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

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

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

1-grams: 4 / 6

2-grams: 3 / 5

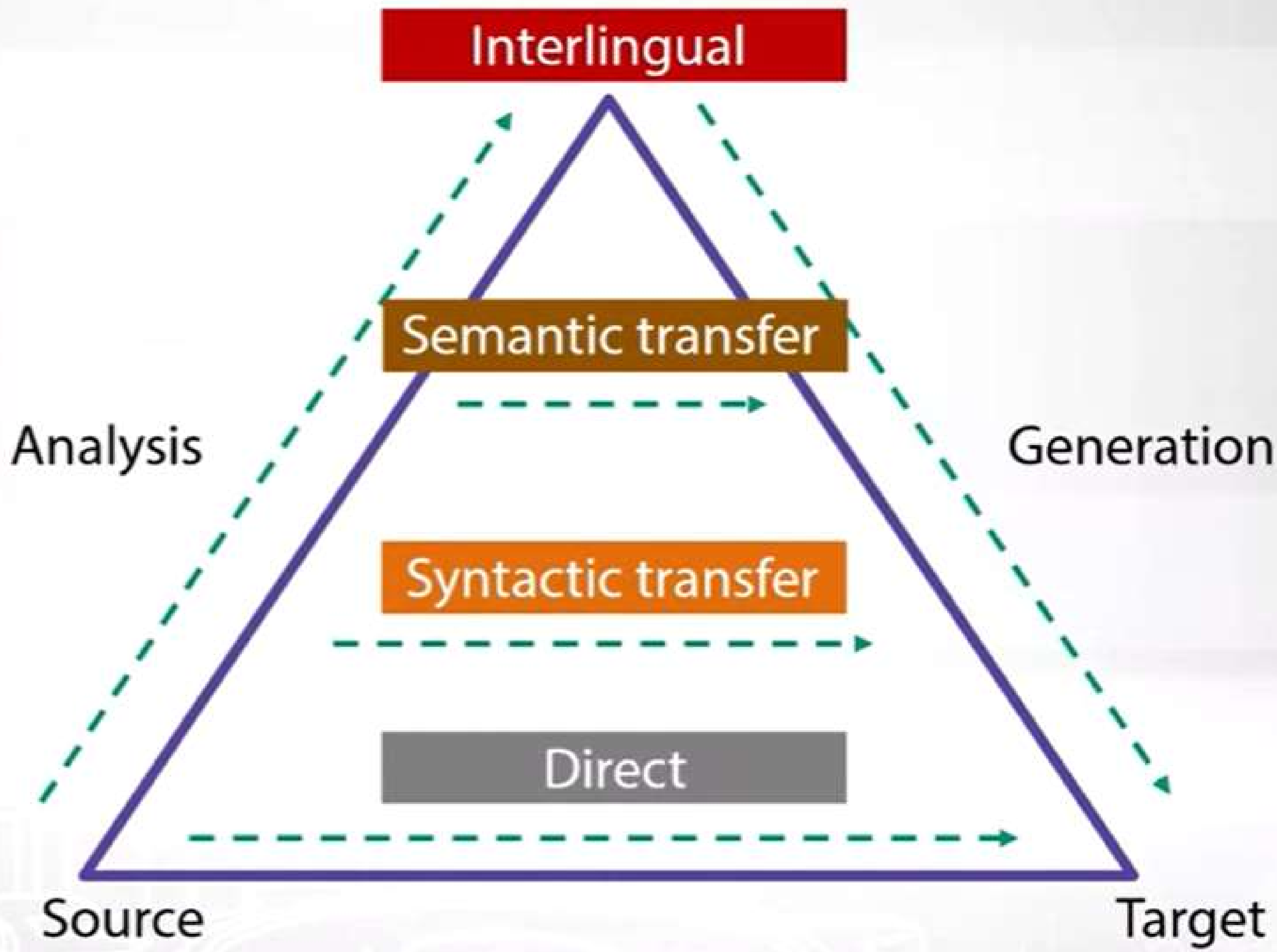
3-grams: 2 / 4

4-grams: 1 / 3

Brevity: min(1, 6/5)

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

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

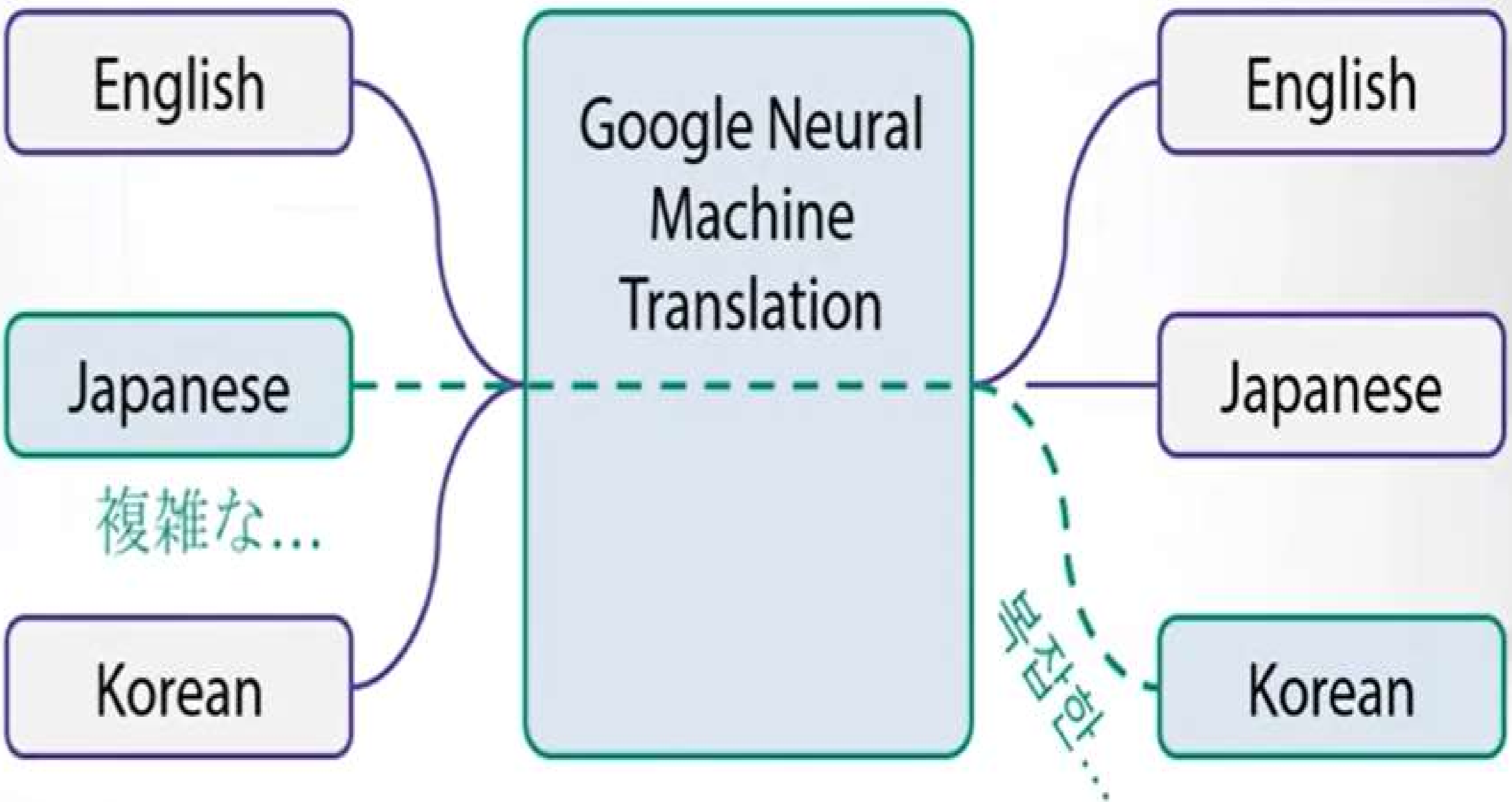
Statistical Machine Translation (SMT):

- 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



The main equation

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

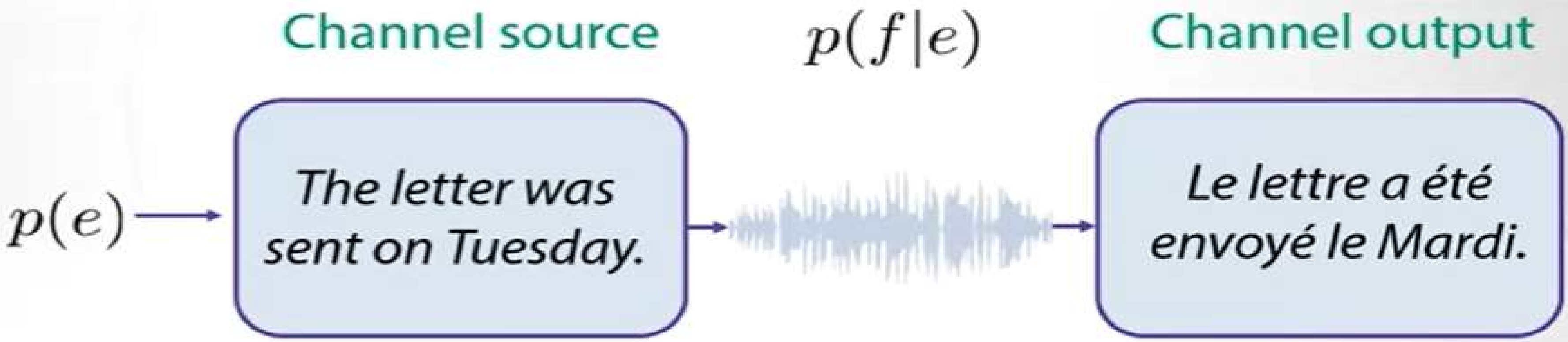
$$e^* = \operatorname*{argmax}_{e \in E} p(e|f) = \operatorname*{argmax}_{e \in E} \frac{p(f|e)p(e)}{p(f)} =$$

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

Why is it easier to deal with?

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

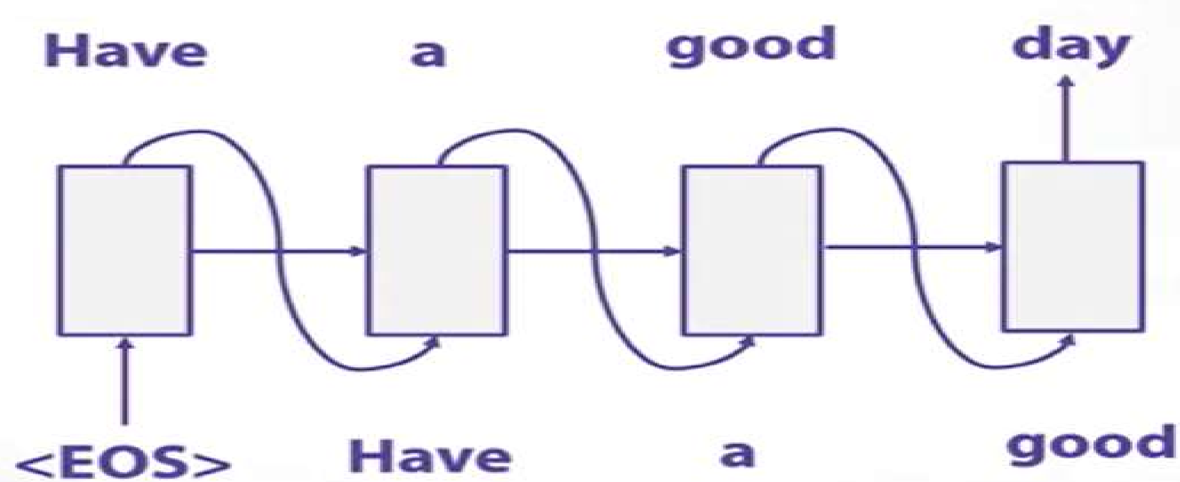
- $p(e)$ models the *fluency* of the translation
- $p(f|e)$ models the *adequacy* of the translation
- argmax is the search problem implemented by a *decoder*



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				V_f
	0.1							
		0.1	0.2	0.4			0.1	
			0.8			0.2		
	0.2	0.3			0.5			
wool		0.2		$\boxed{0.7}$		0.1		$p(f_j\|e_i)$
			0.9				0.1	
V_e								

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 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.

a?

