Lecture 4

Корбут Даниил
Deep Learning Research Engineer, Insilico Medicine

telegram: @rtriangle
vk: rtriangle
email: korbut.daniel@gmail.com

План занятия

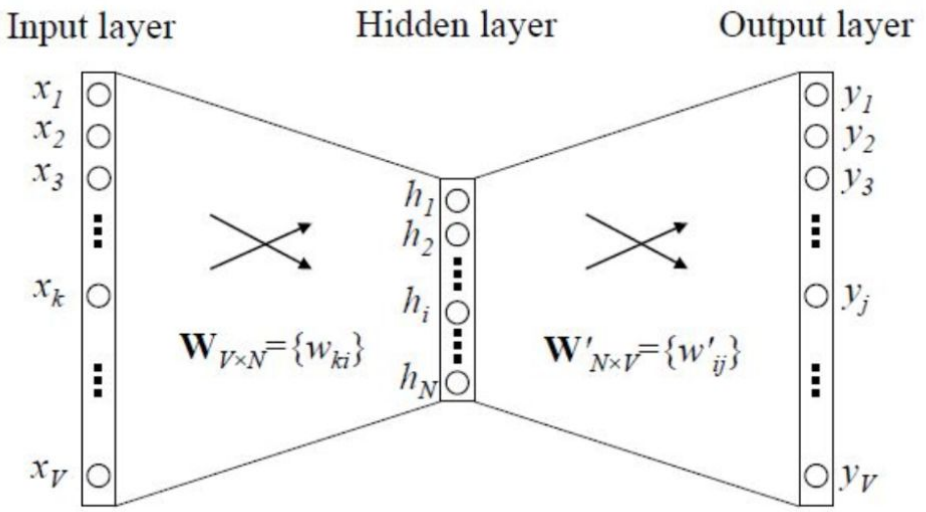
- Напоминание с прошлой лекции
- Более продвинутые техники NLP
- Популярные задачи NLP и методы их решения

Следующее занятие (последнее)

- Временные ряды и NN
- Рекомендательные системы на основе NN
- Кластеризация и методы понижения размерности
- Ваши пожелания!

