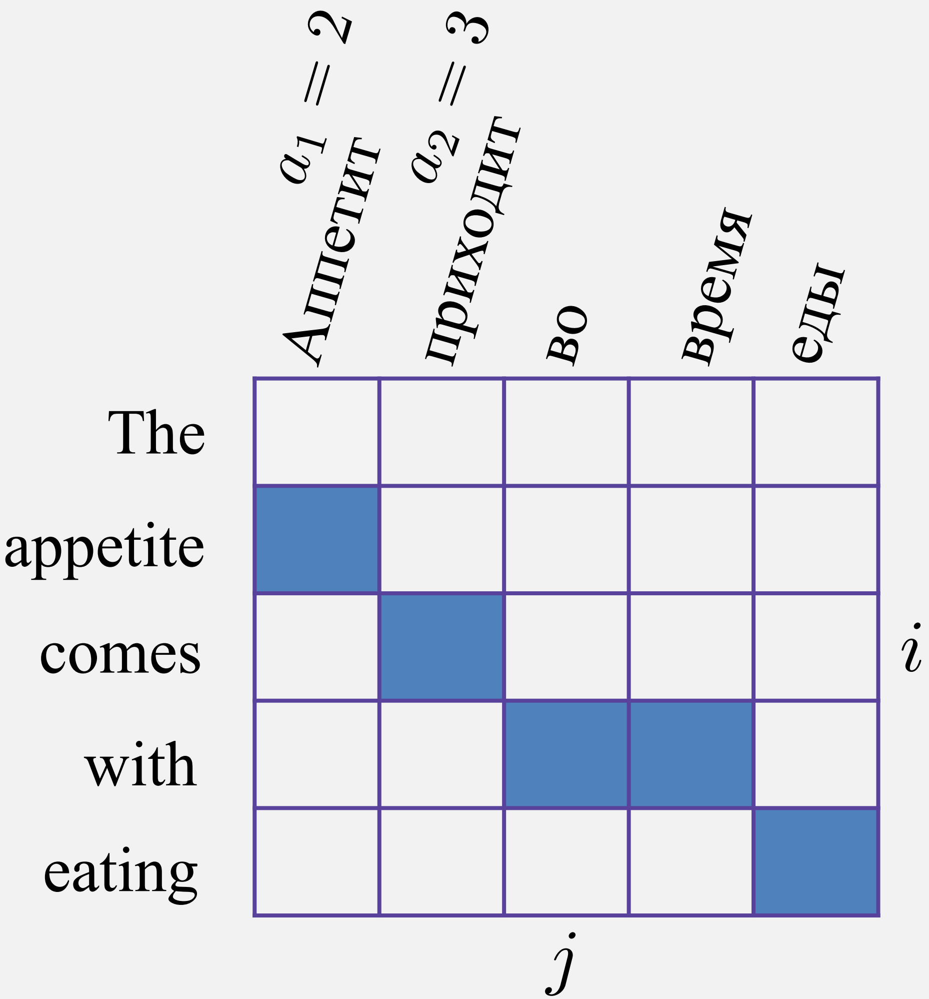
Word alignment matrix



	Аппетит	приходит	во	время	еды
The					
appetite	■				
comes		■			
with			■	■	
eating					■

Each target word is allowed to have only one source!

Word alignment matrix



	$a_1 = 2$ Аппетит	$a_2 = 3$ приходит	во	время	еды
The					
appetite	■				
comes		■			
with			■	■	
eating					■

Each target word is allowed to have only one source!

Word alignment matrix

$a_1 = 2$ $a_2 = 3$ $a_3 = 4$

	Аппетит	приходит	во	время	еды
The					
appetite	■				
comes		■			
with			■	■	
eating					■

i

j

Each target word is allowed to have only one source!

Word alignment matrix

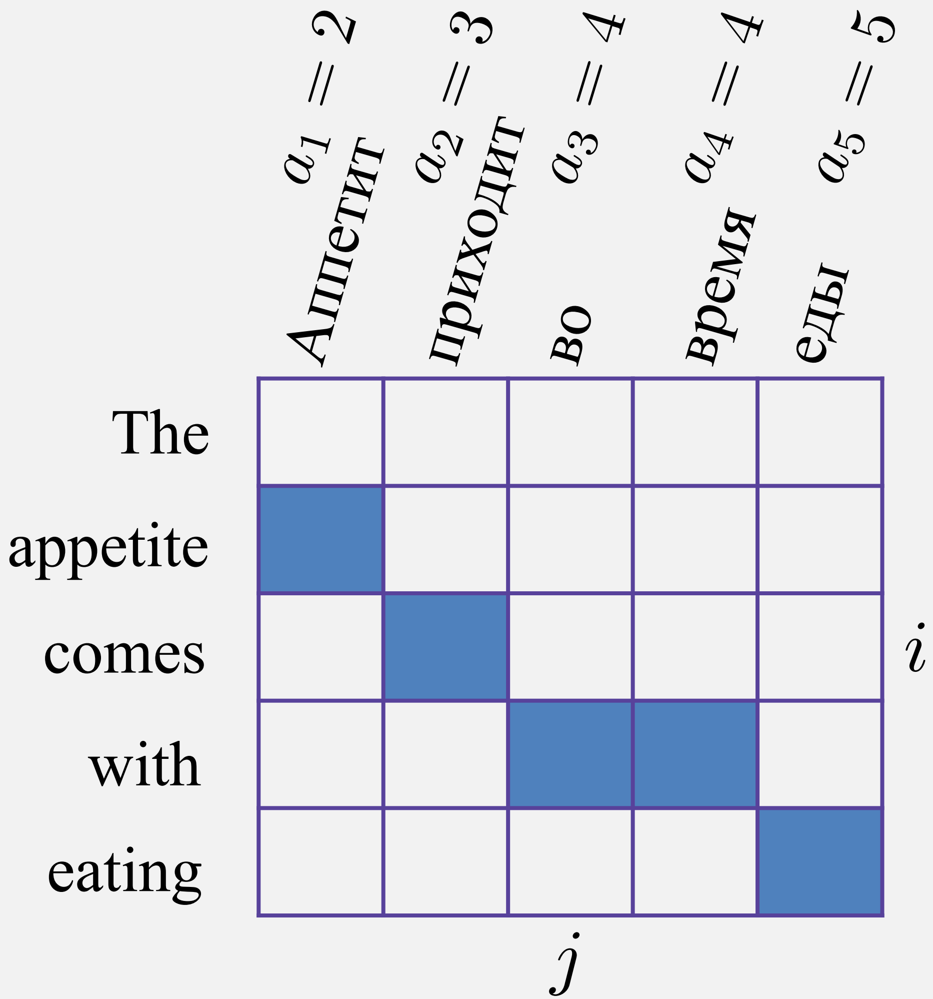
	$a_1 = 2$	$a_2 = 3$	$a_3 = 4$	$a_4 = 4$	
	Аппетит	приходит	во	время	еды
The					
appetite	■				
comes		■			
with			■	■	
eating					■

i

j

Each target word is allowed to have only one source!

Word alignment matrix



	Аппетит $a_1 = 2$	приходит $a_2 = 3$	во $a_3 = 4$	время $a_4 = 4$	еды $a_5 = 5$
The					
appetite	■				
comes		■			
with			■	■	
eating					■

Each target word is allowed to have only one source!

Sketch of learning algorithm

1. **Probabilistic model (generative story)**

   Given **e**, model the generation of **f**:

$$p(f, a | e, \Theta) = \ ?$$

The most creative step:

- How do we parametrize the model?
- Is it too complicated or too unrealistic?

Sketch of learning algorithm

1. **Probabilistic model (generative story)**

   Given **e**, model the generation of **f**:

$$p(f, a|e, \Theta) = ?$$

observable
variables

The most creative step:

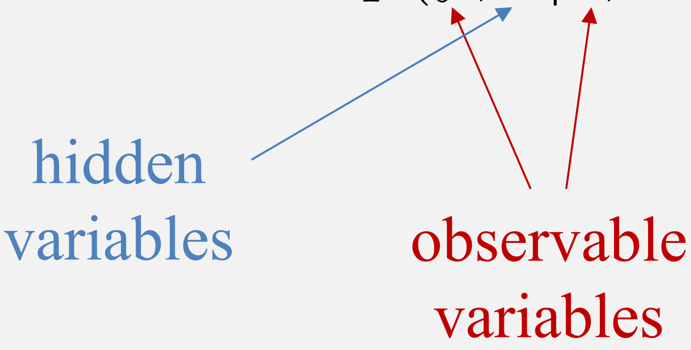
- How do we parametrize the model?
- Is it too complicated or too unrealistic?

Sketch of learning algorithm

1. Probabilistic model (generative story)

Given **e**, model the generation of **f**:

$$p(f, a|e, \Theta) = ?$$



hidden variables

observable variables

The most creative step:

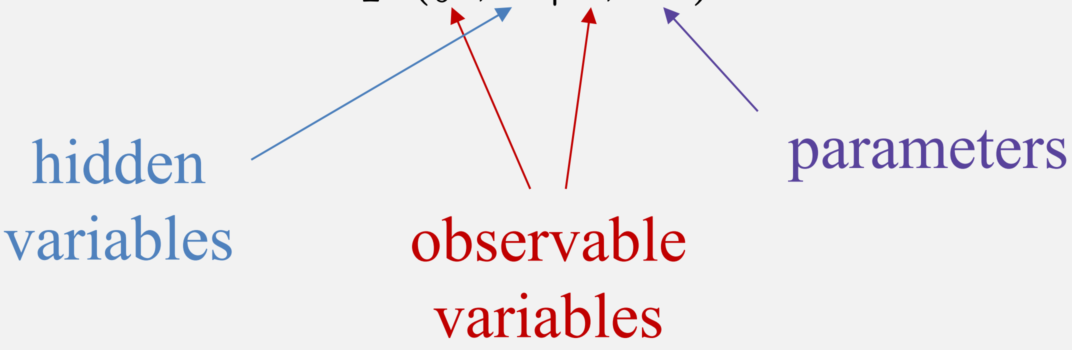
- How do we parametrize the model?
- Is it too complicated or too unrealistic?

Sketch of learning algorithm

1. **Probabilistic model (generative story)**

   Given **e**, model the generation of **f**:

$$p(f, a | e, \Theta) = ?$$



hidden variables

observable variables

parameters

The most creative step:

- How do we parametrize the model?
- Is it too complicated or too unrealistic?

Sketch of learning algorithm

2. Likelihood maximization for the incomplete data:

$$p(f|e,\Theta) = \sum_{a} p(f, a|e, \Theta) \to \max_{\Theta}$$

Sketch of learning algorithm

2. Likelihood maximization for the incomplete data:

$$p(f|e,\Theta) = \sum_{a} p(f,a|e,\Theta) \rightarrow \max_{\Theta}$$

3. EM-algorithm to the rescue!

Iterative process:

- E-step: estimates posterior probabilities for alignments
- M-step: updates Θ – parameters of the model

Generative story

$$p(f, a|e) = p(J|e)$$

1. Choose the length of the foreign sentence

Generative story

$$p(f, a|e) = p(J|e) \prod_{j=1}^{J} p(a_j | a_1^{j-1}, f_1^{j-1}, J, e) \times$$

1. Choose the length of the foreign sentence
2. Choose an alignment for each word (given lots of things)

Generative story

$$p(f,a|e) = p(J|e)\prod_{j=1}^{J} p(a_j|a_1^{j-1}, f_1^{j-1}, J, e)\times$$
$$\times p(f_j|a_j, a_1^{j-1}, f_1^{j-1}, J, e)$$

1. Choose the length of the foreign sentence
2. Choose an alignment for each word (given lots of things)
3. Choose the word (given lots of things)

IBM model 1

$$p(f, a|e) = p(J|e) \prod_{j=1}^{J} p(a_j) p(f_j|a_j, e)$$

Uniform prior

ε

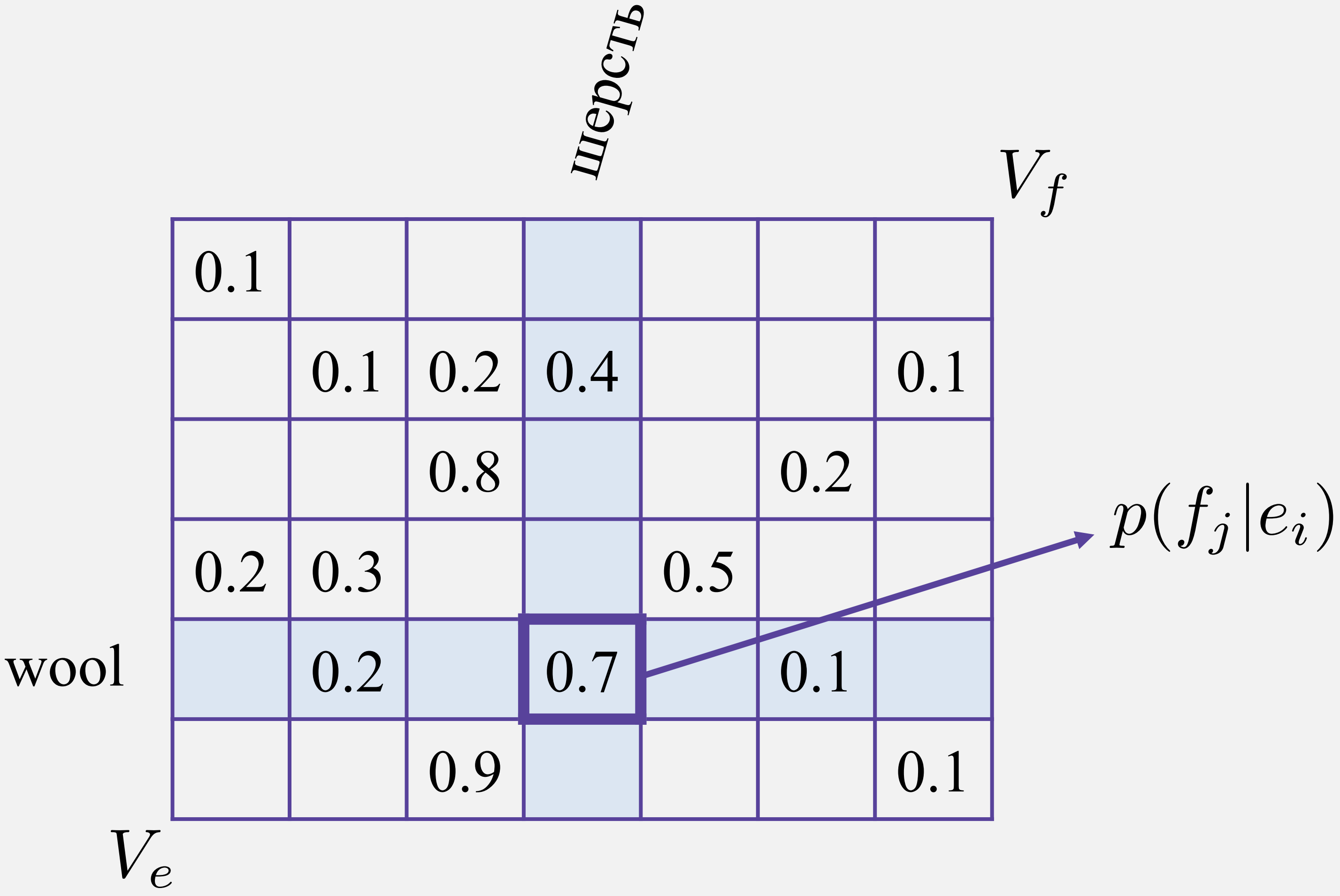
Translation table

$t(f_j|e_{a_j})$

+ The model is simple and has not too many parameters

– The alignment prior does not depend on word positions

Translation table



				шерсть			
	0.1						
		0.1	0.2	0.4			0.1
			0.8			0.2	
	0.2	0.3			0.5		
wool		0.2		0.7		0.1	
			0.9				0.1