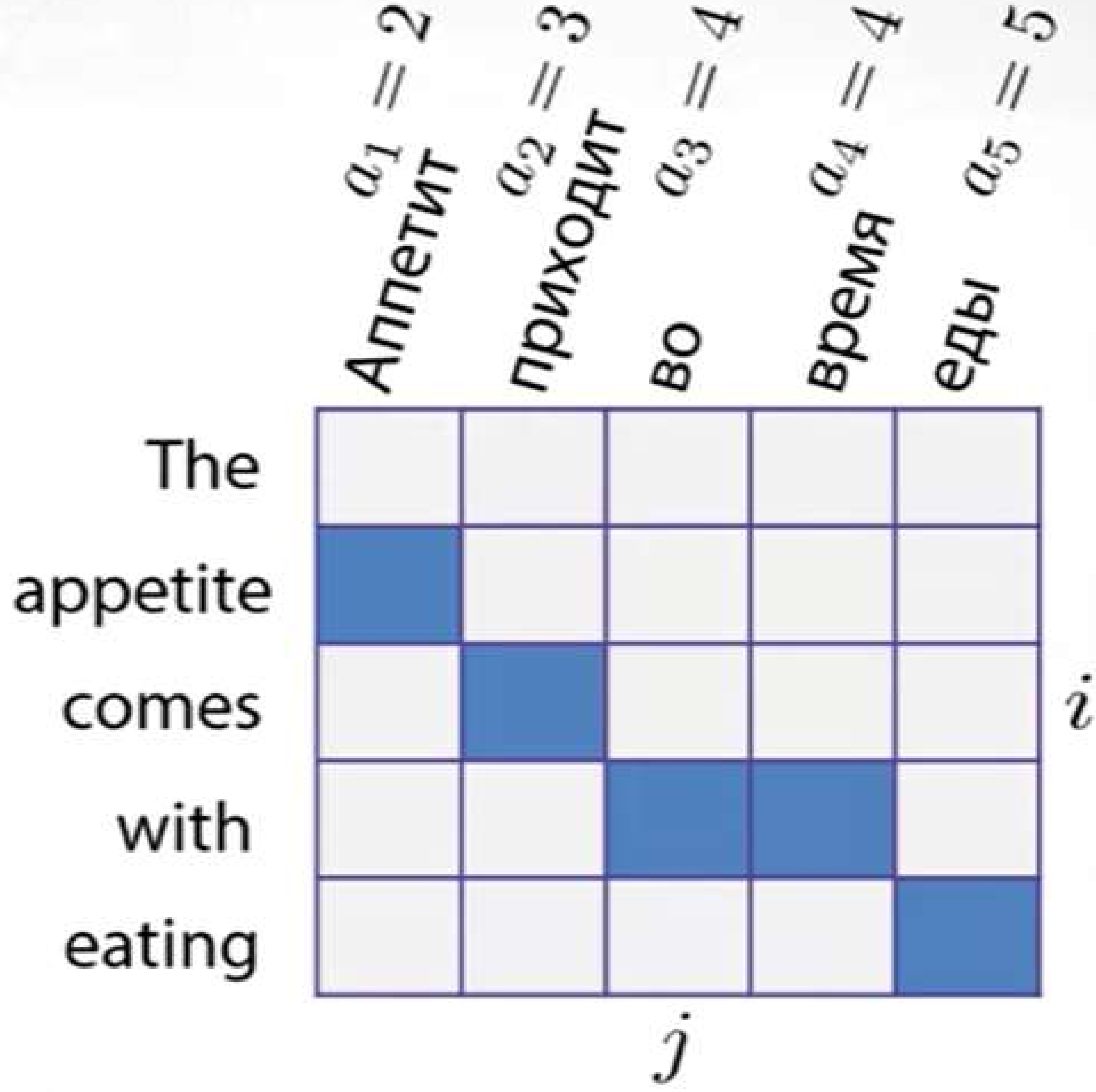
f: Аппетит приходит во время еды.

Recap: Bayes' rule

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

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

Word alignment matrix



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**:

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

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:

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

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

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

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

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

$$p(f, a|e) = p(J|e) \prod_{j=1}^{J} \underbrace{p(a_j)}_{\substack{\text{Uniform prior} \\ \varepsilon}} \underbrace{p(f_j|a_j, e)}_{\substack{\text{Translation table} \\ t(f_j|e_{a_j})}}$$

\+ The model is simple and has not too many parameters

– The alignment prior does not depend on word positions

Translation table

V_f

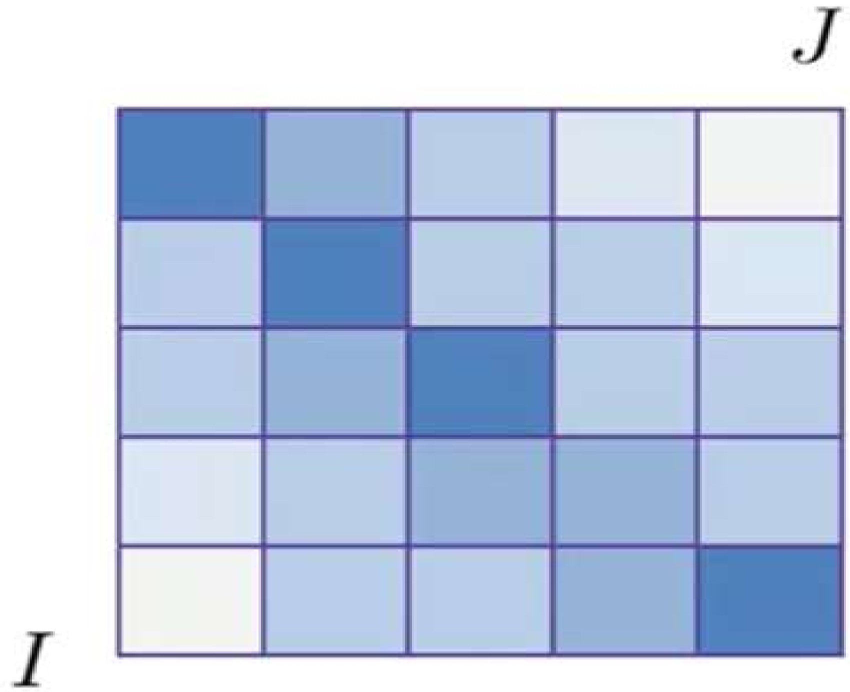
				шерсть			
	0.1						
		0.1	0.2	0.4			0.1
			0.8			0.2	
	0.2	0.3			0.5		
wool		0.2		$\boxed{0.7}$		0.1	
			0.9				0.1

V_e

$p(f_j|e_i)$

Position-based prior

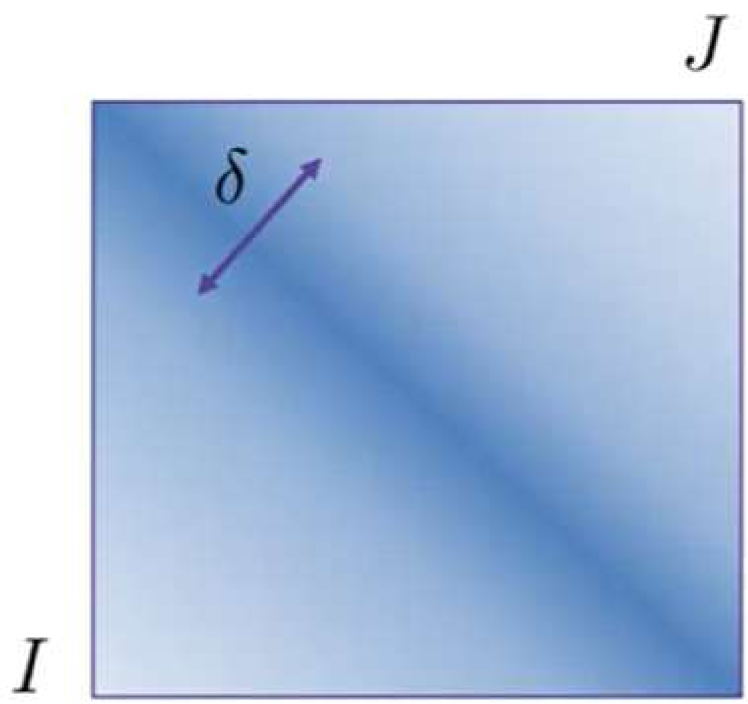
- For each pair of the **lengths** of the sentences:
  - $I \times J$ matrix of probabilities



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

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

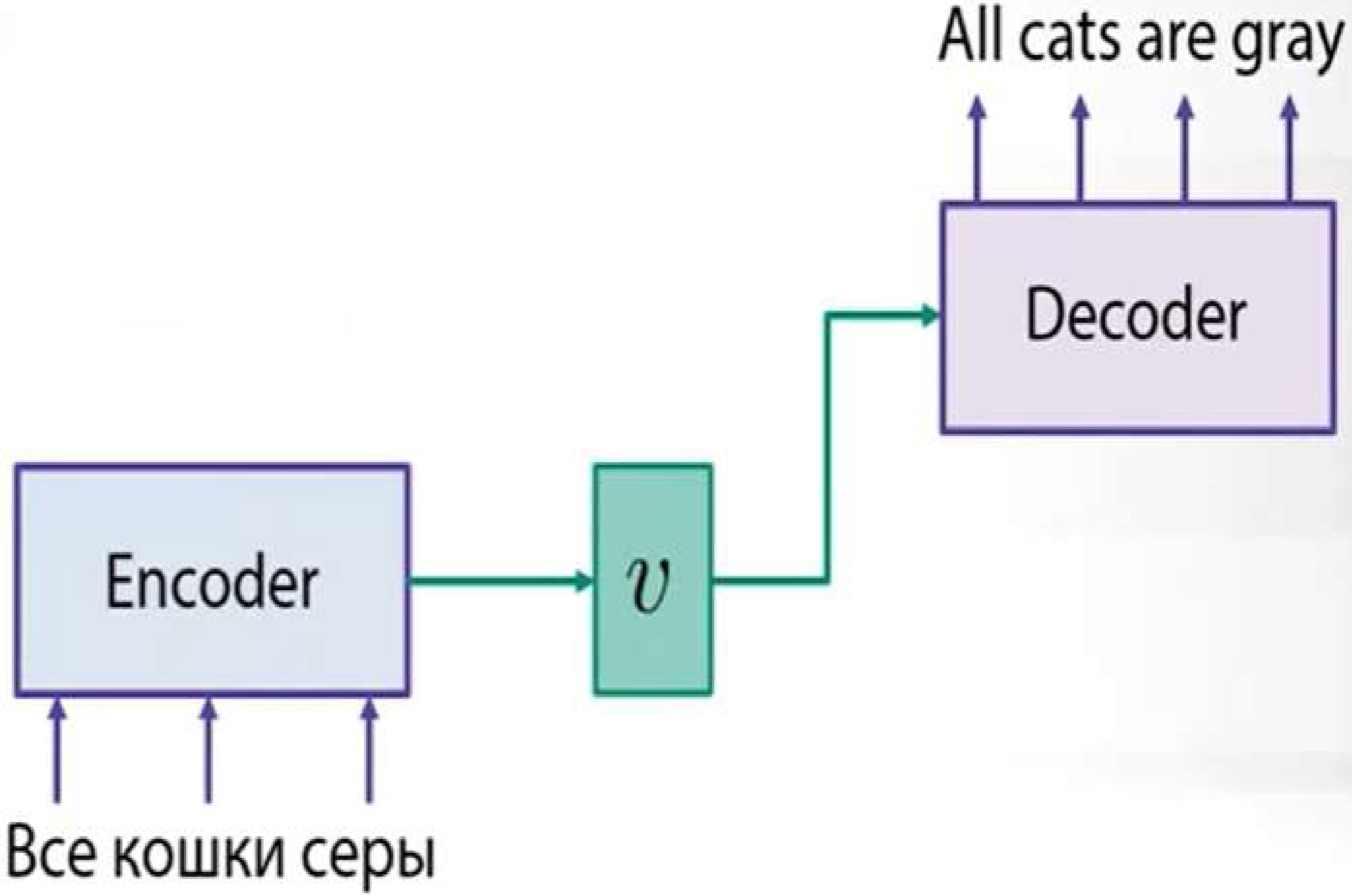
Transition probabilities $d(a_j|a_{j-1}, I, J)$