Напоминание: word embeddings

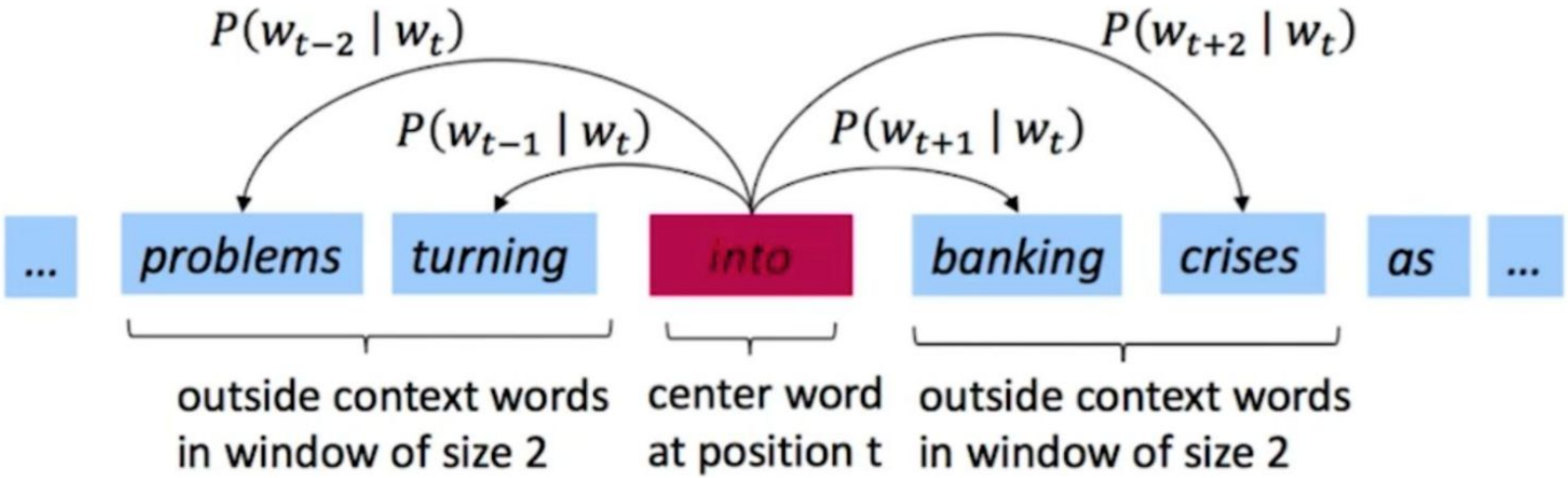
Word2Vec



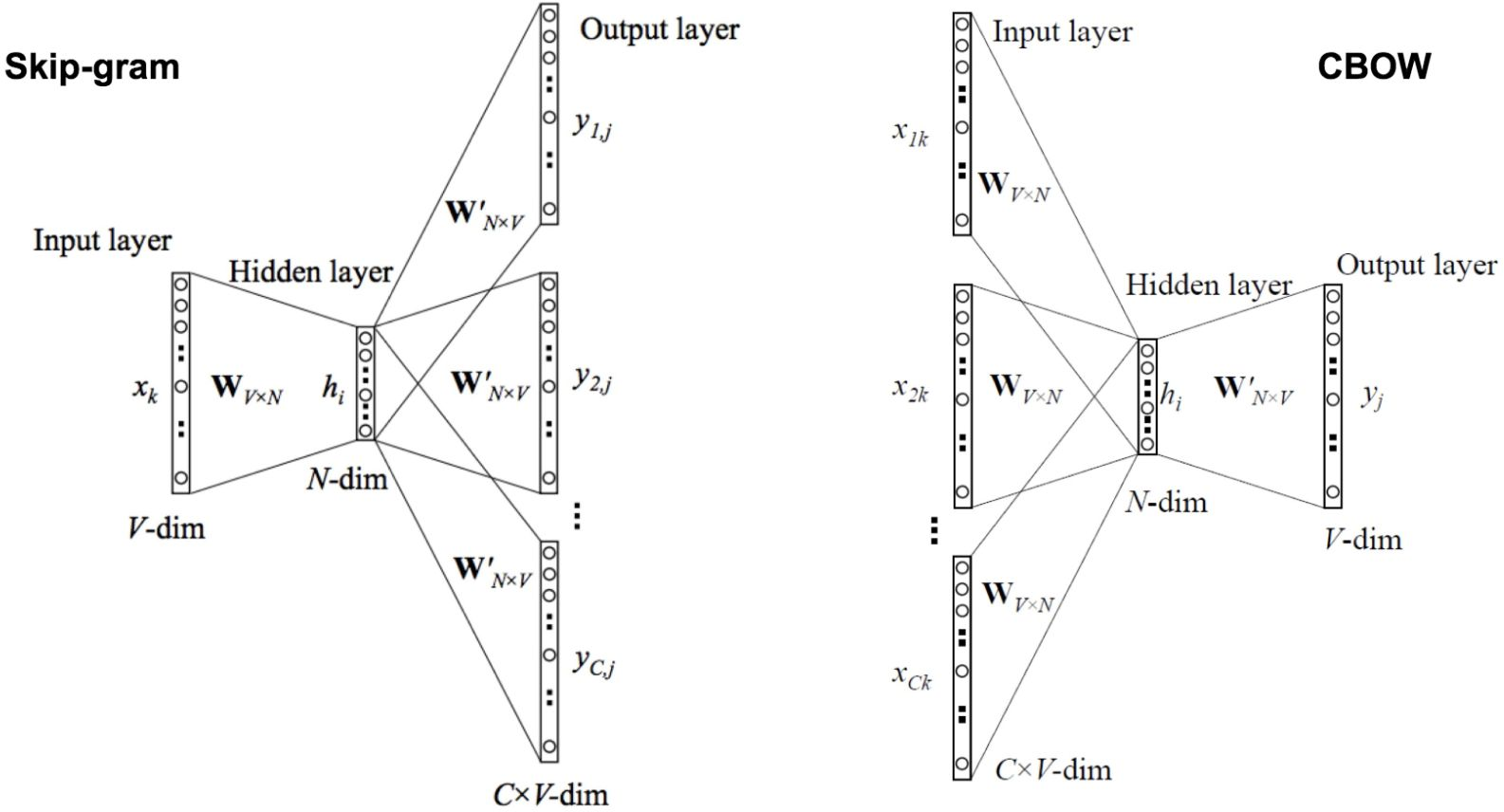
- ➔ Predict context words given a word (or vice versa)
- ➔ maximize probability of seeing word and its context together