IBM model 2

$$p(f, a|e) = p(J|e) \prod_{j=1}^{J} p(a_j|j, I, J) p(f_j|a_j, e)$$

Position-based prior
$d(a_j|j, I, J)$

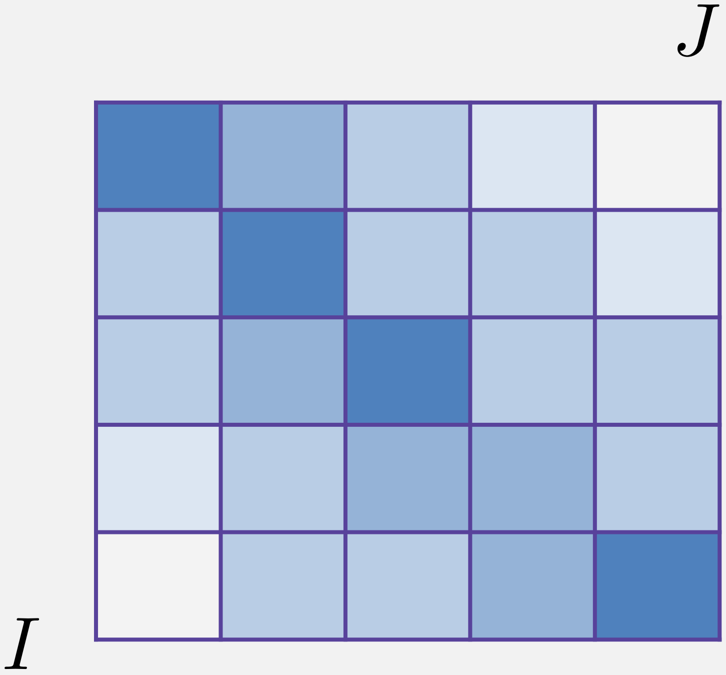
Translation table
$t(f_j|e_{a_j})$

+ The alignments depend on position-based prior

– Quite a lot of parameters for the alignments

Position-based prior

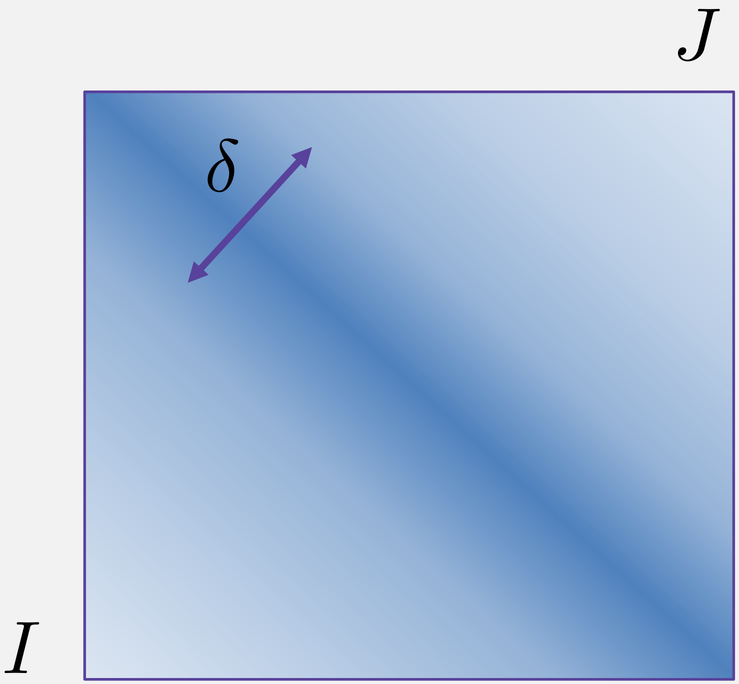
- For each pair of the **lengths** of the sentences:
    - $I \times J$ matrix of probabilities



Dyer et al. A Simple, Fast, and Effective Reparameterization of IBM Model 2, 2013

Re-parametrization, Dyer et. al 2013

- If we know, it's going to be diagonal – let's model it diagonal!
- Much less parameters, easier to train on small data



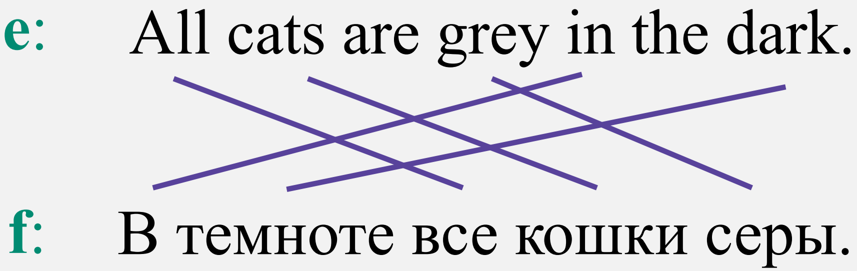
Dyer et al. A Simple, Fast, and Effective Reparameterization of IBM Model 2, 2013

HMM for the prior

$$p(f, a|e) = \prod_{j=1}^{J} p(a_j|a_{j-1}, I, J)p(f_j|a_j, e)$$

Transition probabilities
$d(a_j|a_{j-1}, I, J)$

Translation table
$t(f_j|e_{a_j})$



e: All cats are grey in the dark.

f: В темноте все кошки серы.

Resume

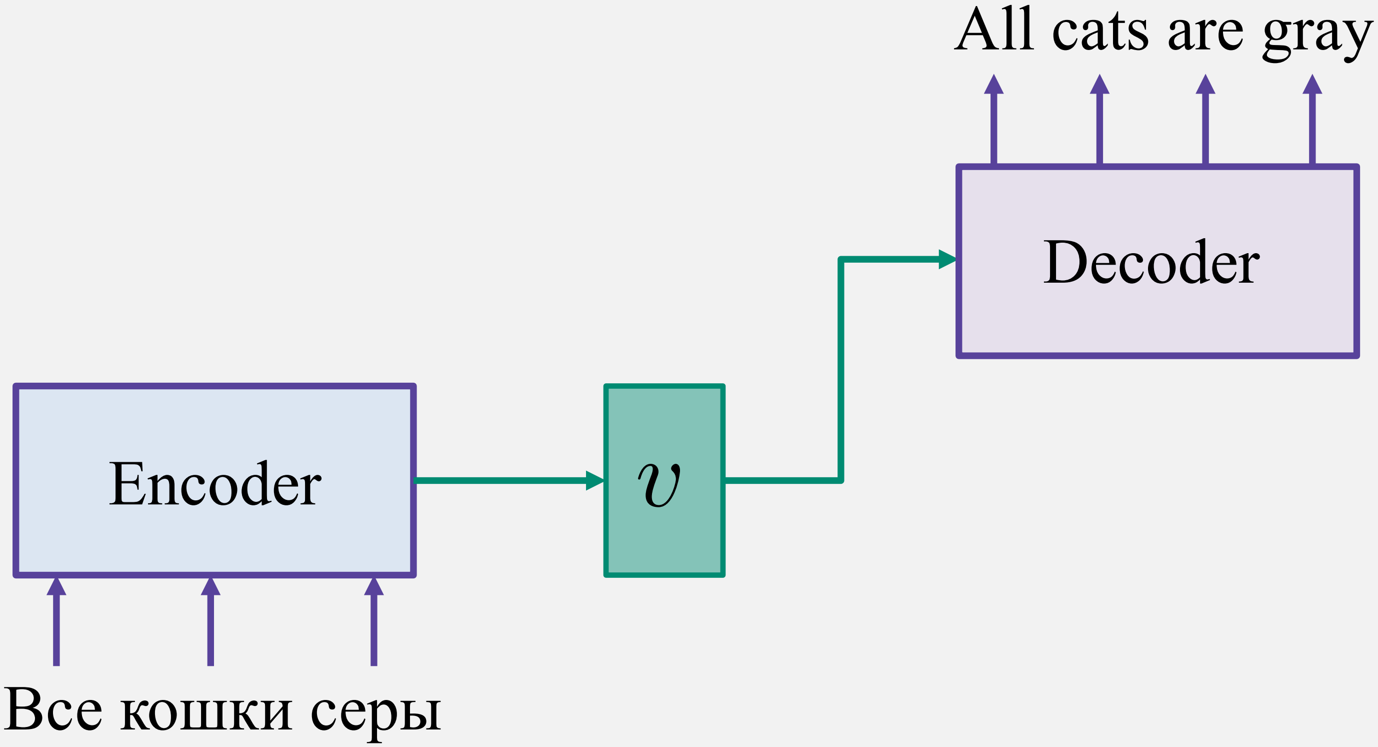
- IBM models – first working systems of MT

- Lot’s of problems with models 1 and 2:
  - How to deal with *spurious words*
  - How to control *fertility*
  - ….

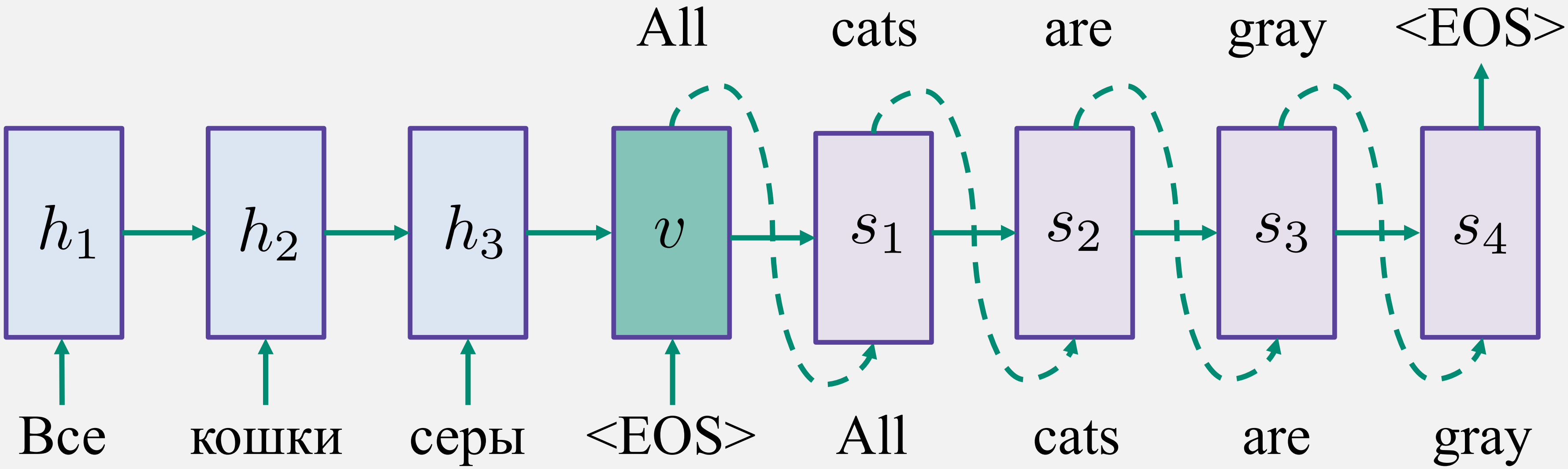
- Most importantly, how to do many-to-many alignments?
  - Phrased-based machine translation (Koehn’s book)