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

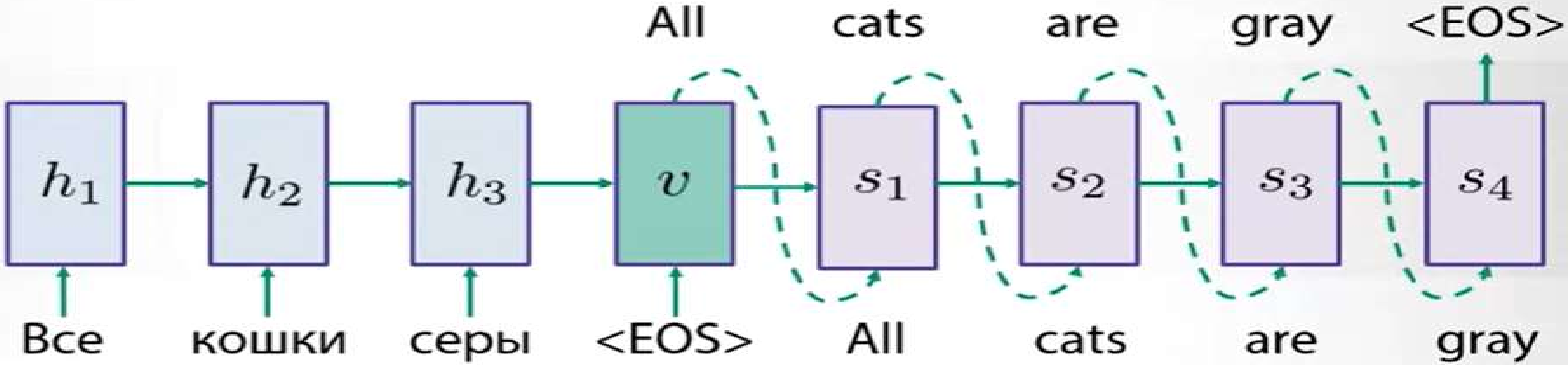
e: All cats are grey in the dark.

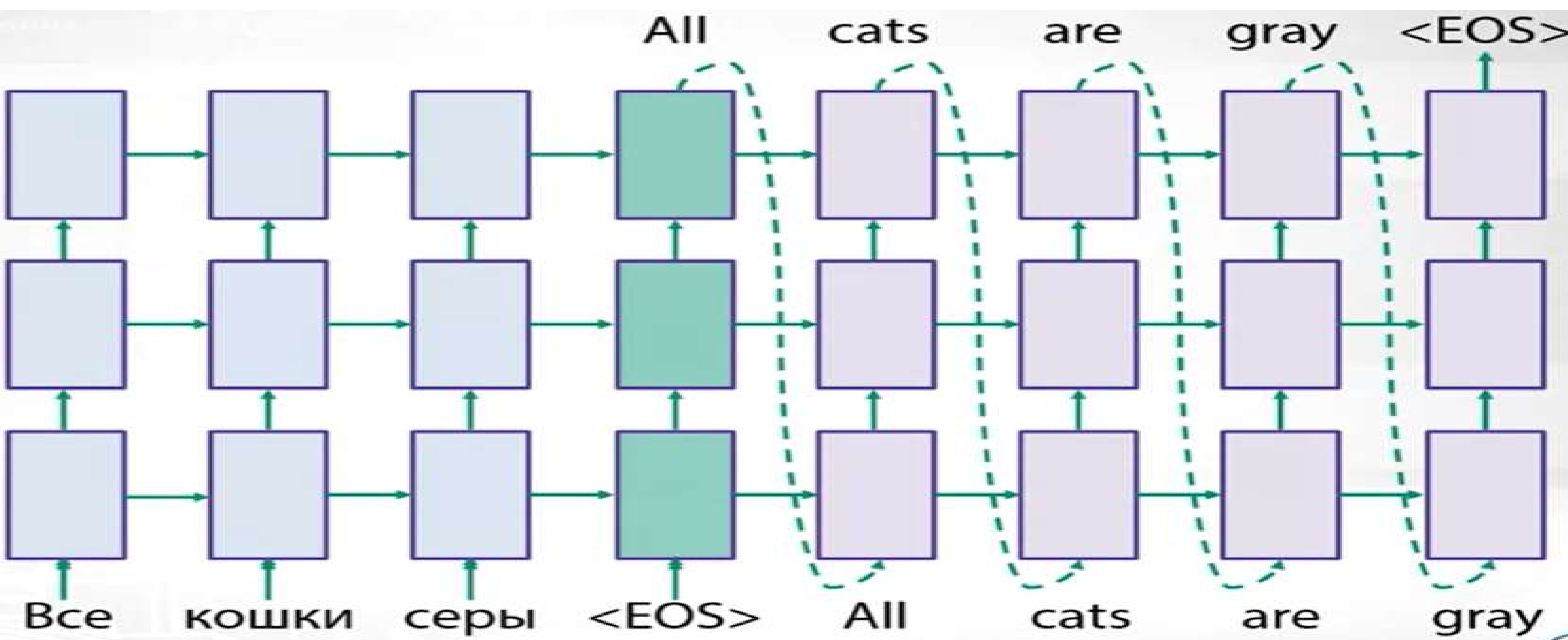
f: В темноте все кошки серы.

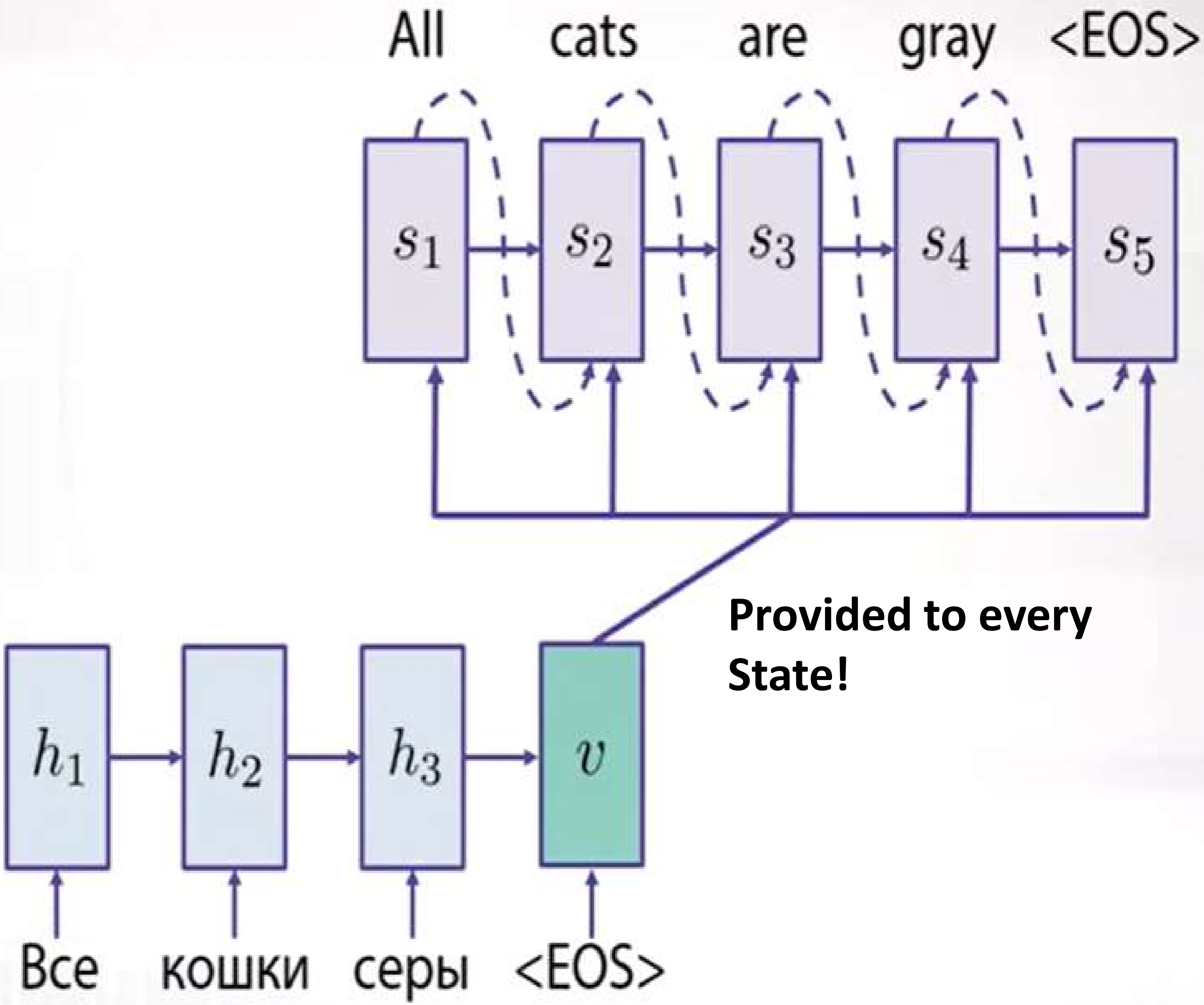
Sequence to sequence



Sequence to sequence






All
cats
are
gray
<EOS>
s_1
s_2
s_3
s_4
s_5
Provided to every State!
h_1
h_2
h_3
v
Все
кошки
серы
<EOS>

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

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

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

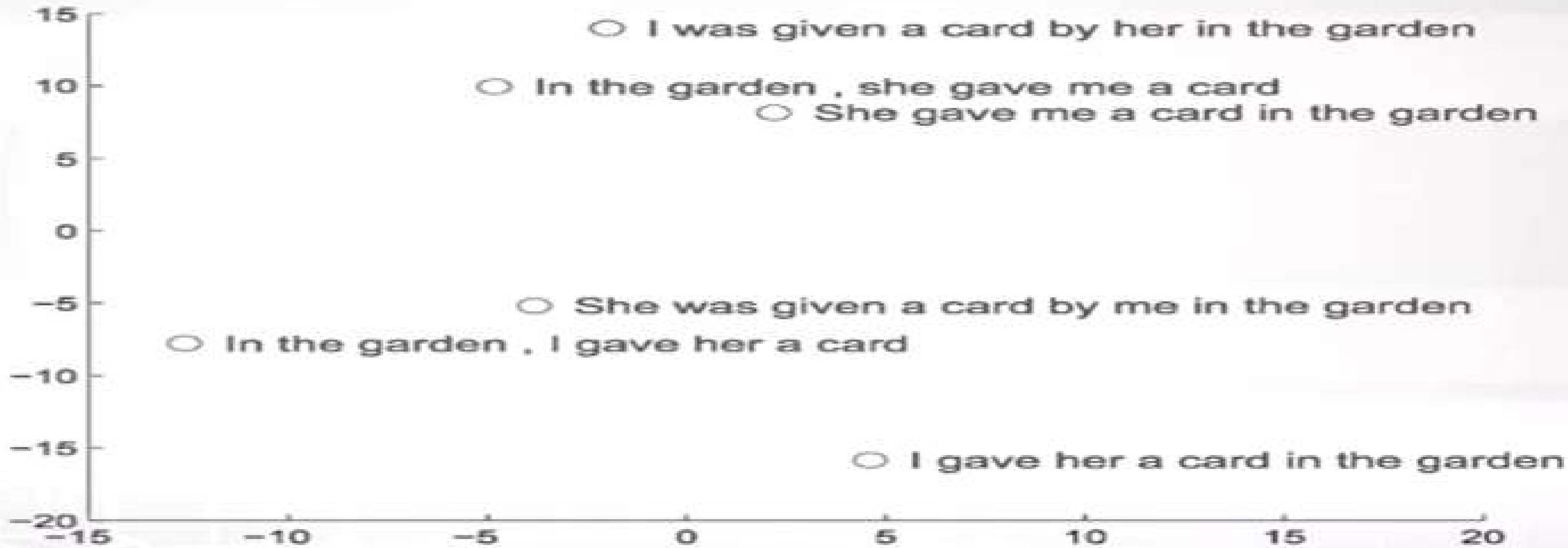
- **Decoder:** performs language modeling given this vector