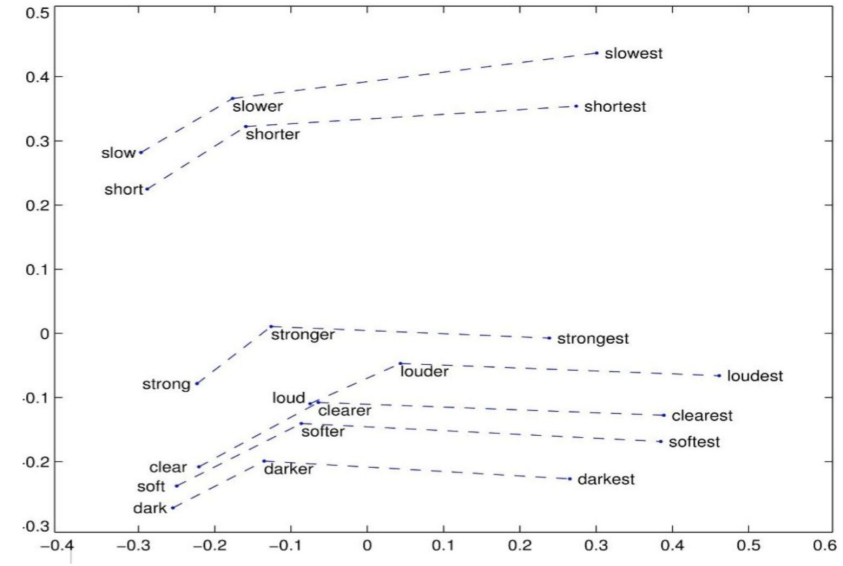
Word2Vec

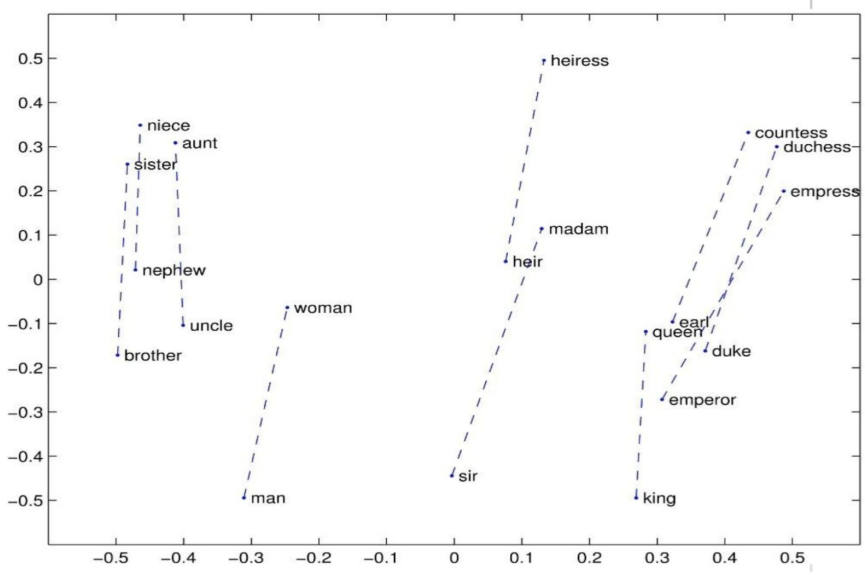
Going through text corpus by sliding windows