Encoder-decoder architecture

Sequence to sequence

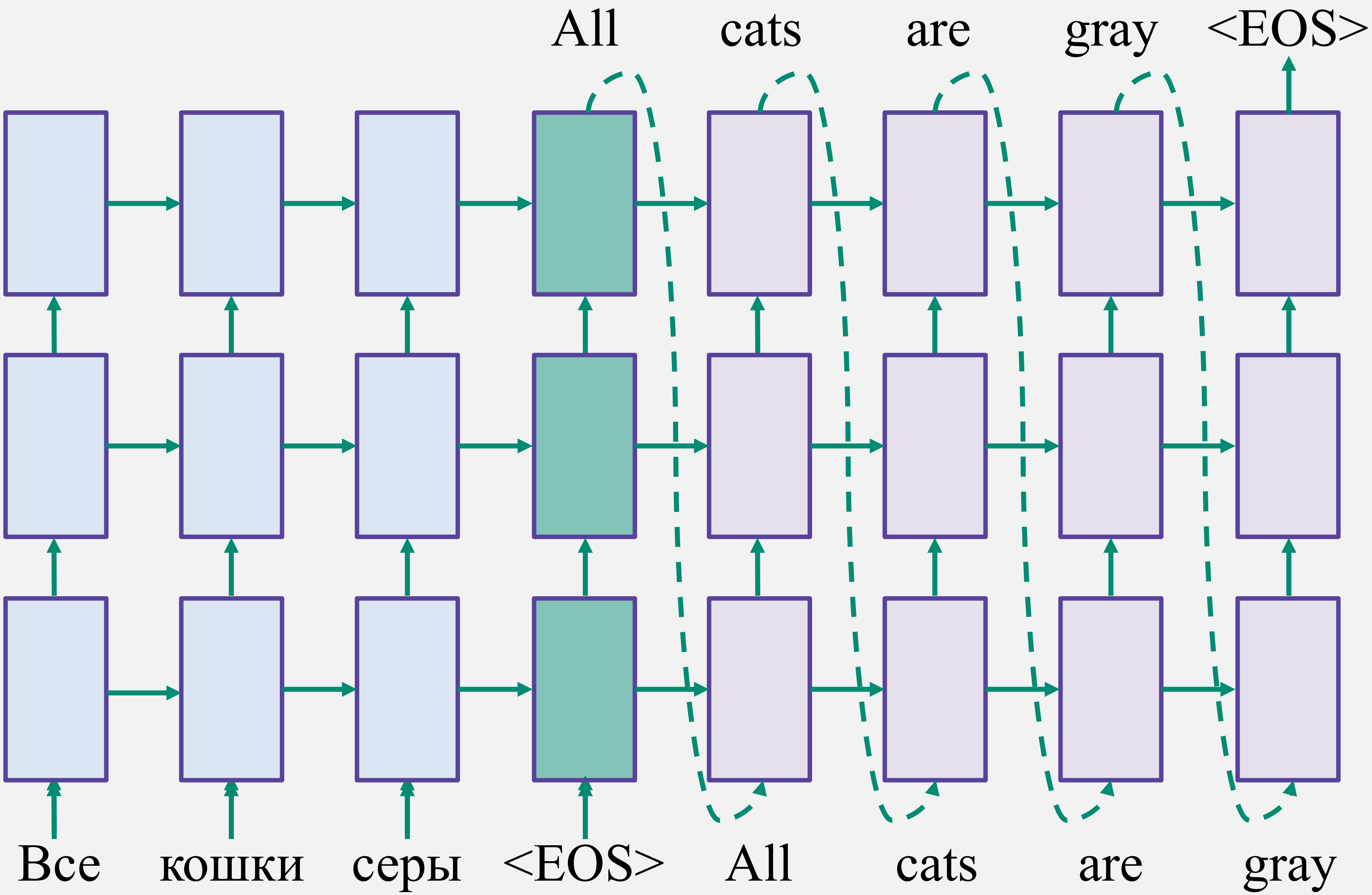


Sequence to sequence

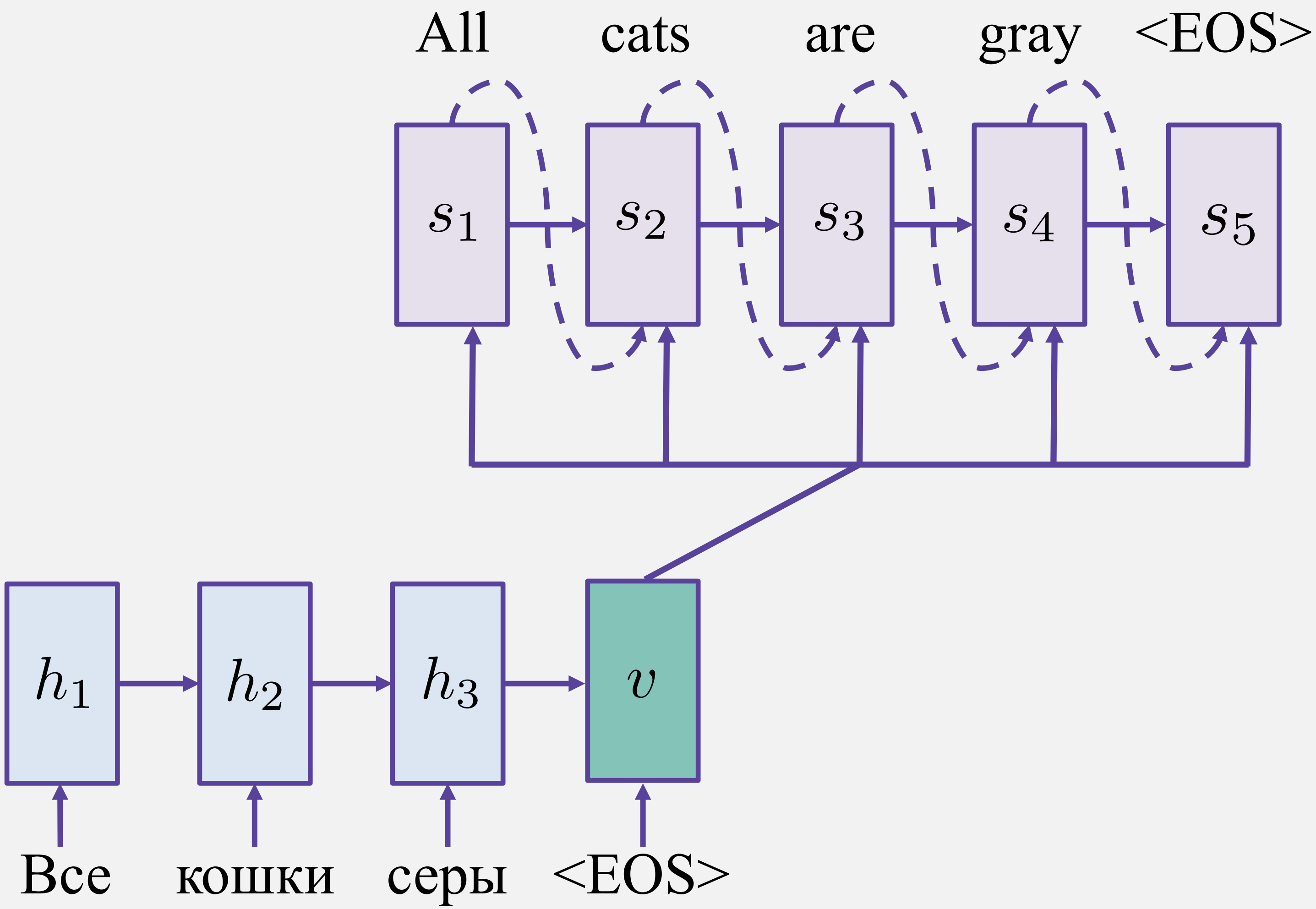


Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with Neural Network, 2014.

Sequence to sequence



Sequence to sequence



Cho et. al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014.

Sequence to sequence

$$p(y_1, \ldots y_J | x_1, \ldots x_I) = \prod_{j=1}^{J} p(y_j | v, y_1, \ldots y_{j-1})$$

- **Encoder:** maps the source sequence to the hidden vector

RNN: $h_i = f(h_{i-1}, x_i)$ $v = h_I$

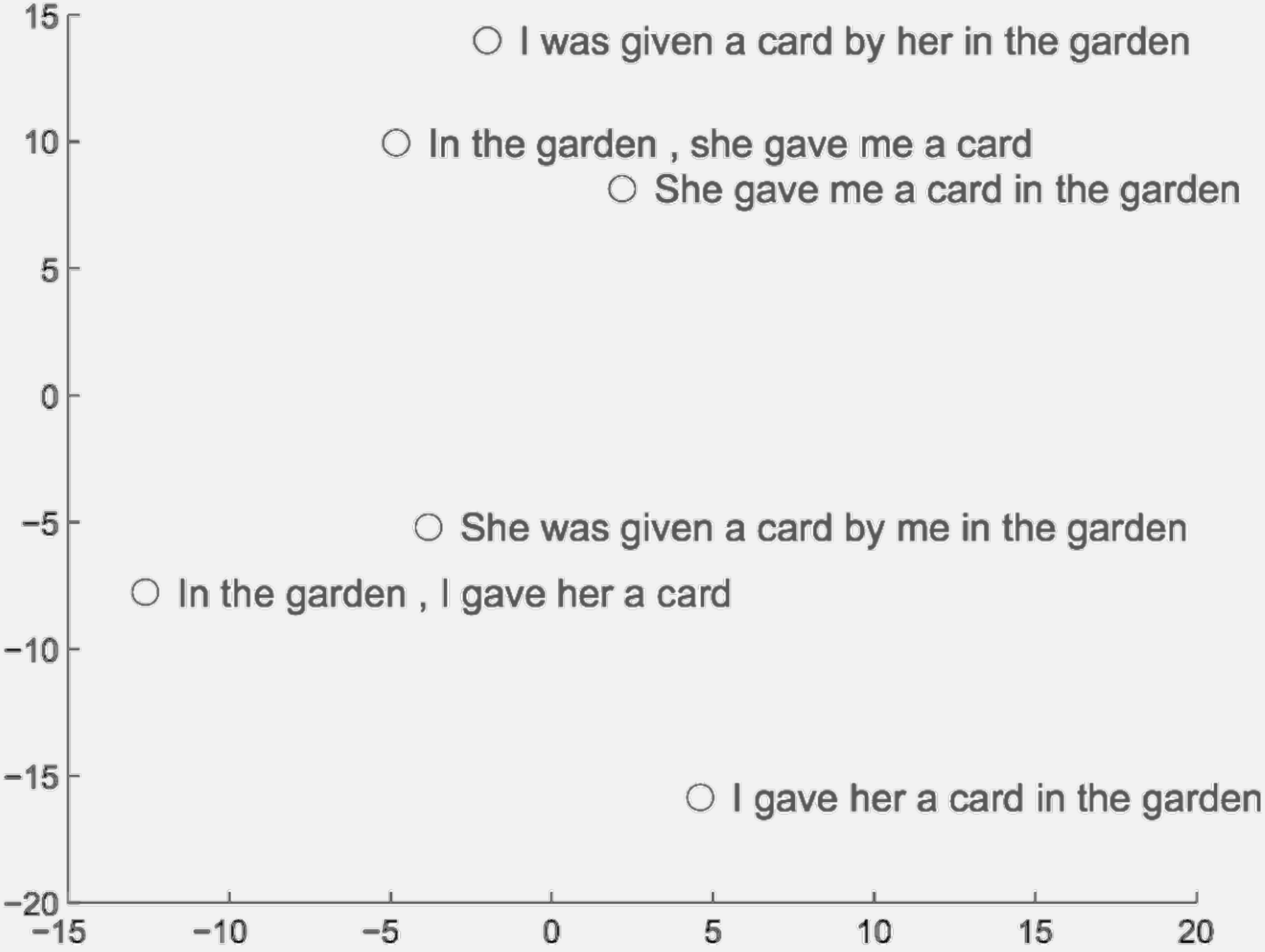
- **Decoder:** performs language modeling given this vector

RNN: $s_j = g(s_{j-1}, [y_{j-1}, v])$

- **Prediction** (the simplest way):

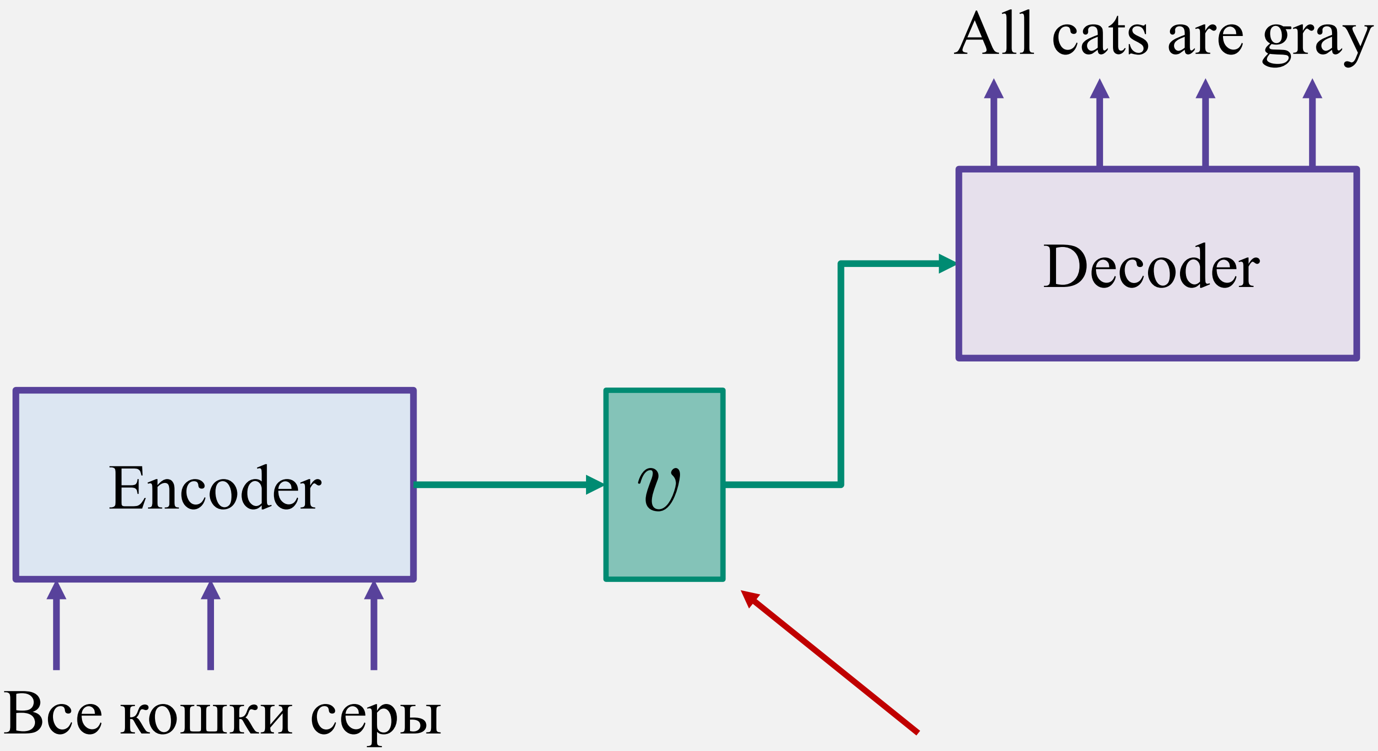
$$p(y_j | v, y_1, \ldots y_{j-1}) = softmax\,(Us_j + b)$$

Hidden representations are good…



Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with Neural Network, 2014.