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

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

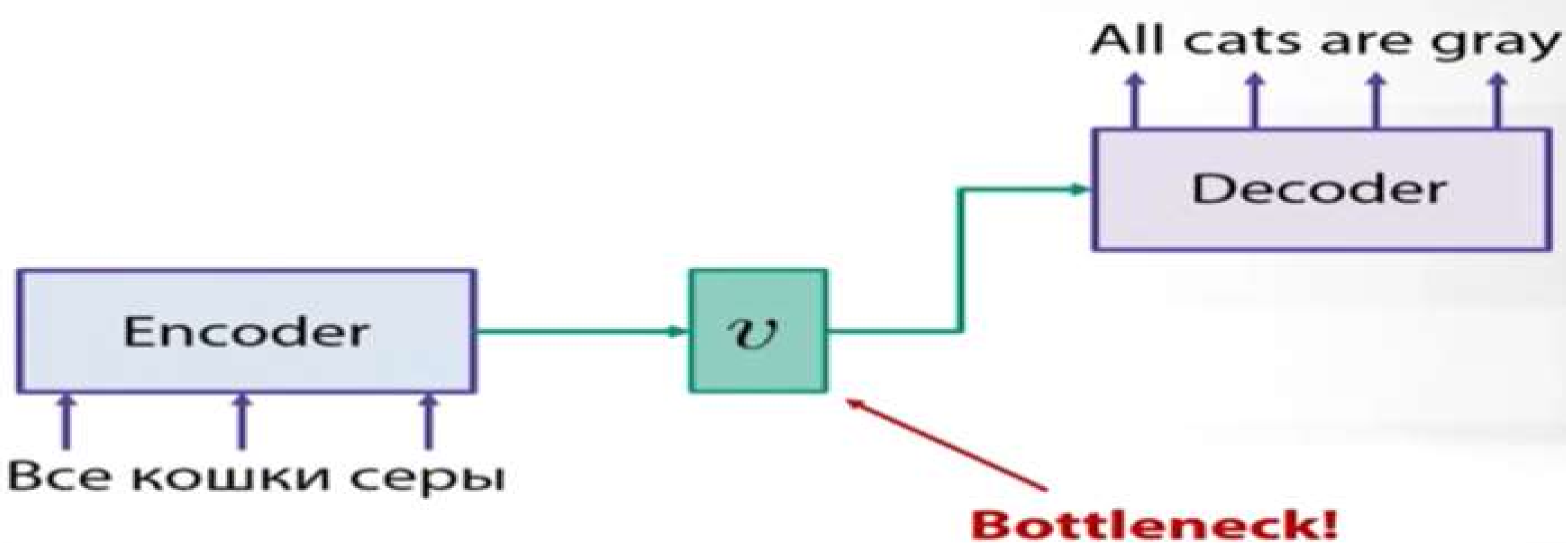
$$p(y_j | v, y_1, \dots y_{j-1}) = softmax\left(U s_j + b\right)$$

Hidden representations are good...

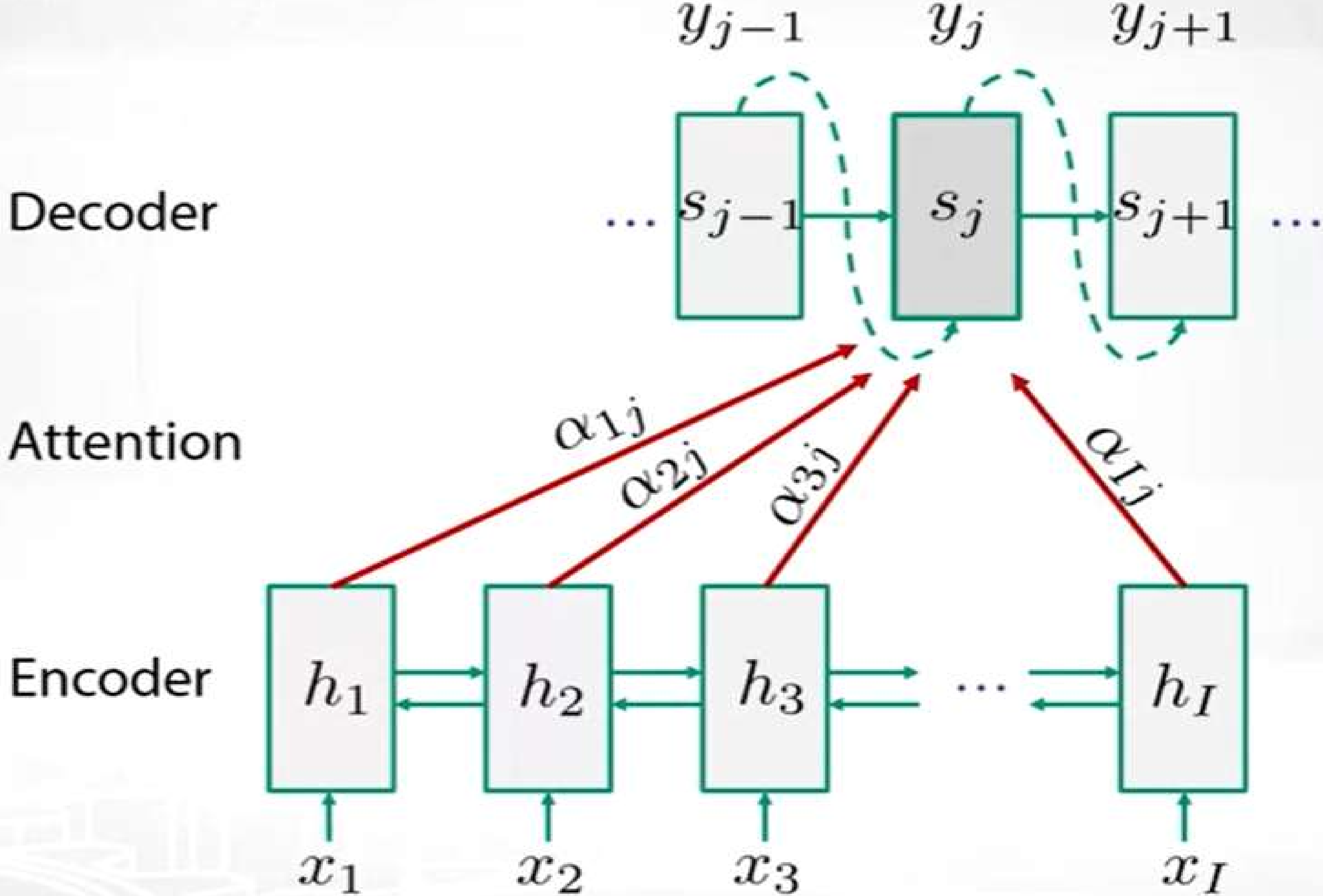


Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with Neural Network, 2014.

... but still a bottleneck



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

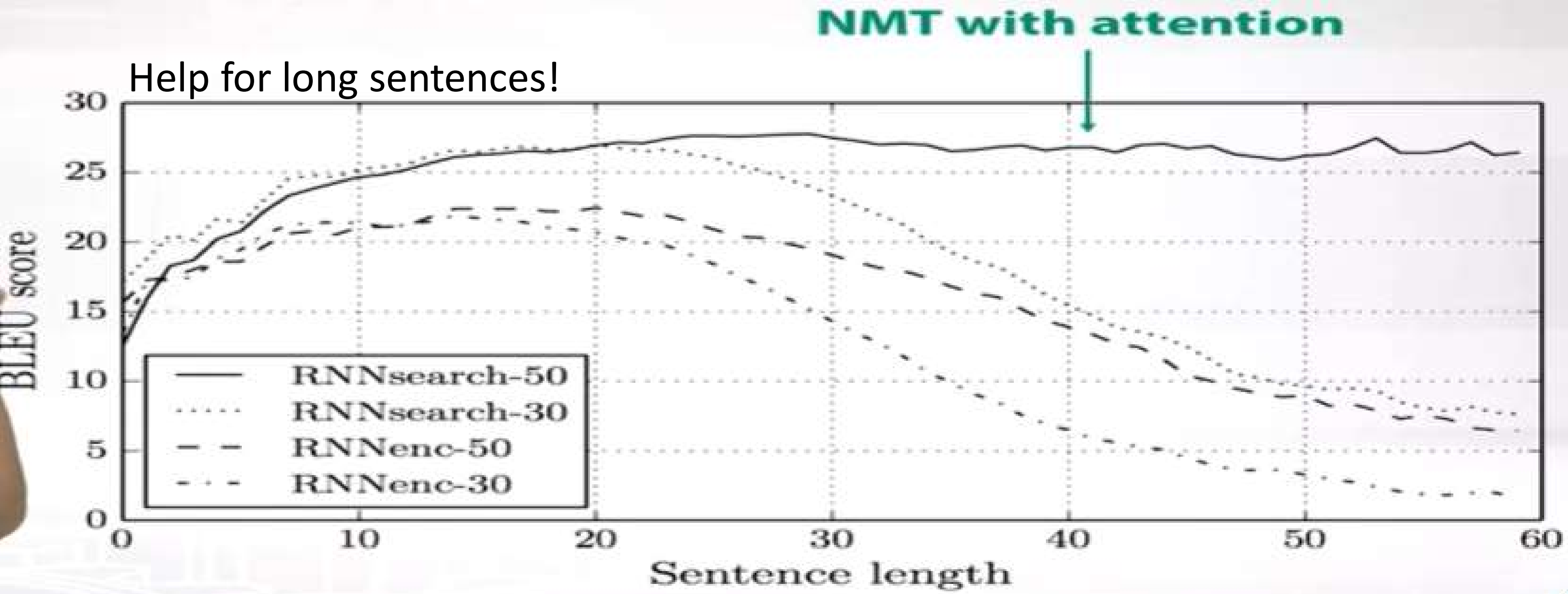
$$p(y_1, \ldots y_J | x_1, \ldots x_I) = \prod_{j=1}^{J} p(y_j | v_j, y_1, \ldots y_{j-1})$$

- Still encoder-decoder architecture with RNNs:

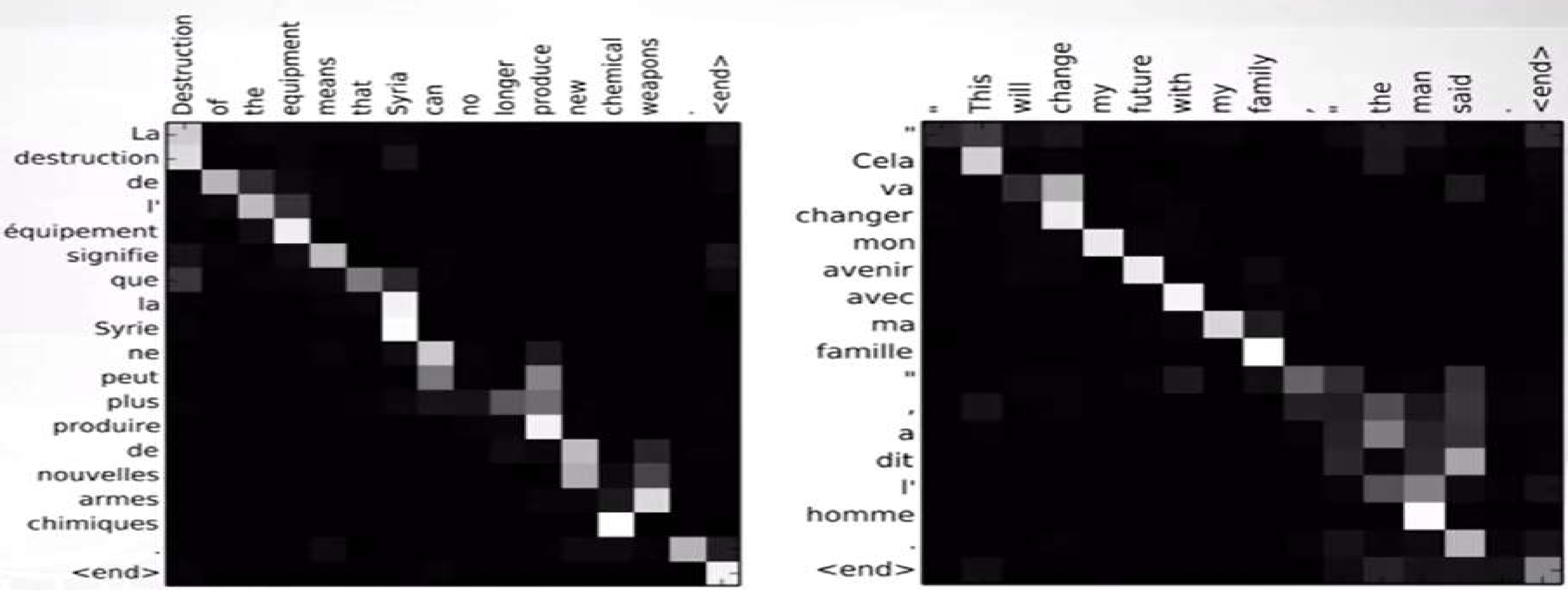
$$h_i = f(h_{i-1}, x_i) \qquad s_j = g(s_{j-1}, [y_{j-1}, v_j])$$

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

Help for long sentences!