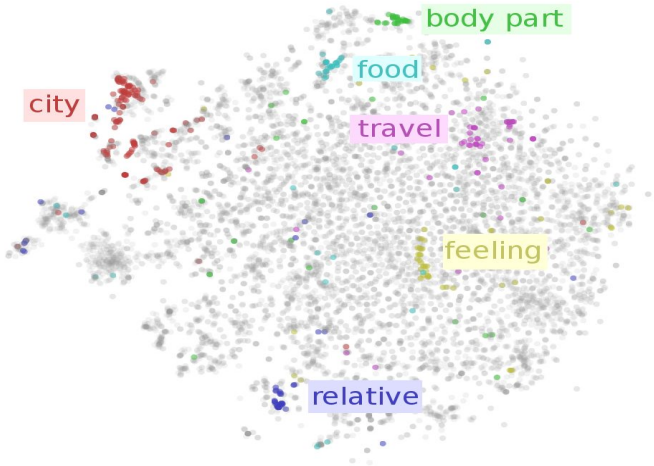


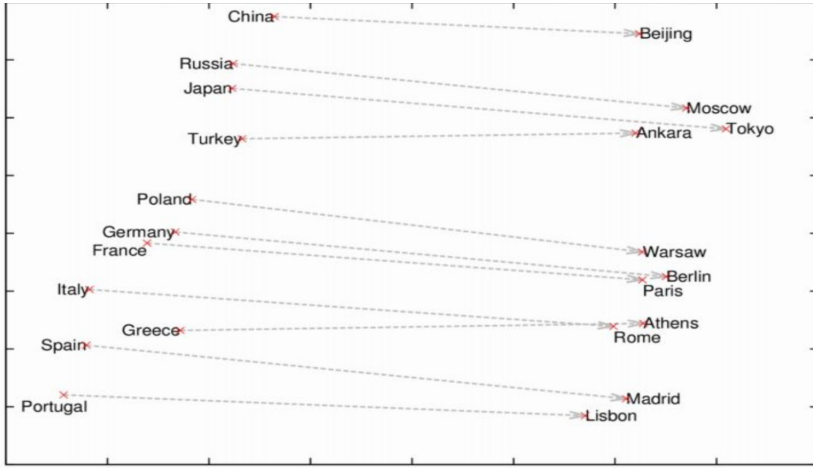
Word2Vec


Skip-gram
Input layer
Hidden layer
Output layer
x_k
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
$y_{1,j}$
$y_{2,j}$
$y_{C,j}$
V-dim
N-dim
$C\times V$-dim
CBOW
Input layer
Hidden layer
Output layer
x_{1k}
x_{2k}
x_{Ck}
$\mathbf{W}_{V\times N}$
h_i
$\mathbf{W}'_{N\times V}$
y_j
N-dim
V-dim
$C\times V$-dim


slow
short
slower
shorter
slowest
shortest
strong
stronger
strongest
loud
louder
loudest
clear
clearer
clearest
soft
softer
softest
dark
darker
darkest

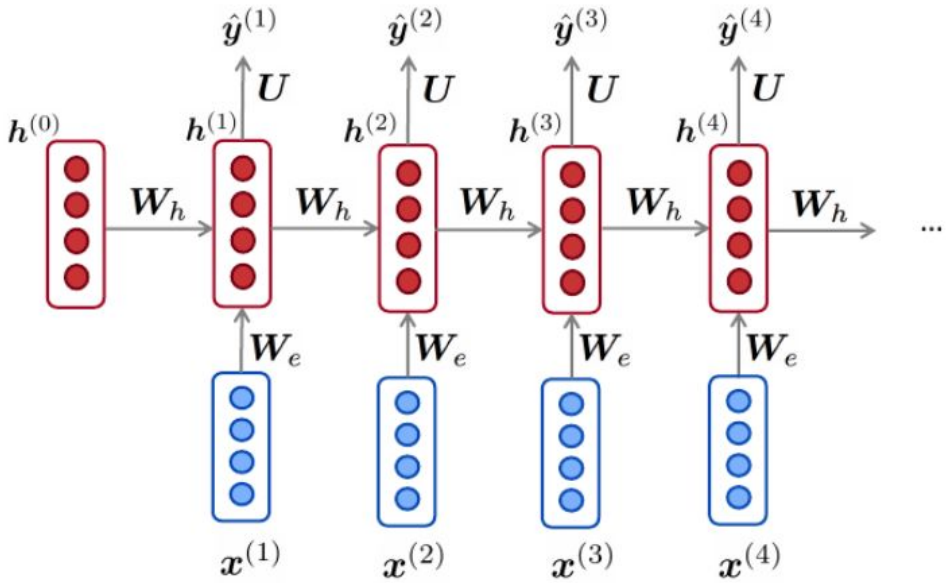

niece
aunt
sister
nephew
uncle
brother
woman
man
heiress
madam
heir
sir
countess
duchess
empress
earl
queen
duke
emperor
king