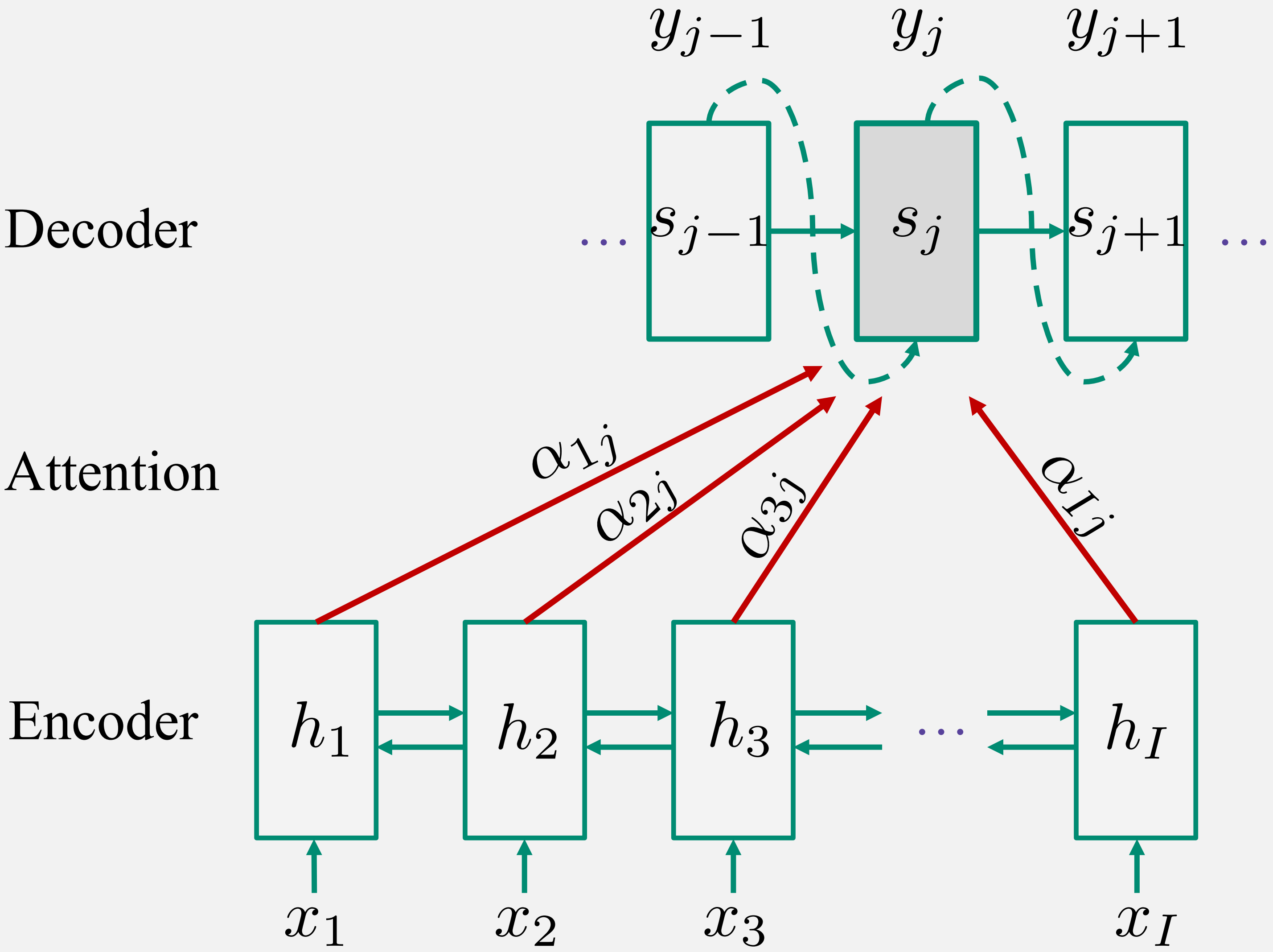
… but still a bottleneck



Bottleneck!

Attention mechanism

Attention mechanism



Bahdanau et. al - Neural Machine Translation by jointly learning to align and translate, 2015.

Attention mechanism

- Encoder states are weighted to obtain the representation relevant to the decoder state:

$$v_j = \sum_{i=1}^{I} \alpha_{ij} h_i$$

- The weights are learnt and should find the most relevant encoder positions:

$$\alpha_{ij} = \frac{\exp(sim(h_i, s_{j-1}))}{\sum_{i'=1}^{I} \exp(sim(h_{i'}, s_{j-1}))}$$

How to compute attention weights?

- **Additive attention:**

$$sim(h_i, s_j) = w^T \tanh(W_h h_i + W_s s_j)$$

- **Multiplicative attention:**

$$sim(h_i, s_j) = h_i^T W s_j$$

- **Dot product also works:**

$$sim(h_i, s_j) = h_i^T s_j$$

Put all together

$$p(y_1, \ldots y_J | x_1, \ldots x_I) = \prod_{j=1}^{J} p(y_j | v_j, y_1, \ldots y_{j-1})$$

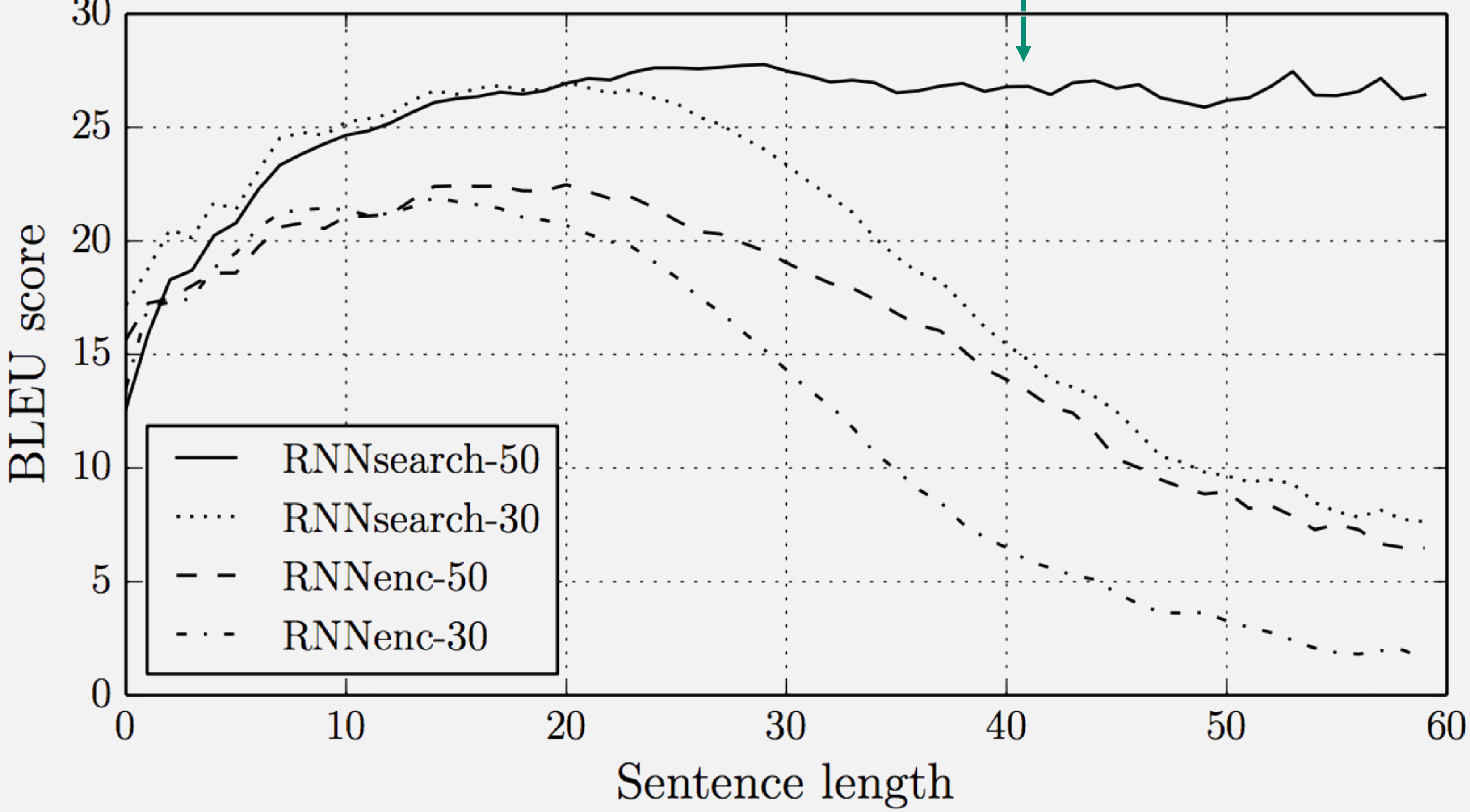
- Still encoder-decoder architecture with RNNs:

$$h_i = f(h_{i-1}, x_i) \qquad s_j = g(s_{j-1}, [y_{j-1}, v_j])$$

- But the source representations differ for each position j of the decoder.

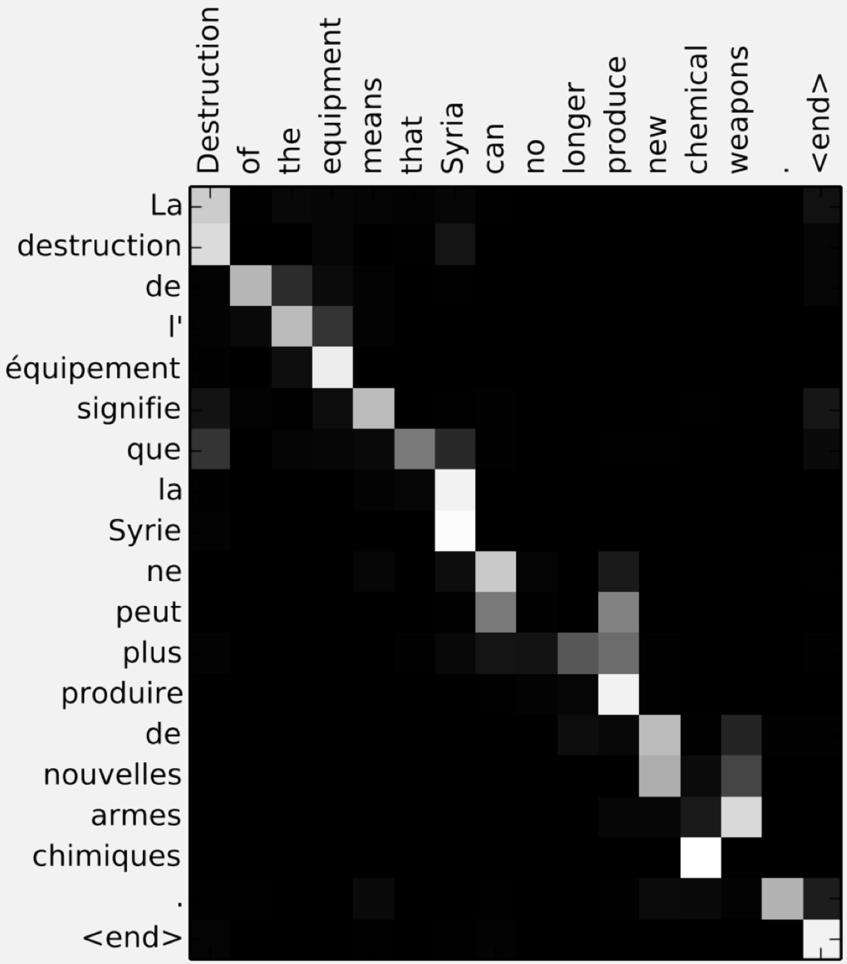
Helps for long sentences

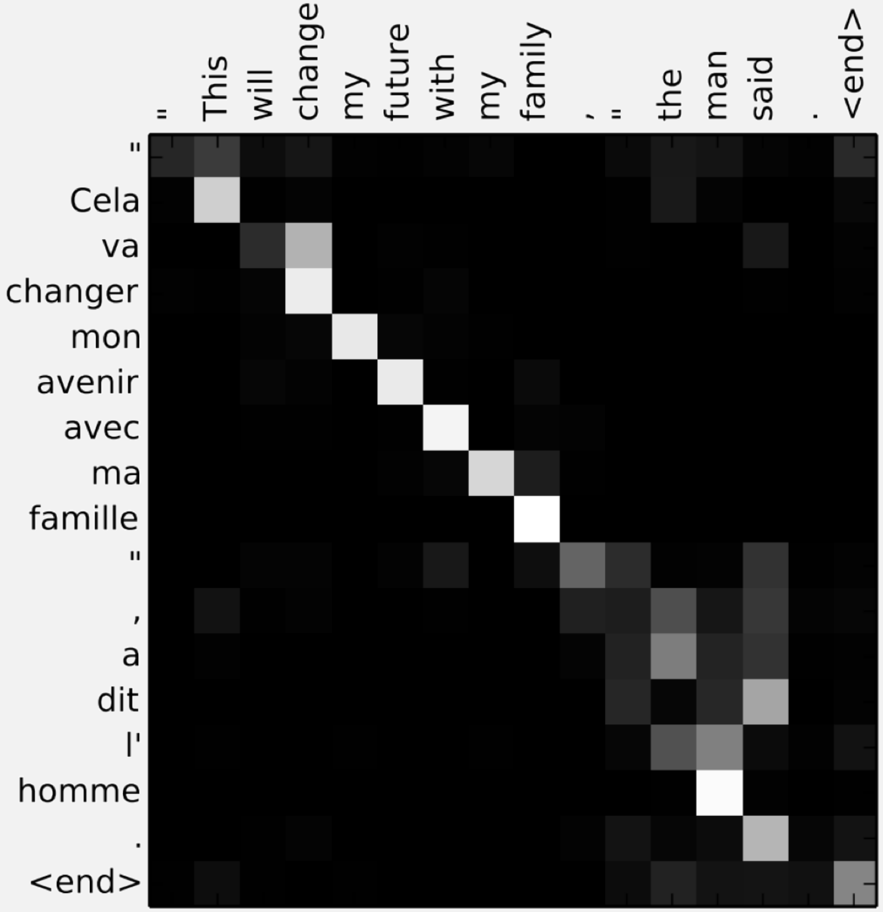
NMT with attention



Bahdanau et. al. Neural Machine Translation by jointly learning to align and translate, 2015.

Example: attention (alignments)





Bahdanau et. al. Neural Machine Translation by jointly learning to align and translate, 2015.

Is the attention similar to what humans do?

- *For humans:* **saves time**

Attention saves time when reading (i.e. we look only to the relevant parts of the sentence).

- *For machines:* **wastes time**

To compute the attention weights, the model carefully examines ALL the positions, thus wastes even more time.

Local attention

1. **Find the most relevant position $\mathbf{a_j}$ in the source**

- Monotonic alignments: $a_j = j$
- Predictive alignments: $a_j = I \cdot \sigma(b^T \tanh(W s_j))$

2. **Attend only positions within a window $\mathbf{[a_j - h; a_j + h]}$**

- Compute scores as usual
- Probably multiply by a Gaussian centered in a_j

Luong et. al. Effective Approaches to Attention-based Neural Machine Translation, 2015.