Example: attention (alignments)



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 $\mathrm{a_j}$ 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 $[\mathrm{a_j} - \mathrm{h}; \mathrm{a_j} + \mathrm{h}]$**

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

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**

DEALING WITH VOCABULARY:

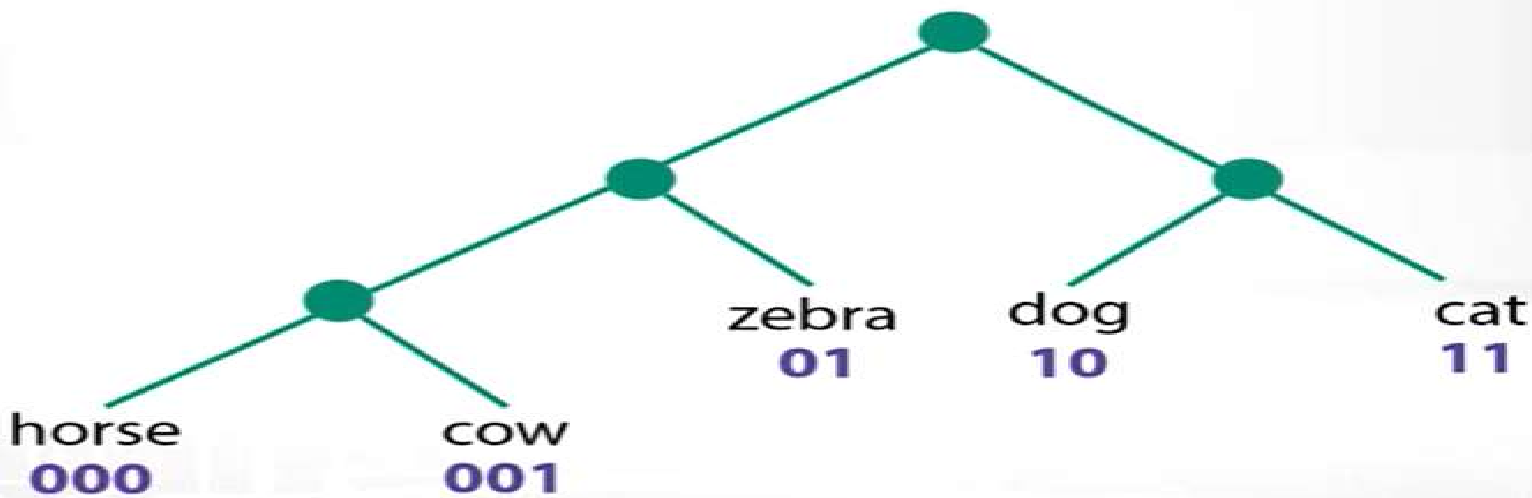
Outline

- 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

Hierarchical softmax

Each word is uniquely represented by a binary code:

- 0 means "go left", 1 means "go right"



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

Model binary decisions along the path in the tree:

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

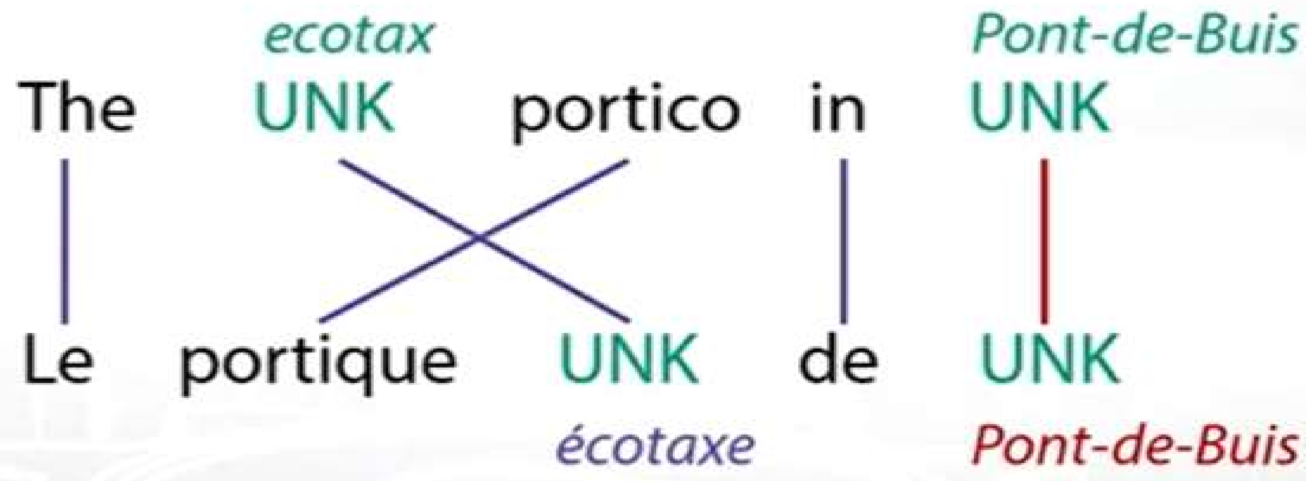
How to construct a tree (balanced vs. semantic):

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

Copy mechanism

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

Look-up in a dictionary **Copy name**



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

Towards open vocabulary

Still problems:

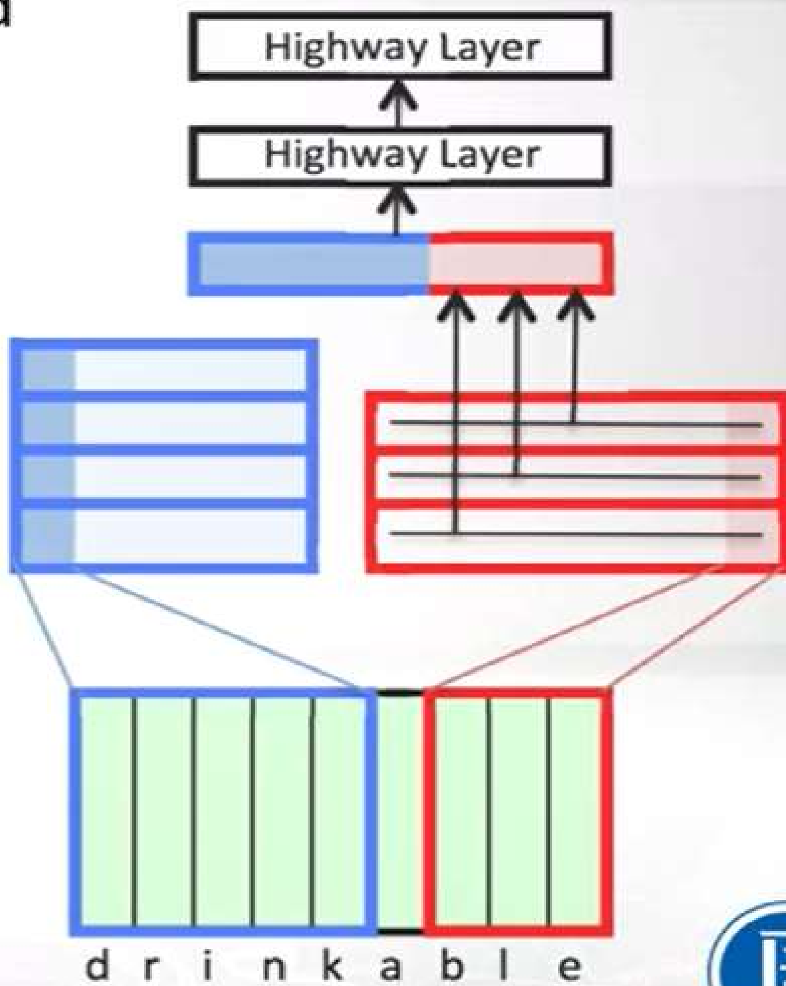
- Transliteration: Christopher ↦ Kryštof
- Multi-word alignment: Solar system ↦ Sonnensystem
- Rich morphology: nejneobhospodařovávatelnějšímu
- Informal spelling: goooooood morning !!!!!

SUB WORD MODELS:

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



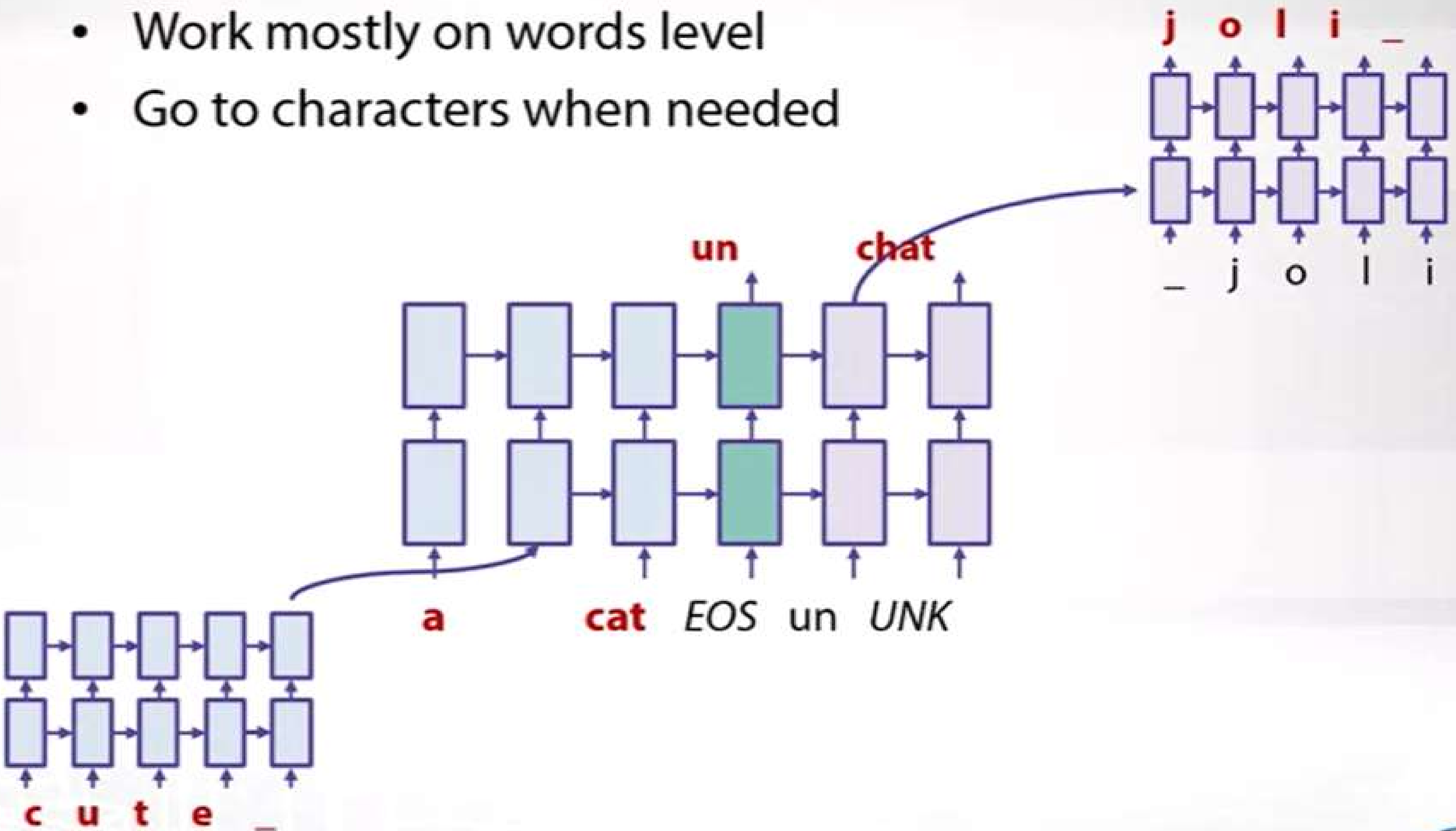
Ling, et. al. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. EMNLP 2015.