body part
food
city
travel
feeling
relative

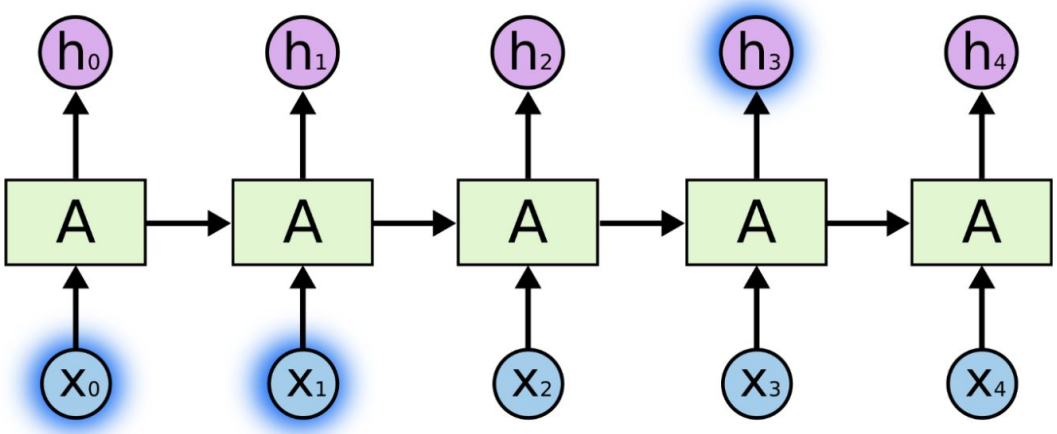

China
Beijing
Russia
Japan
Moscow
Tokyo
Turkey
Ankara
Poland
Warsaw
Germany
France
Berlin
Paris
Italy
Athens
Greece
Rome
Spain
Madrid
Portugal
Lisbon

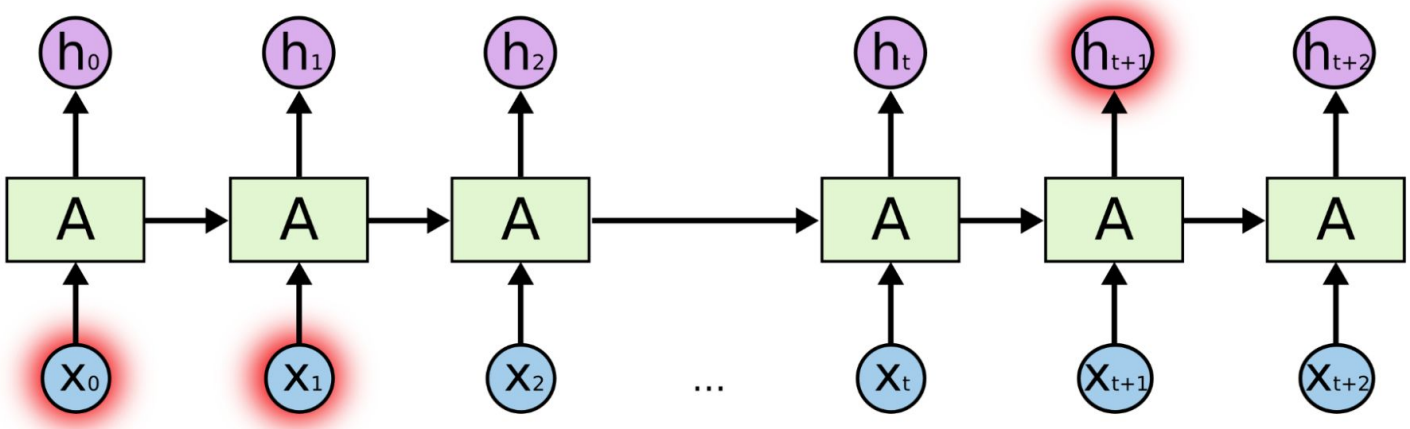
Recurrent neural networks (RNNs)

$$
\begin{aligned}
h_t &= W f(h_{t-1}) + W^{(hx)} x_{[t]} \\
\hat{y}_t &= W^{(S)} f(h_t)
\end{aligned}
$$