Global vs local attention

System	Perplexity	BLEU
global (location)	6.4	19.3
global (dot)	6.1	20.5
global (mult)	6.1	19.5
local-m (dot)	>7.0	x
local-m (mult)	6.2	20.4
local-p (dot)	6.6	19.6
local-p (mult)	**5.9**	**20.9**

Luong et. al. Effective Approaches to Attention-based Neural Machine Translation, 2015.

Global vs local attention

	System	Perplexity	BLEU
$W s_j$ →	global (location)	6.4	19.3
$h_i^T s_j$ →	global (dot)	6.1	20.5
$h_i^T W s_j$ →	global (mult)	6.1	19.5
	local-m (dot)	>7.0	x
	local-m (mult)	6.2	20.4
	local-p (dot)	6.6	19.6
	local-p (mult)	**5.9**	**20.9**

Luong et. al. Effective Approaches to Attention-based Neural Machine Translation, 2015.

How to deal with a vocabulary?

Outline

- Computing *softmax* for a large vocabulary is slow!
  - Hierarchical softmax
- Even a large vocabulary has *OOV words*:
  - Copy mechanism
  - Sub-word modeling
    - Word-character hybrid models
    - Byte-pair encoding

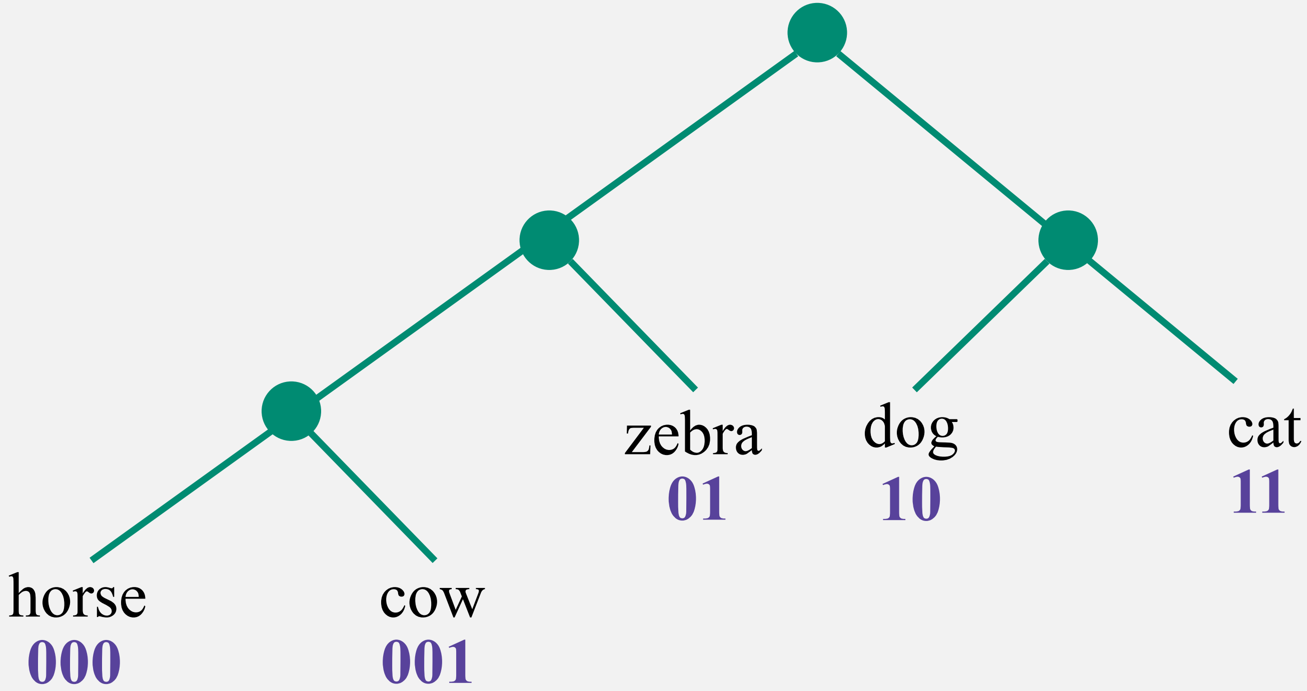
Outline

- Computing *softmax* for a large vocabulary is slow!
  - **Hierarchical softmax**
- Even a large vocabulary has *OOV words*:
  - Copy mechanism
  - Sub-word modeling
    - Word-character hybrid models
    - Byte-pair encoding

Hierarchical softmax

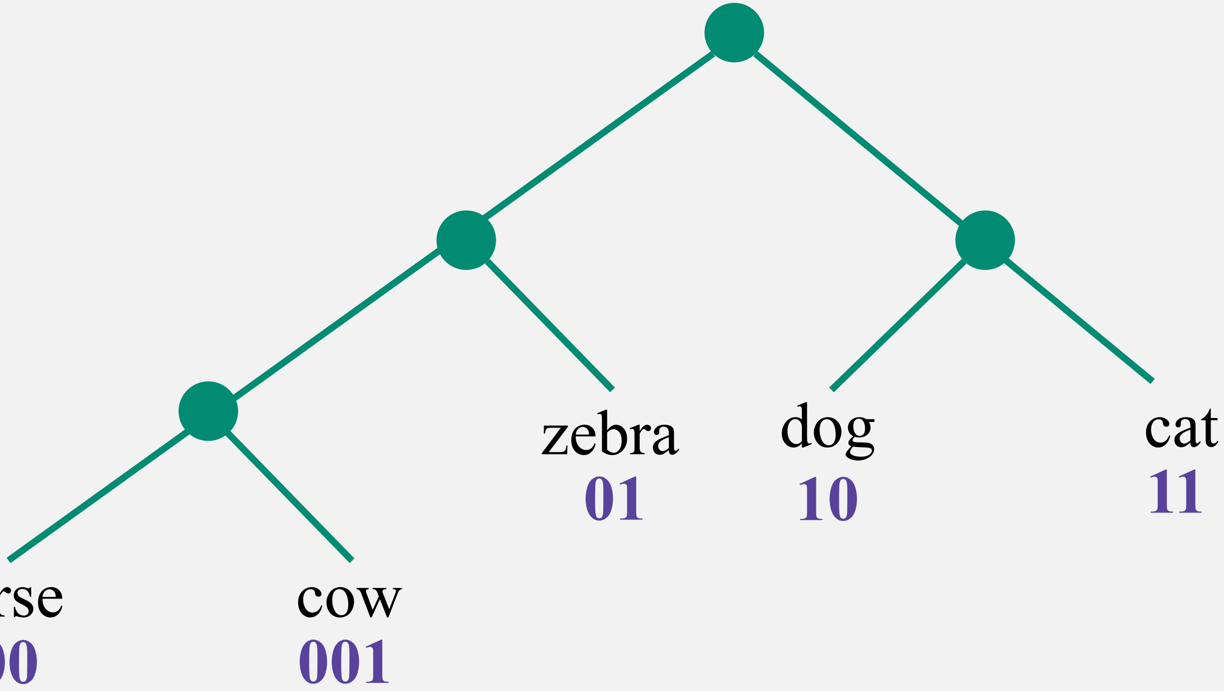
Each word is uniquely represented by a binary code:

- 0 means “go left”, 1 means “go right”



Hierarchical softmax

E.g. for **zebra** the code is $d = (\underset{d_1}{0}, \underset{d_2}{1})$



Scaling softmax

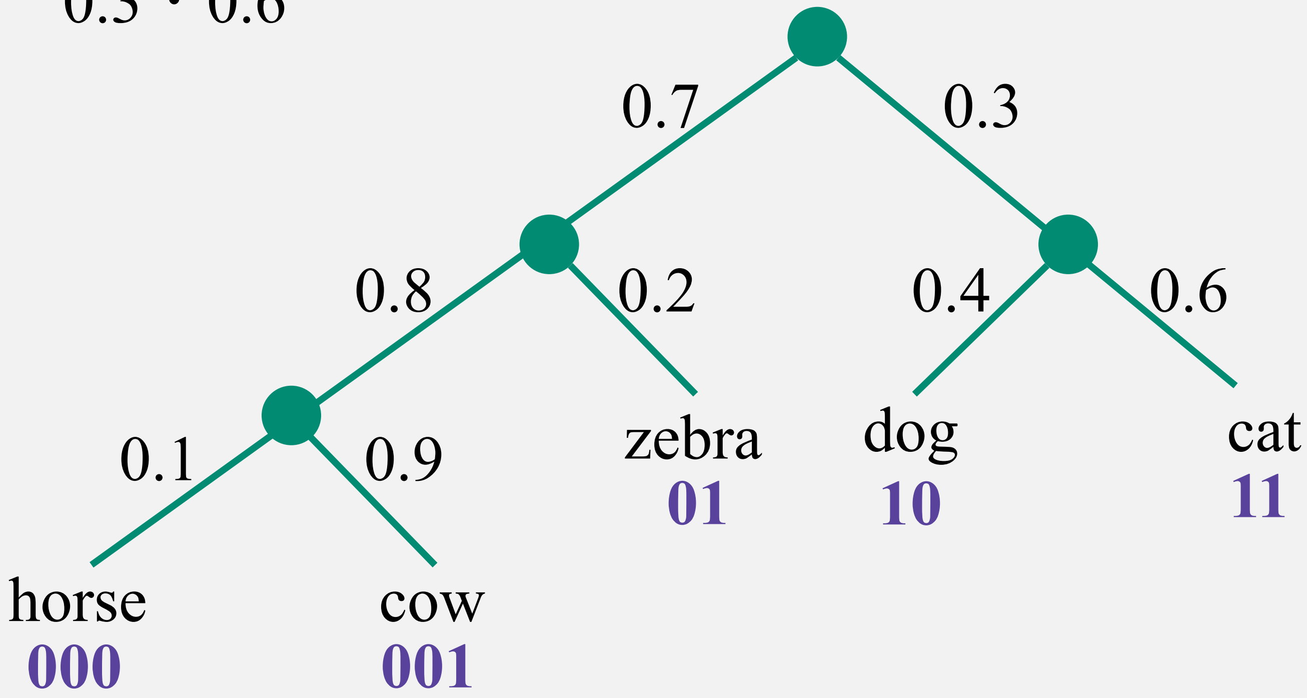
Express the probability of a word (zebra) as a product of probabilities of the binary decisions along the path (d_1, d_2).

$$p(w_n = w | w_1^{n-1}) = \prod_i p(d_i | w_1^{n-1})$$

Do you believe that it sums to 1?

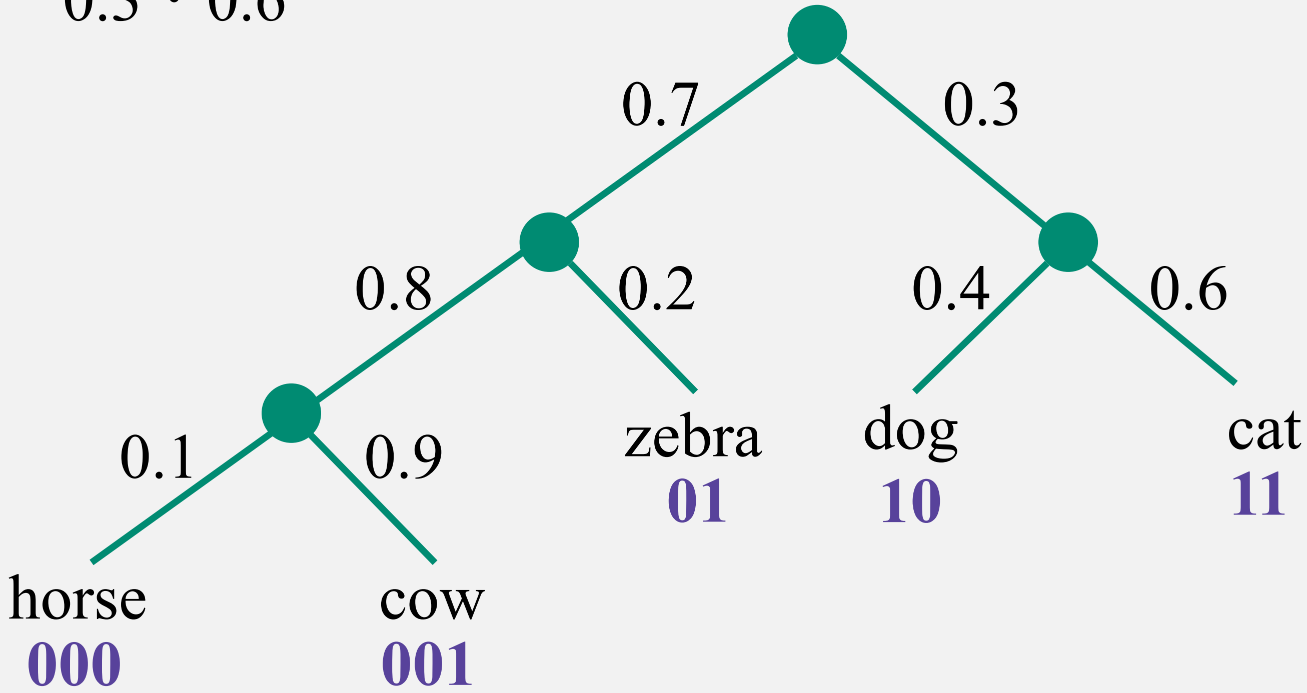
Hierarchical softmax

$$\begin{array}{ll} & 0.7 \cdot 0.8 \cdot 0.1 \\ & 0.7 \cdot 0.8 \cdot 0.9 \\ + & 0.7 \cdot 0.2 \\ & 0.3 \cdot 0.4 \\ & 0.3 \cdot 0.6 \end{array}$$