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.

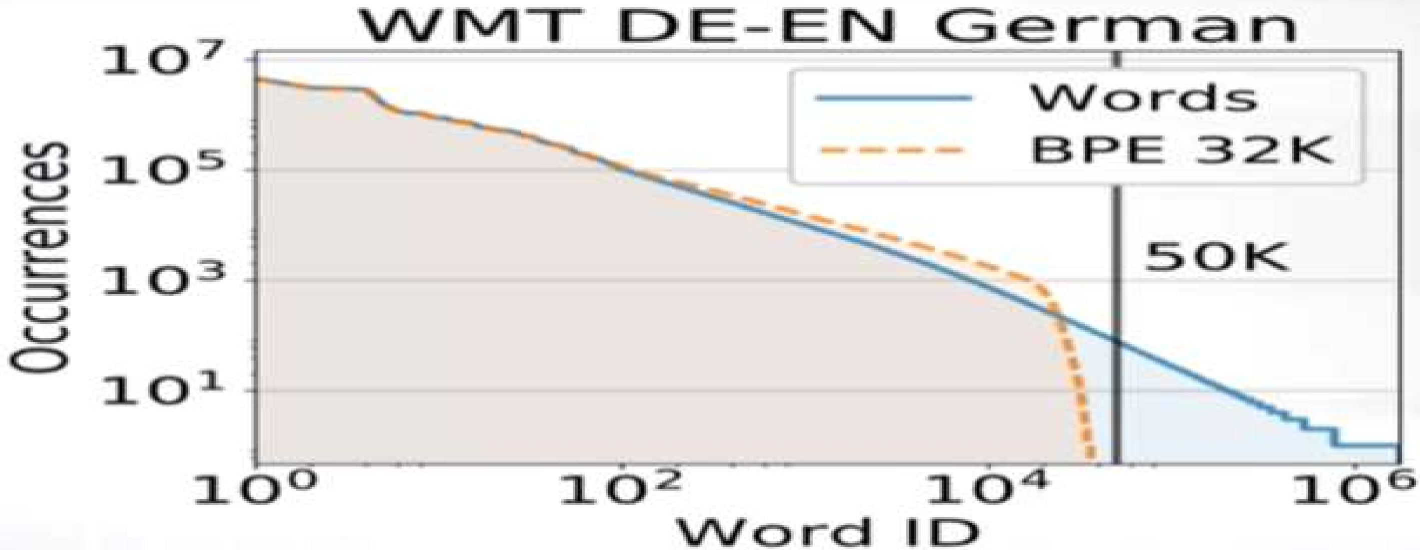
Byte-pair encoding

- 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?



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

Sequence to sequence

- Machine Translation
- Summarization
- Text simplification
- Language to code
- Chit-chat bot
- Question answering
- Listen, attend and spell: speech recognition
- Show, attend and tell: image caption generation
-

Summarization

Original Text

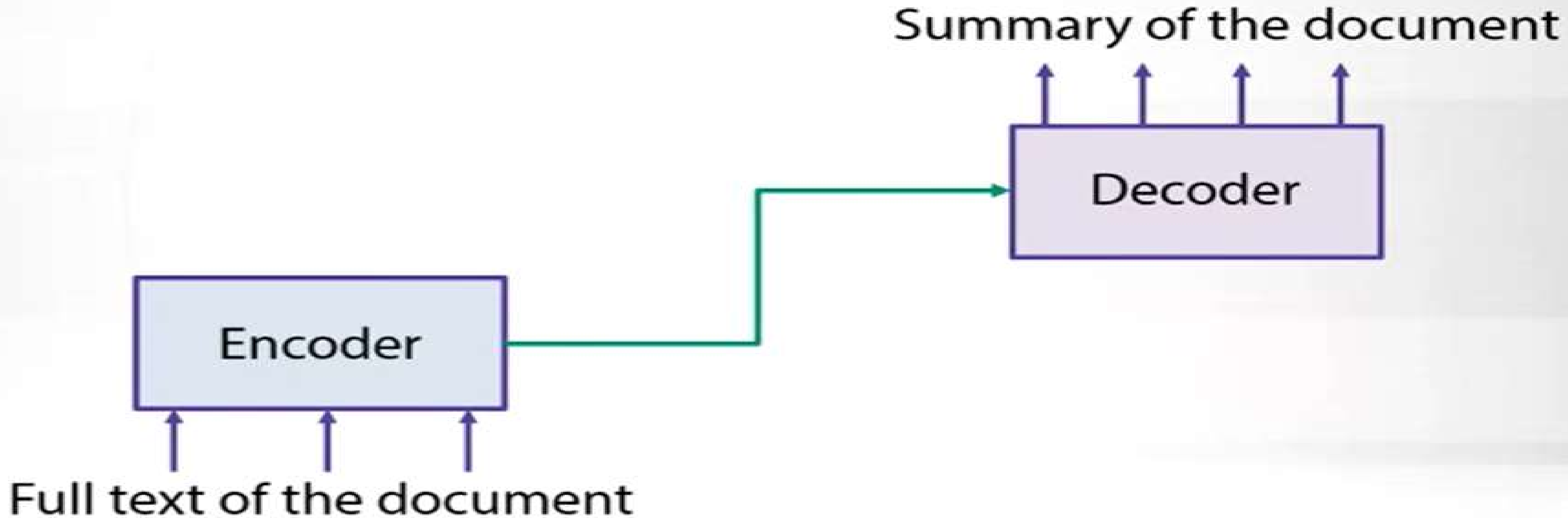
Alice and Bob took the train to visit the zoo. They saw a baby giraffe, a lion, and a flock of colorful tropical birds.

Extractive Summary

Alice and Bob visit the zoo. saw a flock of birds.

Abstractive summary

Alice and Bob visited the zoo and saw animals and birds.



From Google research blog

Dataset: Annotated English Gigaword – 10 mln. documents
catalog.ldc.upenn.edu/LDC2012T21

Model: sequence to sequence with attention + beam search

Code: open-source TF implementation
github.com/tensorflow/models/tree/master/research/textsum

Results?

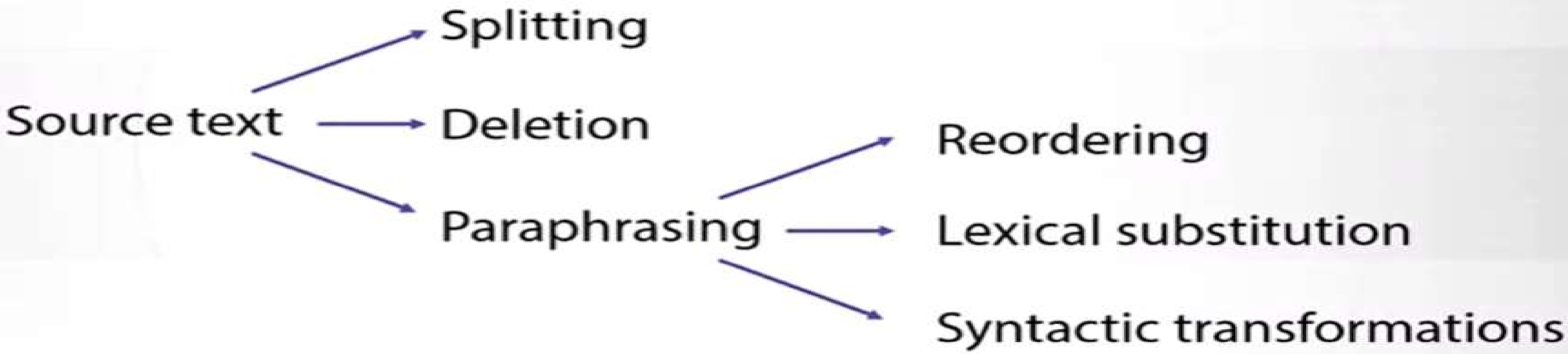
Input: Article 1st sentence	Model-written headline
metro-goldwyn-mayer reported a third-quarter net loss of dlrs 16 million due mainly to the effect of accounting rules adopted this year	*mgm reports 16 million net loss on higher revenue*
starting from july 1, the island province of hainan in southern china will implement strict market access control on all incoming livestock and animal products to prevent the possible spread of epidemic diseases	*hainan to curb spread of diseases*

Simplification

Text simplification – reducing the lexical and syntactical complexity of text.

a.	**Normal:** As Isolde arrives at his side, Tristan dies with her name on his lips. **Simple:** As Isolde arrives at his side, Tristan dies while speaking her name.
b.	**Normal:** Alfonso Perez Munoz, usually referred to as Alfonso, is a former Spanish footballer, in the striker position. **Simple:** Alfonso Perez is a former Spanish football player.
c.	**Normal:** Endemic types or species are especially likely to develop on islands because of their geographical isolation. **Simple:** Endemic types are most likely to develop on islands because they are isolated.

Operations to simplify text



Rule-based approach for paraphrasing

Lexical	[RB]	solely	→	only
	[NN]	objective	→	goal
	[JJ]	undue	→	unnecessary
Phrasal	[VP]	accomplished	→	carried out
	[VP/PP]	make a significant contribution	→	contribute greatly
	[VP/S]	is generally acknowledged that	→	is widely accepted that
Syntactic	[NP/VP]	the manner in which NN	→	the way NN
	[NP]	NNP 's population	→	the people of NNP
	[NP]	NNP 's JJ legislation	→	the JJ law of NNP

- Synchronous context-free grammar (SCFG) rules
- Uppercase indicates non-terminal symbols
- Paraphrase Database
  http://www.cis.upenn.edu/~ccb/ppdb/

Simplification

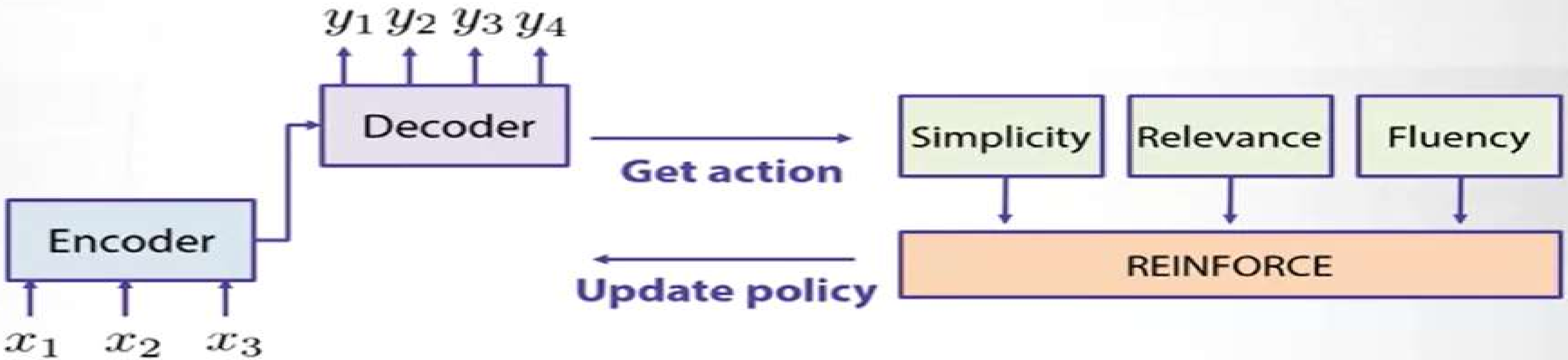
Encoder-decoder framework – yes, but the network might learn just to **copy** the content… How do we force it to **simplify**?