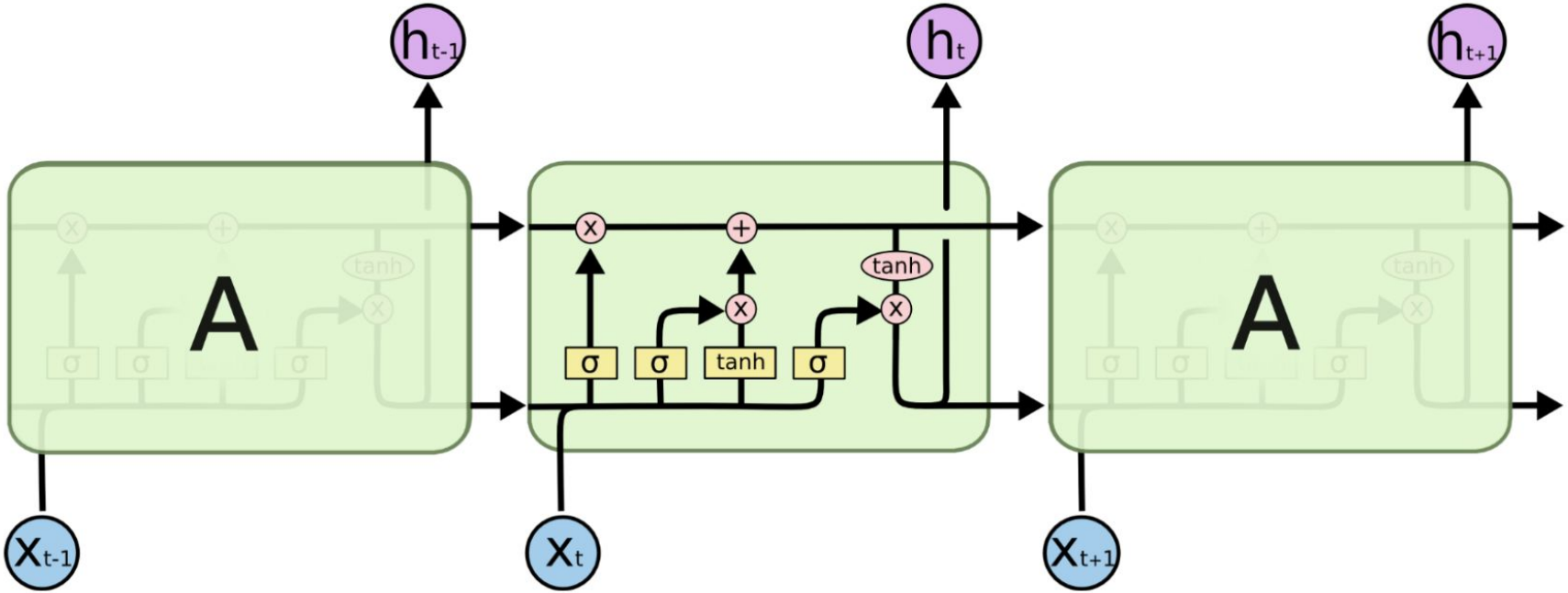


problems


h0
h1
h2
h3
h4
A
A
A
A
A
x0
x1
x2
x3
x4


h0
h1
h2
ht
ht+1
ht+2
A
A
A
A
A
A
x0
x1
x2
...
xt
xt+1
xt+2

LSTM

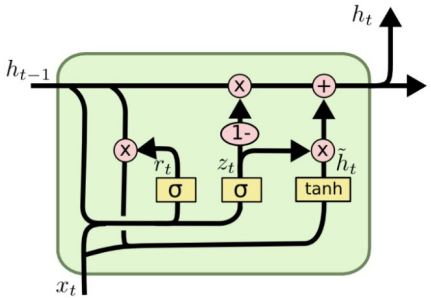


GRU

Let's make LSTM little bit simpler by removing it's state. It's role will be performed by previous output.

$$z = \sigma(W_z \cdot (x_t, h_{t-1})),\ z \in [0,1]^q$$
$$r = \sigma(W_r \cdot (x_t, h_{t-1})),\ r \in [0,1]^q$$
$$\tilde{h_t} = tanh(W \cdot (x_t, r \cdot h_{t-1}))$$
$$h_t = (1-z) \cdot h_{t-1} + z * \tilde{h_t}$$

$$z = \sigma(W_z \cdot (x_t, h_{t-1})),\ z \in [0,1]^q$$
$$r = \sigma(W_r \cdot (x_t, h_{t-1})),\ r \in [0,1]^q$$
$$\tilde{h_t} = tanh(W \cdot (x_t, r \cdot h_{t-1}))$$
$$h_t = (1-z) \cdot h_{t-1} + z * \tilde{h_t}$$



Sequence labeling

Problem

- Given a sequence of *tokens*, infer the most probable sequence of *labels* for these tokens.

Examples

- part of speech tagging
- named entity recognition

Example: part-of-speech tagging

Tokens are words, and labels are PoS tags.

PRON	VERB	DET	PROPN	NOUN
I	saw	a	Heffalump	today.

https://www.coursera.org/learn/language-processing

Example: named entity recognition

Once upon a time, a very long time ago now, about **[last Friday]**, **[Winnie-the-Pooh]** lived in a forest all by himself under the name of **[Sanders]**.