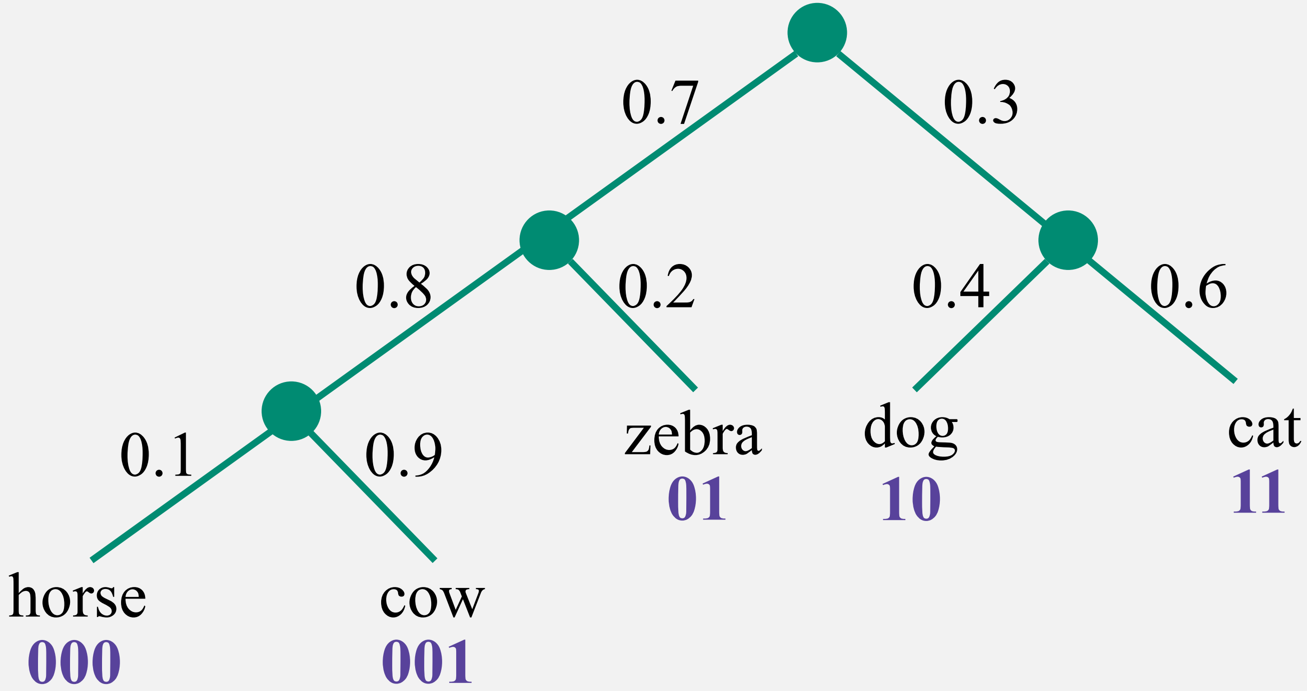
Hierarchical softmax

$$\begin{array}{ll} & 0.7 \cdot 0.8 \cdot 0.1 \\ & 0.7 \cdot 0.8 \cdot 0.9 \\ + & 0.7 \cdot 0.2 \\ & 0.3 \cdot 0.4 \\ & 0.3 \cdot 0.6 \end{array}$$



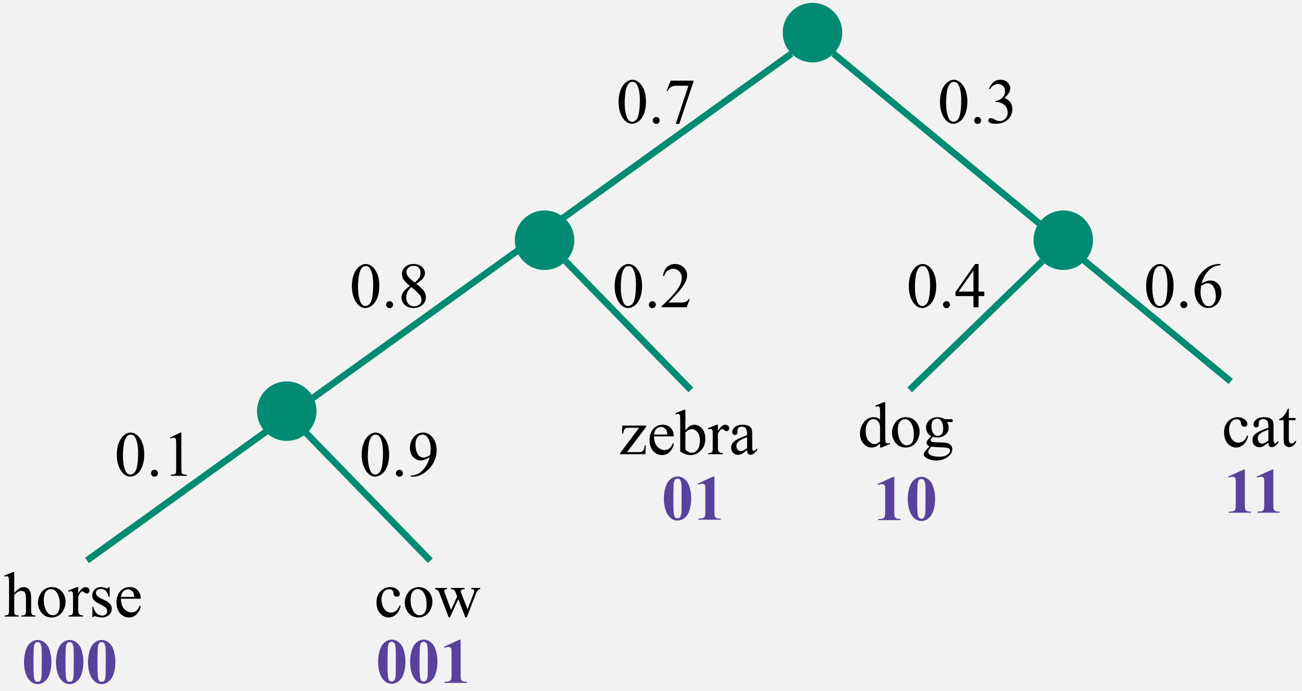
Hierarchical softmax

$$+\begin{array}{l} 0.7 \cdot 0.8 \\ 0.7 \cdot 0.2 \\ 0.3 \cdot 0.4 \\ 0.3 \cdot 0.6 \end{array}$$



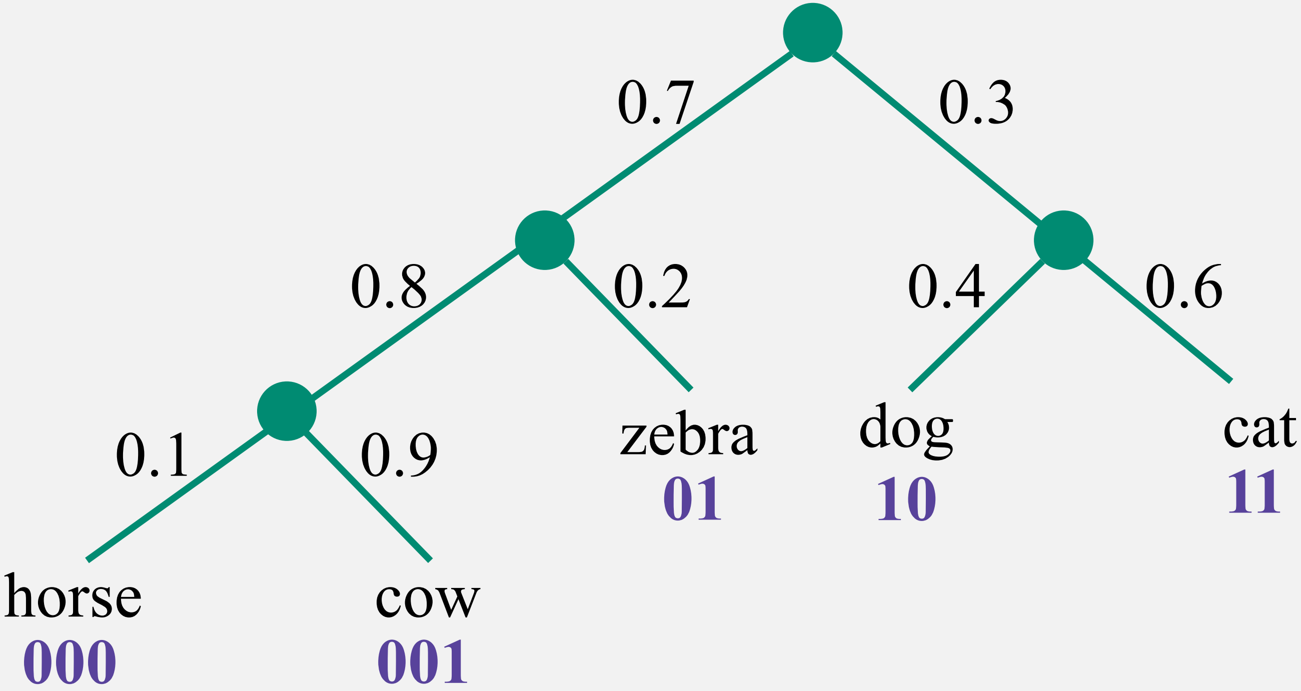
Hierarchical softmax

$$\begin{array}{rl} & 0.7 \cdot 0.8 \\ & 0.7 \cdot 0.2 \\ + & 0.3 \cdot 0.4 \\ & 0.3 \cdot 0.6 \end{array}$$