Reinforcement learning can be used to do **weak supervision.**

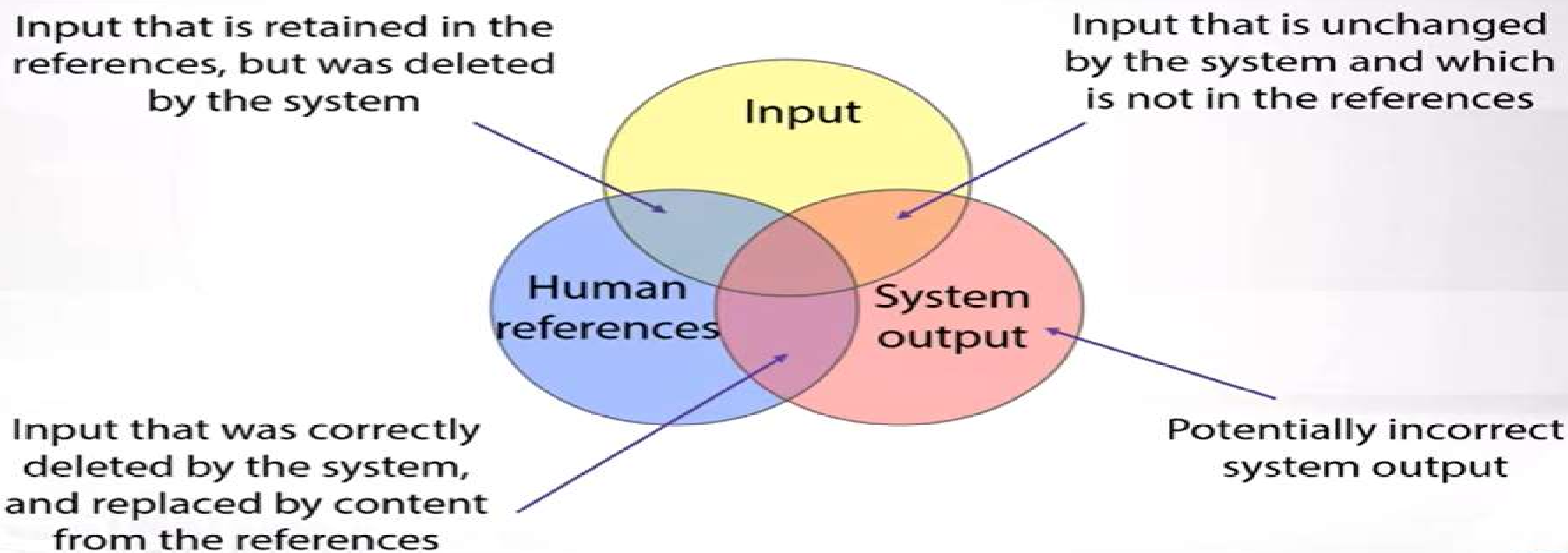
- **Action:** output next word y_j
- **Policy:** $p(y_j|\mathbf{x}, y_1, \ldots y_{j-1})$
- **Reward:** Adequacy + Fluency + Simplicity

Rewards come only when the whole sequence is generated.

Simplification



How to measure simplicity?



How to measure simplicity?

SARI (**s**ystem **a**gainst **r**eferences and **i**nput) – arithmetic average of n-gram precision and recall of

- addition
- copying
- deletion

For example, precision for **addition**:

$$\text{precision} = \frac{\sum_{g \in O} [g \in (O \cap \bar{I} \cap R)]}{\sum_{g \in O} [g \in (O \cap \bar{I})]}$$

SARI: example

INPUT: About 95 species are currently accepted.
REF-1: About 95 species are currently known.
*REF-2: About 95 species are **now** accepted.*
REF-3: 95 species are now accepted.

OUTPUT-1: About 95 you now get in. → 0.2683
*OUTPUT-2: About 95 species are **now** agreed.* → **0.7594**
OUTPUT-3: About 95 species are currently agreed. → 0.5890

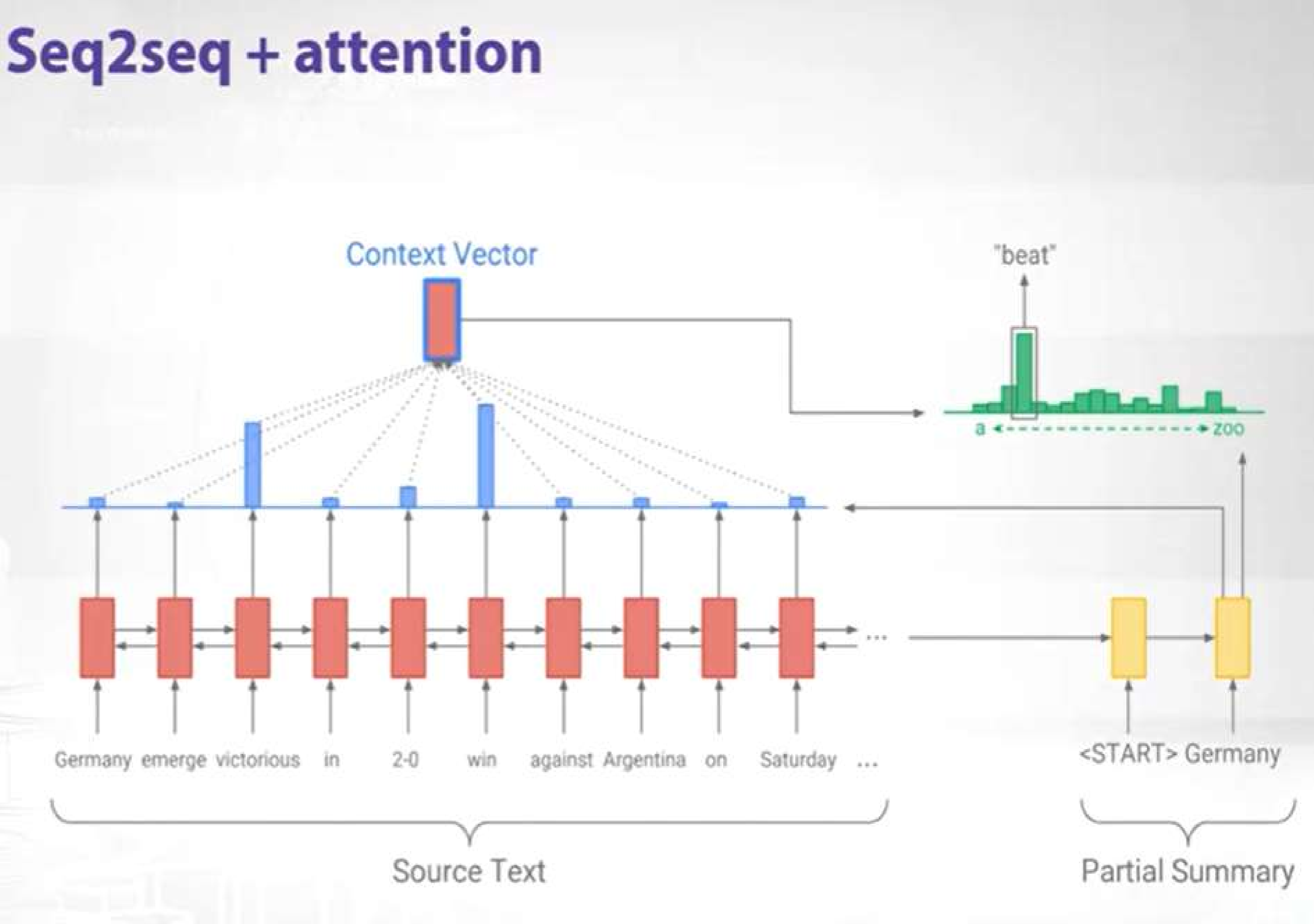
Compare with BLEU

INPUT: About 95 species are currently accepted.
REF-1: About 95 species are currently known.
*REF-2: About 95 species are **now** accepted.*
REF-3: 95 species are now accepted.

OUTPUT-1: About 95 you now get in. → 0.1562
*OUTPUT-2: About 95 species are **now** agreed.* → **0.6435**
OUTPUT-3: About 95 species are currently agreed. → **0.6435**

BLEU does not distinguish between outputs 2 and 3.

SUMMARIZATION WITH POINTER GENERATED NETWORKS:



Original Text (truncated): lagos, nigeria (cnn) a day after winning nigeria's presidency, *muhammadu buhari* told cnn's christiane amanpour that he plans to aggressively fight corruption that has long plagued nigeria and go after the root of the nation's unrest. buhari said he'll "rapidly give attention" to curbing violence in the northeast part of nigeria, where the terrorist group boko haram operates. by cooperating with neighboring nations chad, cameroon and niger, he said his administration is confident it will be able to thwart criminals and others contributing to nigeria's instability. for the first time in nigeria's history, the opposition defeated the ruling party in democratic elections. buhari defeated incumbent goodluck jonathan by about 2 million votes, according to nigeria's independent national electoral commission. the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.

Seq2Seq + Attention: UNK UNK says his administration is confident it will be able to **destabilize nigeria's economy.** UNK says his administration is confident it will be able to thwart criminals and other **nigerians. he says the country has long nigeria and nigeria's economy.**

Closer look into formulas

1. **Attention distribution (over source positions):**

$$e_i^j = w^T \tanh(W_h h_i + W_s s_j + b_{attn})$$

$$p^j = softmax(e^j)$$

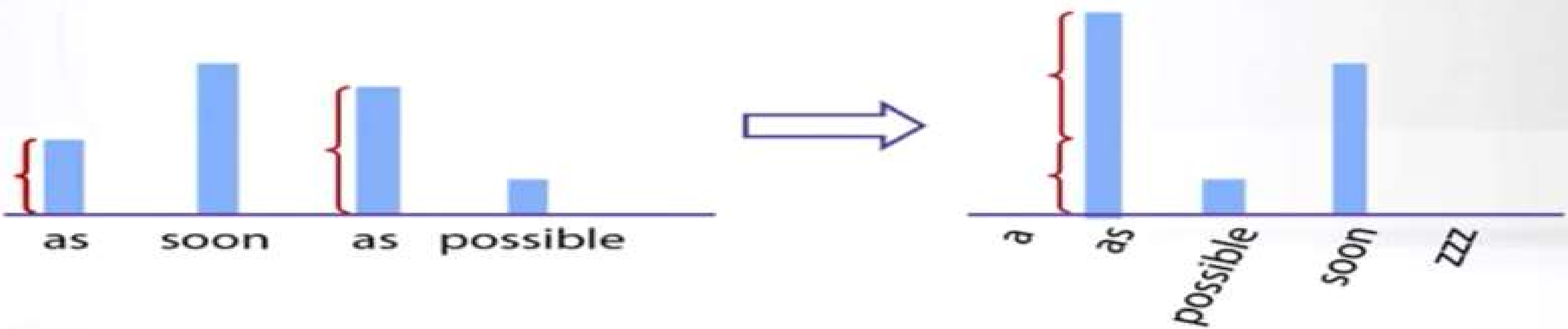
2. **Vocabulary distribution (generative model):**

$$v_j = \sum_i p_i^j h_i$$

$$p_{vocab} = softmax(V'(V[s_j, v_j] + b) + b')$$

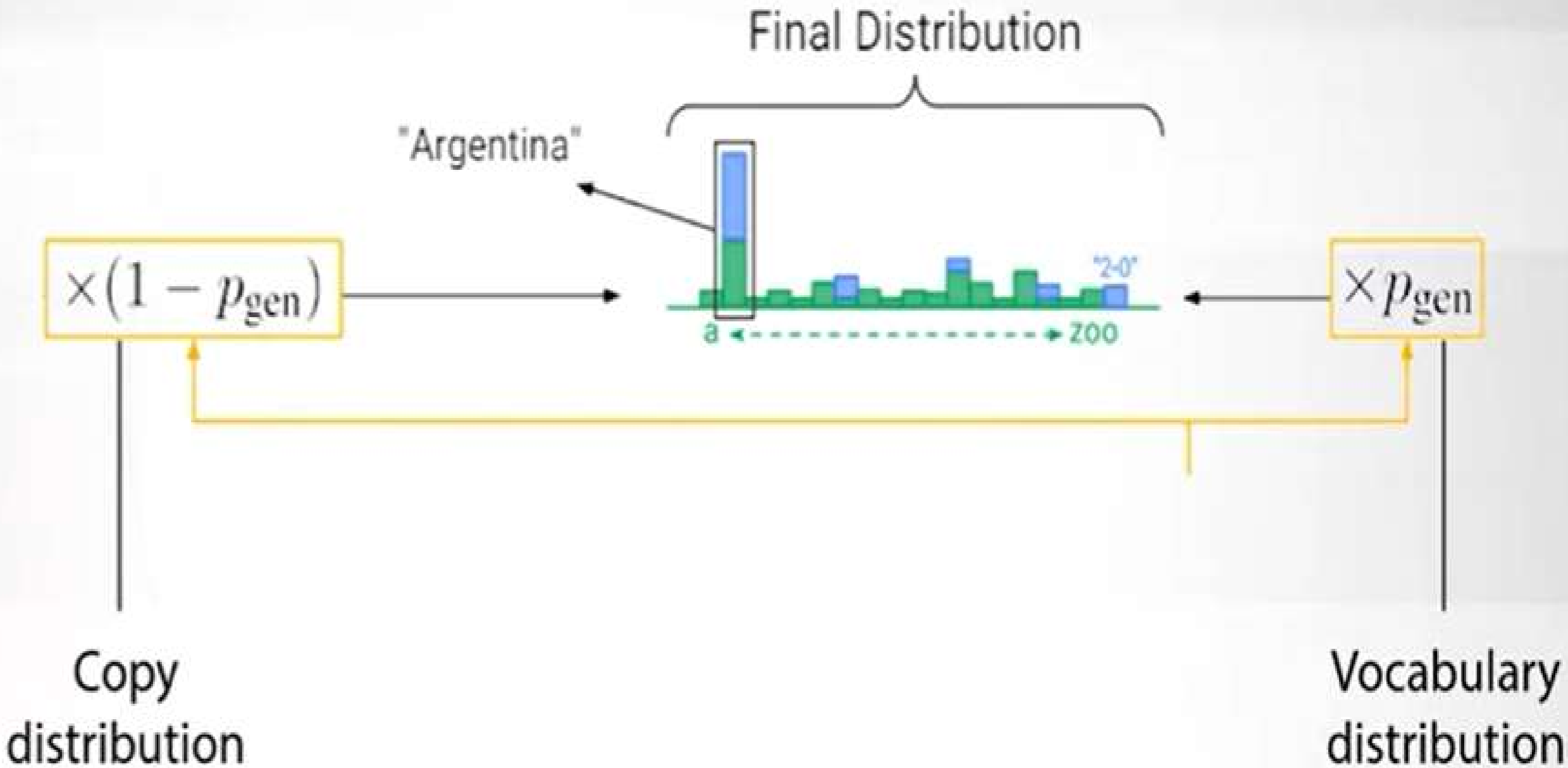
3. **Copy distribution (over words from source):**

$$p_{copy}(w) = \sum_{i:\, x_i = w} p_i^j$$



Attention distribution *Copy distribution*

Pointer-generator network



Closer look into formulas

4. Final distribution:

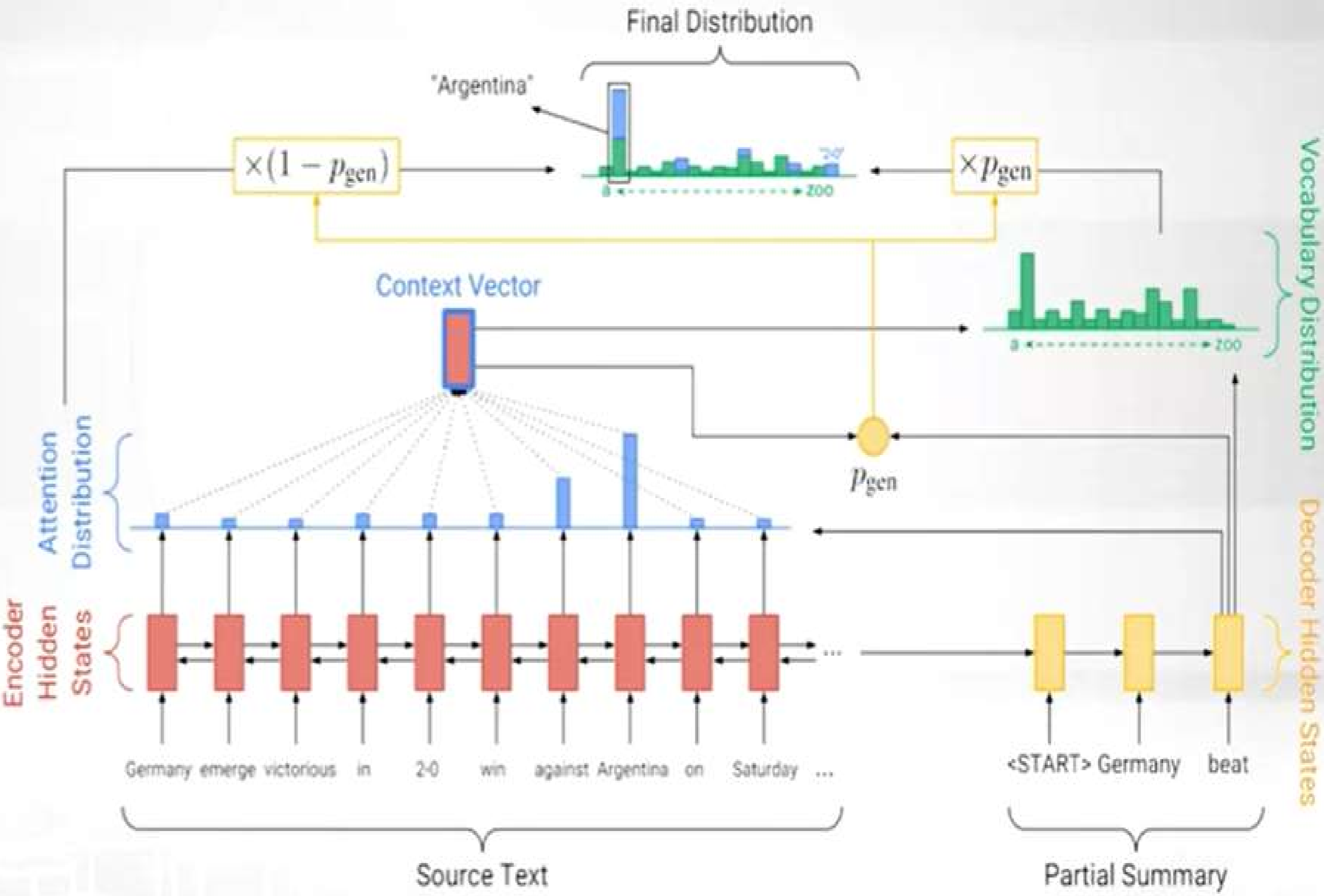
$$p_{final} = p_{gen}\, p_{vocab} + (1 - p_{gen}) p_{copy}$$

$$p_{gen} = \sigma(w_v^T v_j + w_s^T s_j + w_x^T y_{j-1} + b_{gen})$$

5. Training:

$$\text{Loss} = -\frac{1}{J} \sum_{j=1}^{J} \log p_{final}(y_j)$$

Pointer-generator network



Coverage mechanism

Coverage vector:

$$c^j = \sum_{j'=0}^{j-1} p^{j'}$$

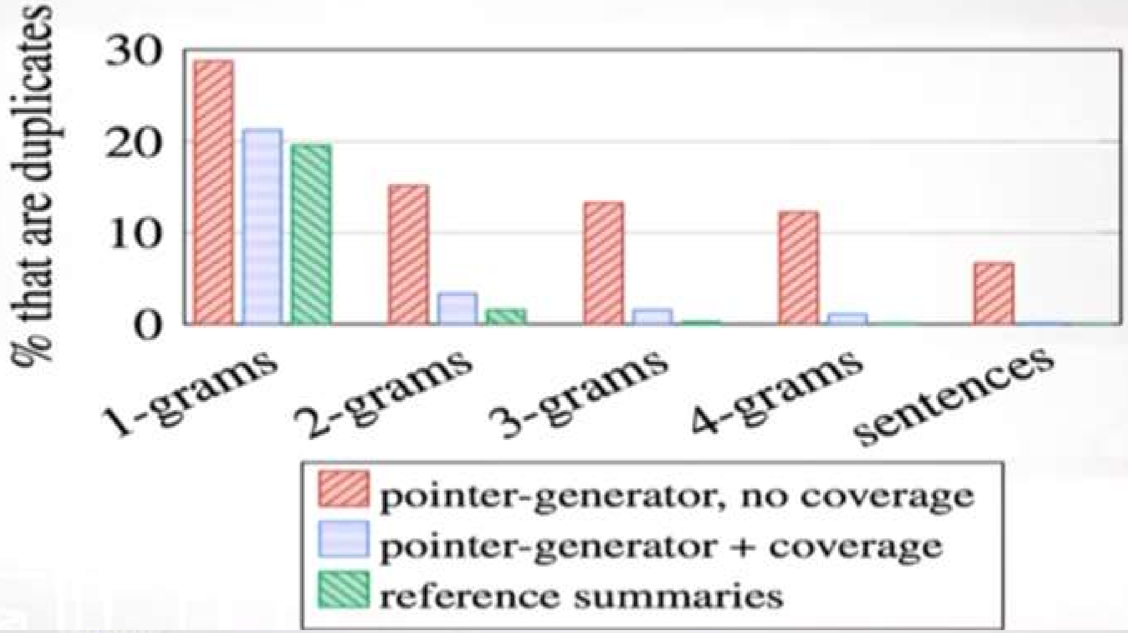
Modified attention:

$$e_i^j = w^T \tanh(W_h h_i + W_s s_j + w_c c_i^j + b_{attn})$$

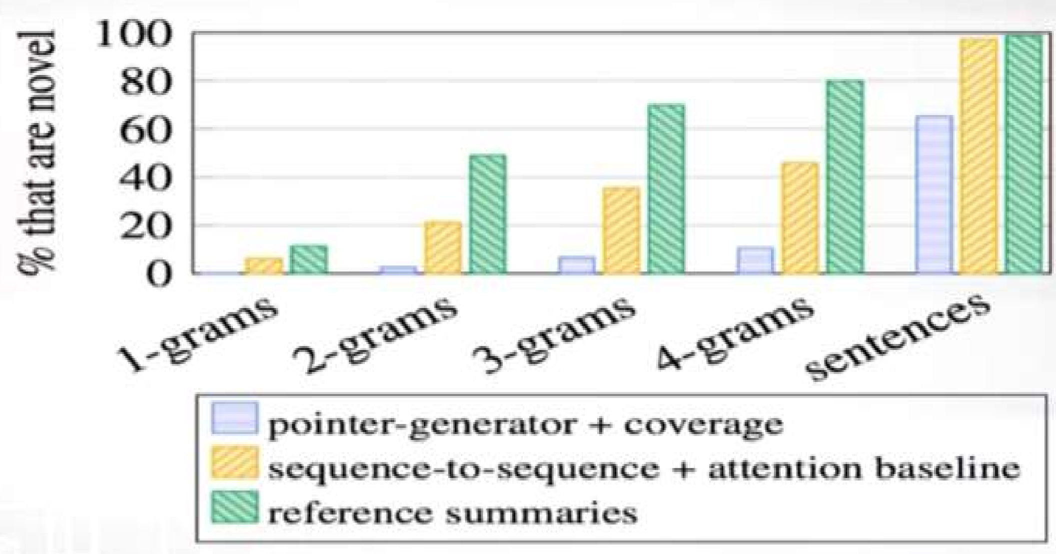
Coverage loss:

$$\text{covloss}_j = \sum_i \min(p_i^j, c_i^j)$$

Model avoids repetitions



But becomes too extractive



Comparison of the models

	ROUGE score		
	1	2	L
abstractive model (Nallapati et al., 2016)	35.46	13.30	32.65
extractive model (Nallapati et al., 2017)	39.6	16.2	35.3
lead-3 baseline	40.34	17.70	36.57
seq2seq + attention	31.33	11.81	28.83
pointer-generator	36.44	15.66	33.42
pointer-generator + coverage	**39.53**	**17.28**	**36.38**