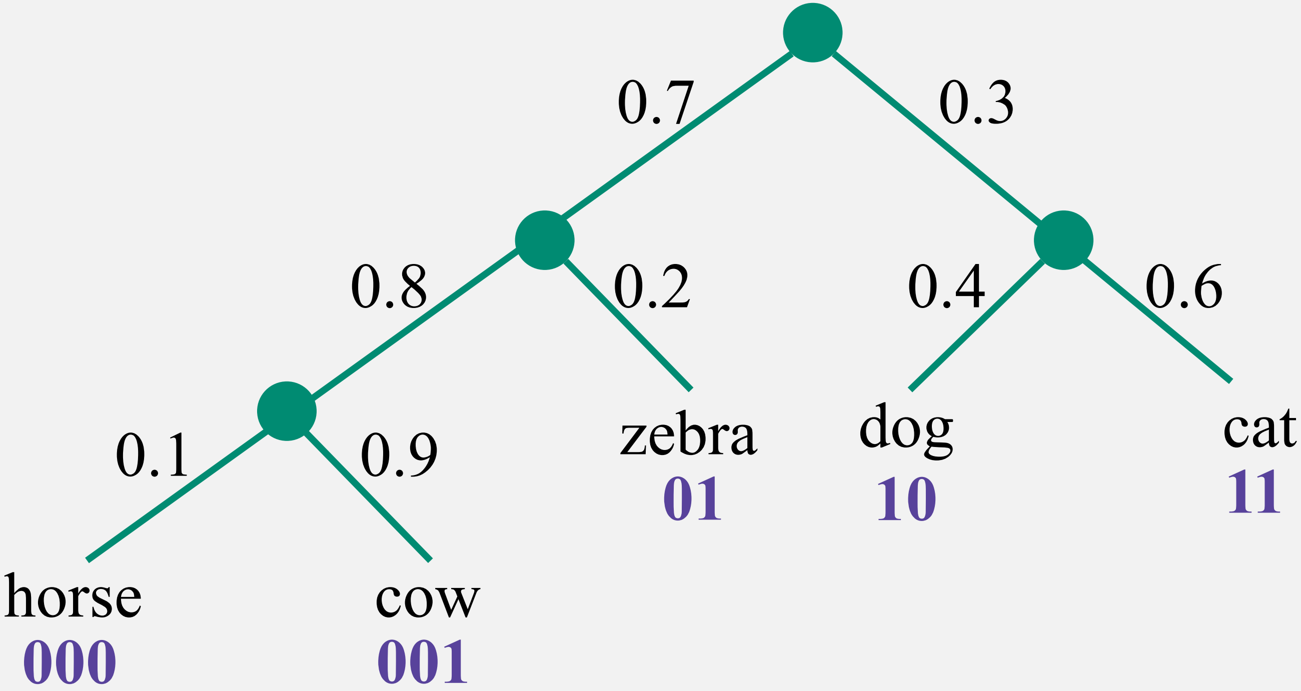
Hierarchical softmax

$$\begin{array}{ll} & 0.7 \\ & 0.3 \cdot 0.4 \\ + & 0.3 \cdot 0.6 \end{array}$$



Hierarchical softmax

$$\begin{array}{rl} & 0.7 \\ & 0.3 \cdot 0.4 \\ + & 0.3 \cdot 0.6 \end{array}$$

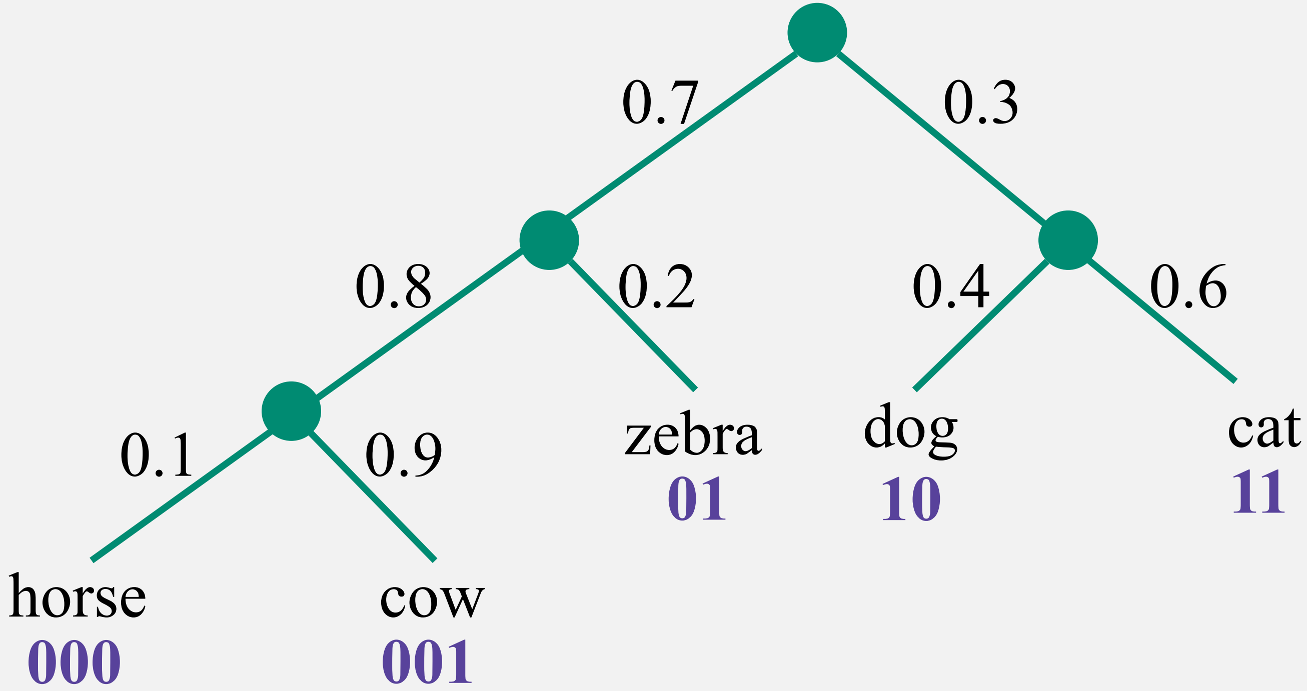


Hierarchical softmax

0.7

0.3

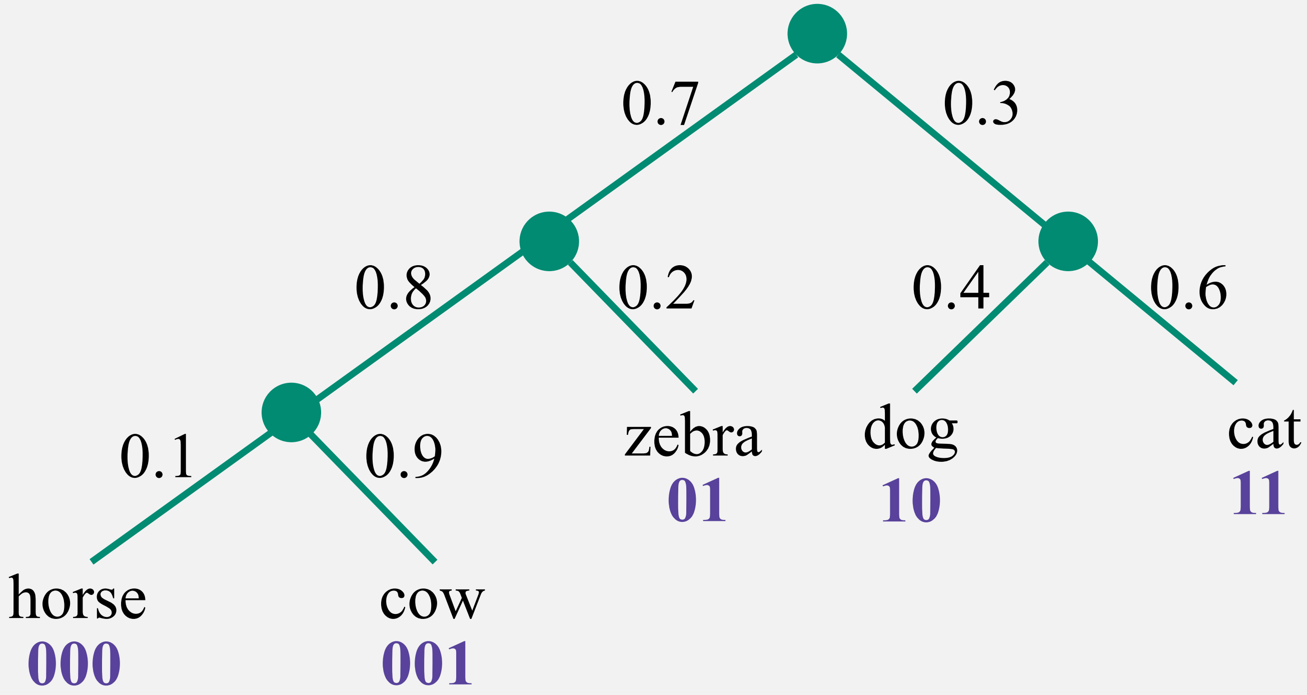
+



Hierarchical softmax

0.7

0.3

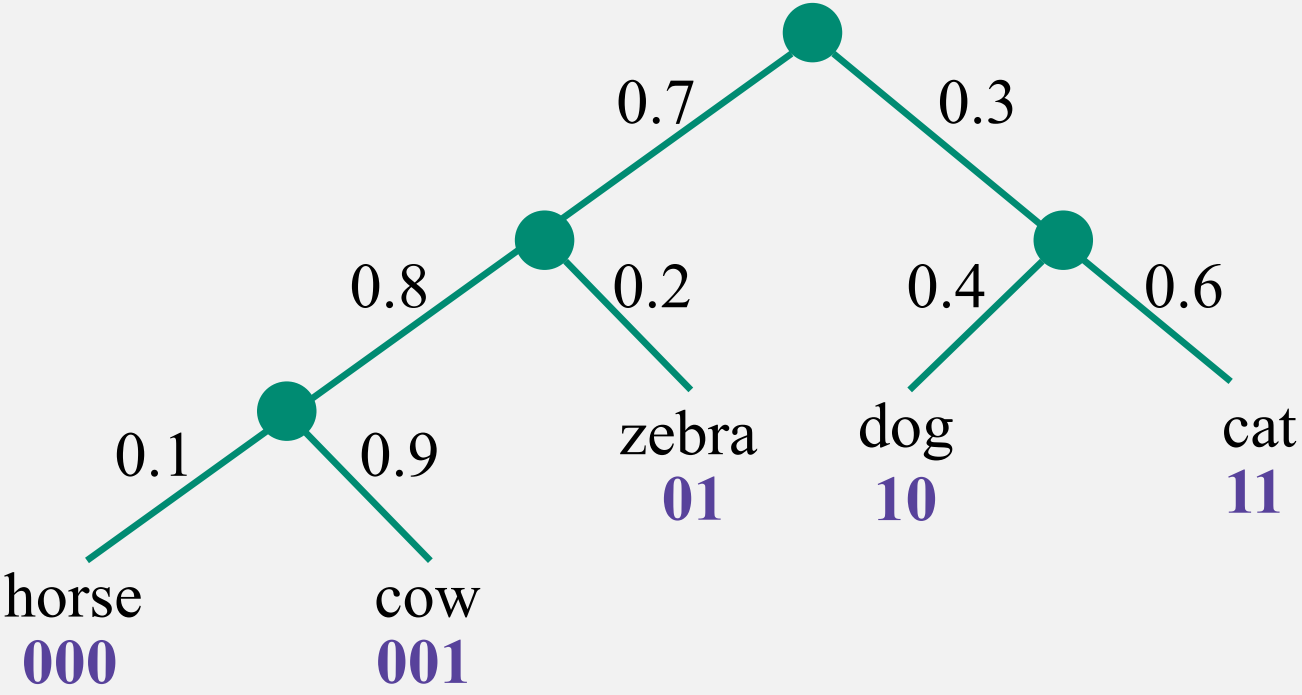


Hierarchical softmax

1.0

+

Congratulations!



Hierarchical softmax

Model binary decisions along the path in the tree:

$$p(w_n = w | w_1^{n-1}) = \prod_i p(d_i | w_1^{n-1})$$

How to construct a tree (balanced vs. semantic):

- Based on some pre-built ontology
- Based on semantic clustering from data
- Huffman tree
- Random

Outline

- Computing *softmax* for a large vocabulary is slow!
  - Hierarchical softmax
- Even a large vocabulary has *OOV words*:
  - **Copy mechanism**
  - Sub-word modeling
    - Word-character hybrid models
    - Byte-pair encoding

Copy mechanism

- Scaling *softmax* is insufficient!
- What do we do with OOV words?
  - Names, numbers, rare words…

	ecotax			*Pont-de-Buis*
The	UNK	portico	in	UNK

Copy mechanism

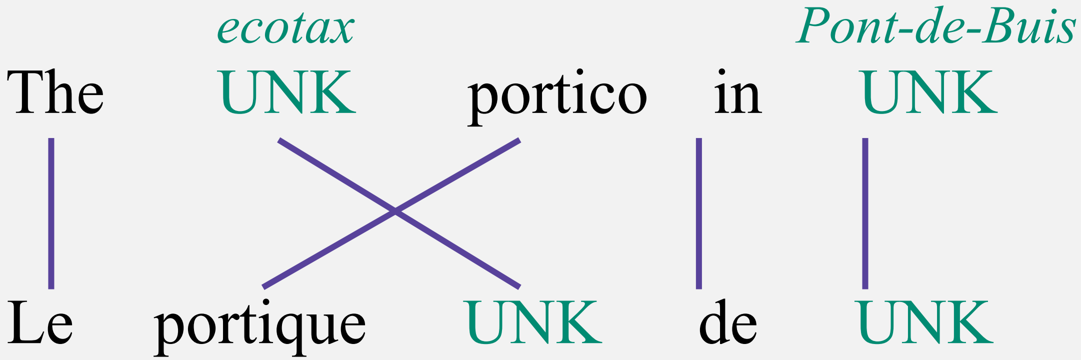
- Scaling *softmax* is insufficient!
- What do we do with OOV words?
  - Names, numbers, rare words…

	ecotax			*Pont-de-Buis*
The	UNK	portico	in	UNK

Le portique UNK de UNK

Copy mechanism

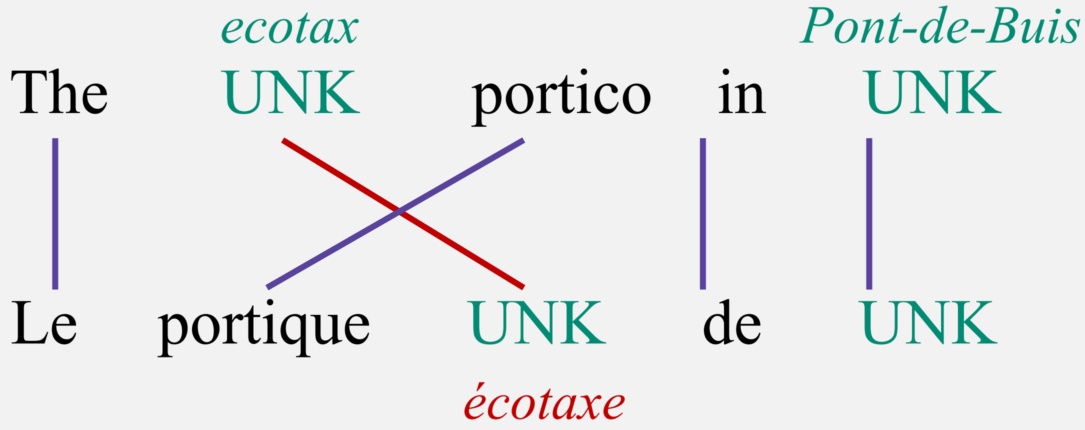
- Scaling *softmax* is insufficient!
- What do we do with OOV words?
  - Names, numbers, rare words…



Copy mechanism

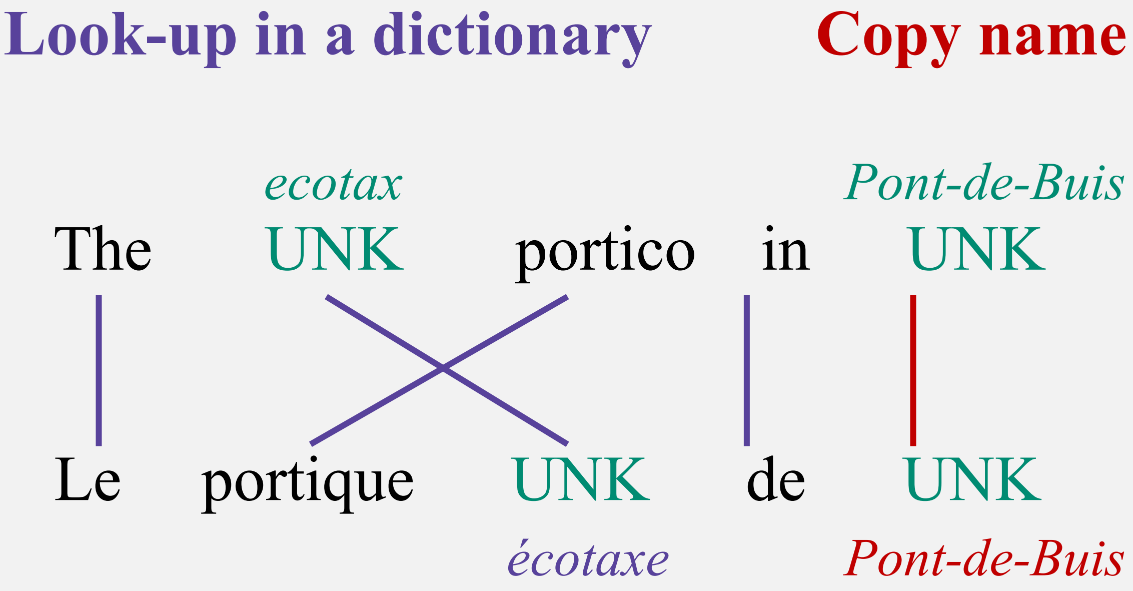
- Scaling *softmax* is insufficient!
- What do we do with OOV words?
  - Names, numbers, rare words…

Look-up in a dictionary



Copy mechanism

- Scaling *softmax* is insufficient!
- What do we do with OOV words?
  - Names, numbers, rare words…



Copy mechanism

Algorithm:

- Provide word alignments in train time
- Learn relative positions for UNK tokens with NMT
- Post-process the translation:
  - Copy the source word
  - Look up in a dictionary

Simple, but super useful technique!

Towards open vocabulary

Still problems:

- Transliteration: Christopher ↦ Kryštof
- Multi-word alignment: Solar system ↦ Sonnensystem
- Rich morphology: nejneobhospodařovávatelnějšímu
- Informal spelling: goooooood morning !!!!!

Outline

- Computing *softmax* for a large vocabulary is slow!
    - Hierarchical softmax
- Even a large vocabulary has *OOV words*:
    - Copy mechanism
    - **Sub-word modeling**
        - Word-character hybrid models
        - Byte-pair encoding

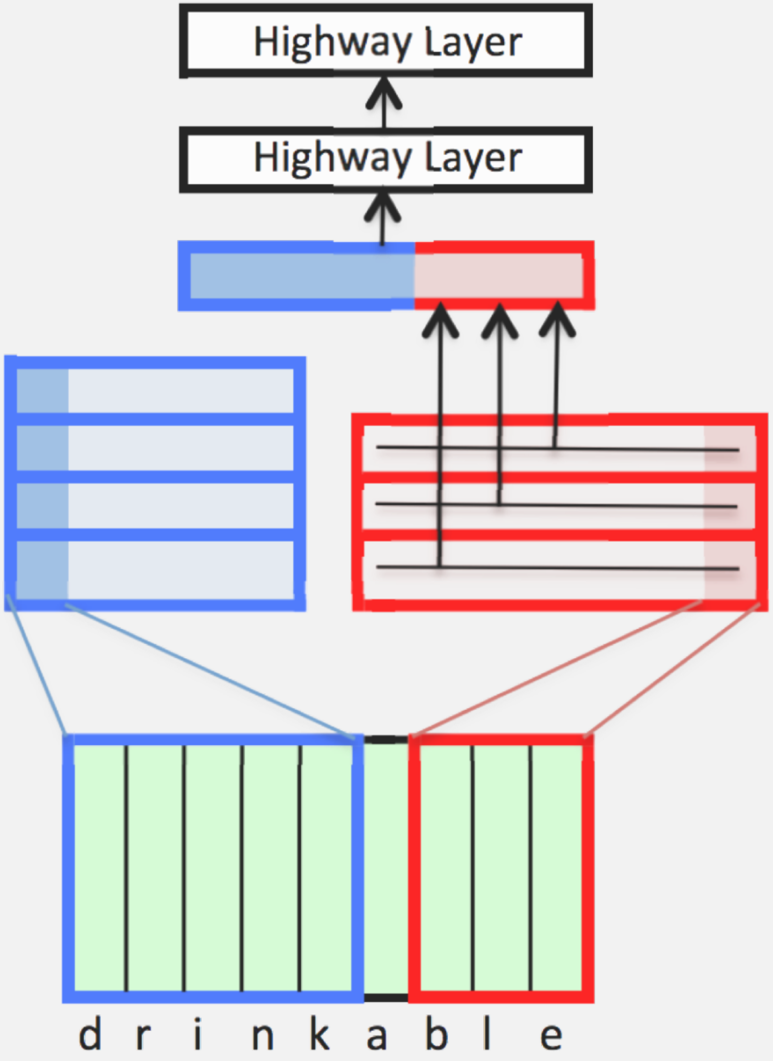
Outline

- Computing *softmax* for a large vocabulary is slow!
  - Hierarchical softmax
- Even a large vocabulary has *OOV words*:
  - Copy mechanism
  - Sub-word modeling
    - **Word-character hybrid models**
    - Byte-pair encoding

Character-based models

Character-based encoder is good for source languages with rich morphology!

- Bi-LSTMs to build word embeddings from characters
- CNNs on characters



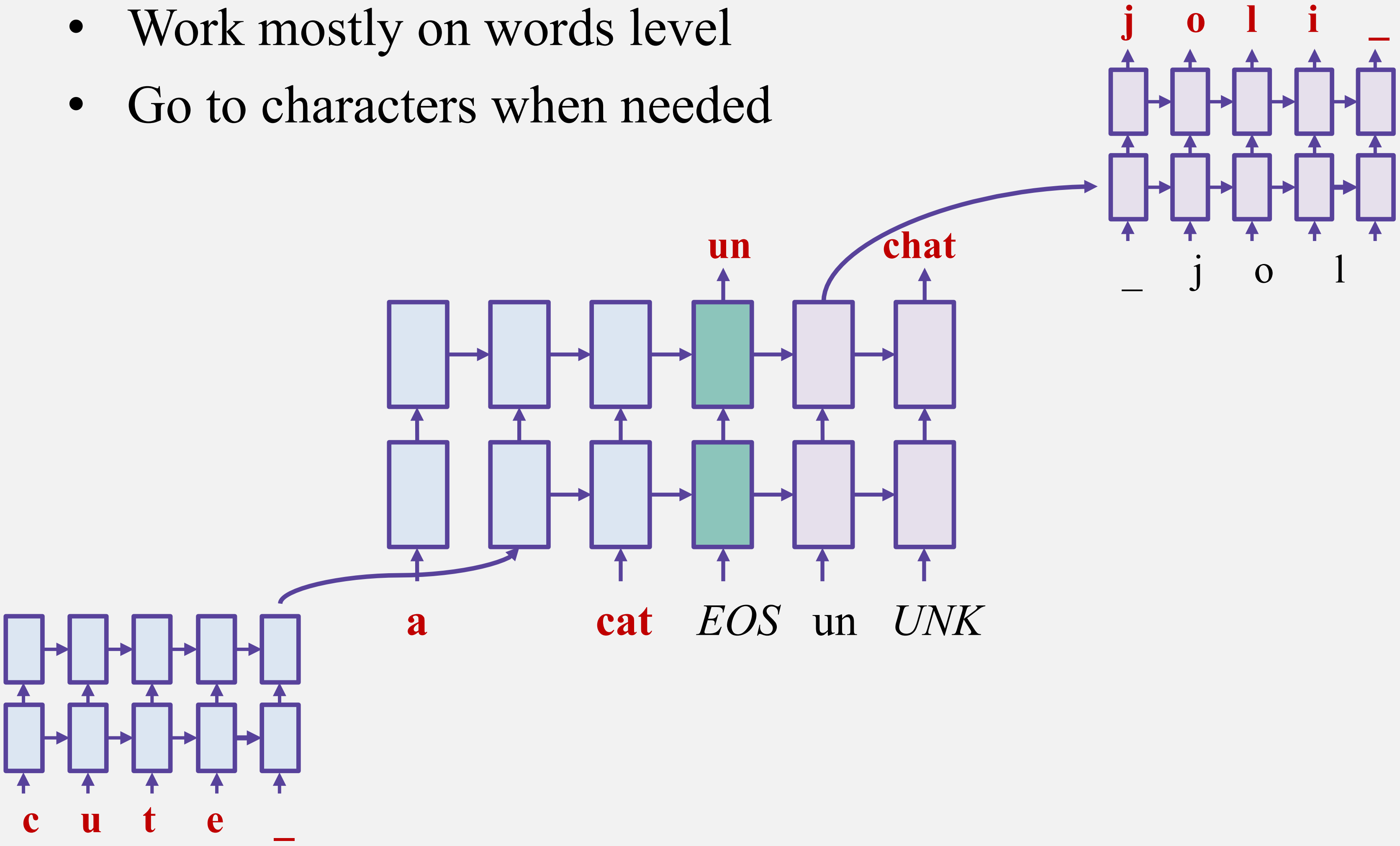
Ling, et. al. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. EMNLP 2015.

Kim, et. al. Character-Aware Neural Language Models. AAAI 2016.

Marta R. Costa-jussà and José A. R. Fonollosa. Character-based Neural Machine Translation. ACL 2016.

Hybrid models: the best of two worlds

- Work mostly on words level
- Go to characters when needed



Thang Luong and Chris Manning. Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models. ACL 2016.

Outline

- Computing *softmax* for a large vocabulary is slow!
    - Hierarchical softmax
- Even a large vocabulary has *OOV words*:
    - Copy mechanism
    - Sub-word modeling
        - Word-character hybrid models
        - **Byte-pair encoding**

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

She sells seashells by the seashore

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

S h e _ s e l l s _ s e a s h e l l s _ b y _ t h e _ s e a s h o r e _

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

S h e _ s e l l s _ s e a s h e l l s _ b y _ t h e _ s e a s h o r e _

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

Sh e _ s e l l s _ s e a sh e l l s _ b y _ t h e _ s e a sh o r e _

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

Sh e _ s e l l s _ s e a sh e l l s _ b y _ t h e _ s e a sh o r e _

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

Sh e _ se l l s _ se a sh e l l s _ b y _ t h e _ se a sh o r e _

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

Sh e _ se l l s _ se a sh e l l s _ b y _ t h e _ se a sh o r e _

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

Sh e _ se ll s _ se a sh e ll s _ b y _ t h e _ se a sh o r e _

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

Sh e _ se ll s _ se a sh e ll s _ b y _ t h e _ se a sh o r e _

Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

Sh e _ se ll s _ sea sh e ll s _ b y _ t h e _ sea sh o r e _

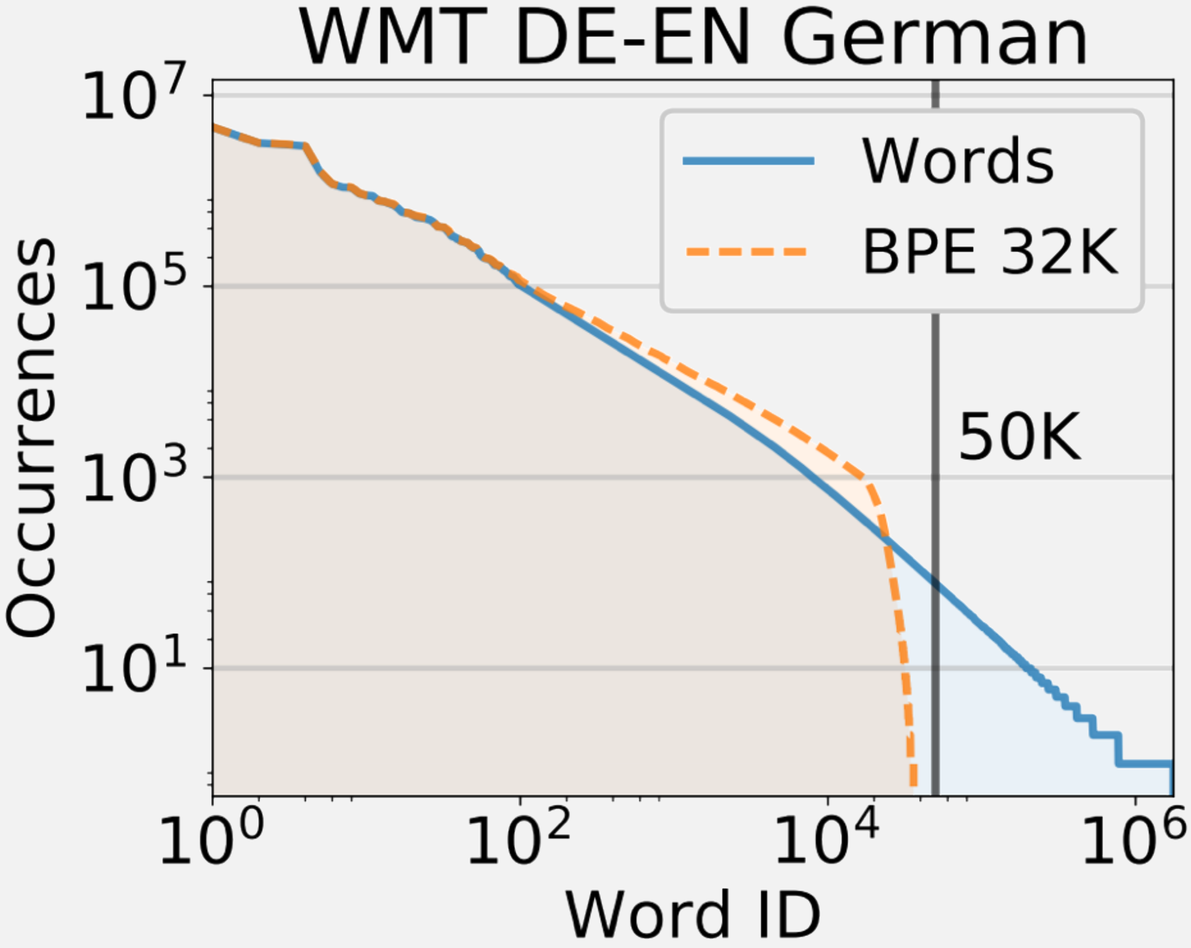
Byte-pair encoding

- Simple way to handle open vocabulary:
  - Start with characters
  - Iteratively replace the most frequent pair with one unit

Sh e _ se ll s _ sea sh e ll s _ b y _ t h e _ sea sh o r e _

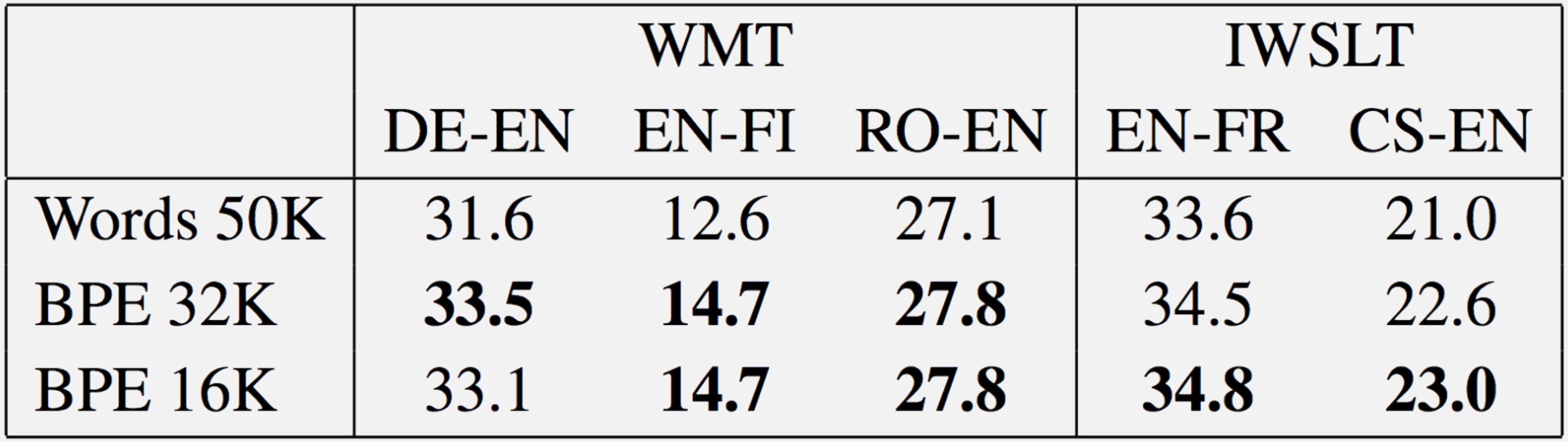
  - End whenever you reach the vocabulary size limit
  - Stick to that vocabulary of sub-word units
  - Apply the same algorithm to test sentences

Why is it so useful?



Denkowski, Neubig. Stronger Baselines for Trustable Results in Neural Machine Translation, 2017.

BLEU score comparison



	WMT			IWSLT	
	DE-EN	EN-FI	RO-EN	EN-FR	CS-EN
Words 50K	31.6	12.6	27.1	33.6	21.0
BPE 32K	**33.5**	**14.7**	**27.8**	34.5	22.6
BPE 16K	33.1	**14.7**	**27.8**	**34.8**	**23.0**

- Byte-pair encoding improves BLEU score
- It is a nice and simple way to handle the vocabulary
- Very common trick in modern NMT

Denkowski, Neubig. Stronger Baselines for Trustable Results in Neural Machine Translation, 2017.