Types of named entities

Any real-world object which may have a proper name:

- persons
- organizations
- locations
- …

Also named entities usually include:

- dates and times
- units
- amounts

Notation:

$\mathbf{x} = x_1, \ldots x_T$ is a sequence of words (input)

$\mathbf{y} = y_1, \ldots y_T$ is a sequence of their tags (labels)

We need to find the most probable sequence of tags given the sentence:

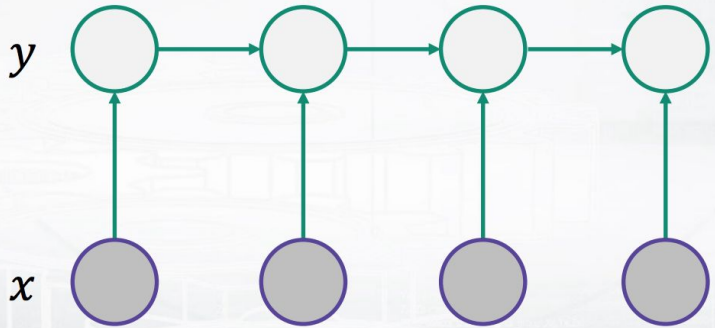
$$\mathbf{y} = \text{argmax}_{\mathbf{y}}\, p(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\mathbf{y}}\, p(\mathbf{x}, \mathbf{y})$$

https://www.coursera.org/learn/language-processing

Maximum Entropy Markov Model

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} p(y_t|y_{t-1}, x_t)$$

Discriminative model



https://www.coursera.org/learn/language-processing

Maximum Entropy Markov Model

$$p(y_t|y_{t-1}, x_t) = \frac{1}{Z_t(y_{t-1}, x_t)} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t)\right)$$

Normalization constant

weight

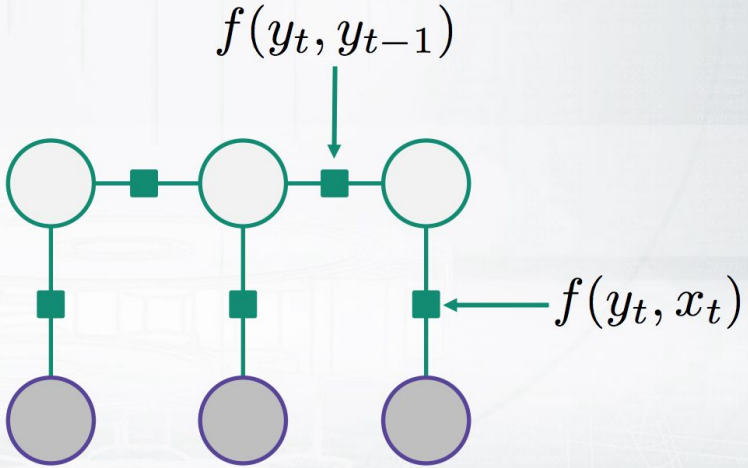
feature

https://www.coursera.org/learn/language-processing

Conditional Random Field (linear chain)

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(x)} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \right)$$

Undirected graph:



https://www.coursera.org/learn/language-processing

Features engineering

Label-observation features:

- $f(y_t, y_{t-1}, x_t) = [y_t = y]\, g_m(x_t)$
- $f(y_t, y_{t-1}, x_t) = [y_t = y][y_{t-1} = y']$
- $f(y_t, y_{t-1}, x_t) = [y_t = y][y_{t-1} = y']\, g_m(x_t)$

https://www.coursera.org/learn/language-processing

	$w_t = v$	$\forall v \in$
	part-of-speech tag for w_t is j	$\forall$ tags j
	w_t is in a phrase of syntactic type j	$\forall$ tags j
Capitalized	w_t matches [A-Z][a-z]+	
AllCaps	w_t matches [A-Z]+	
EndsInDot	w_t matches [^\.]+.*\.	
	w_t matches a dash	
	w_t appears in a list of stop words	
	w_t appears in list of capitals	

https://www.coursera.org/learn/language-processing

Black-box implementations

CRF++	https://sourceforge.net/projects/crfpp/
MALLET	http://mallet.cs.umass.edu/
GRMM	http://mallet.cs.umass.edu/grmm/
CRFSuite	http://www.chokkan.org/software/crfsuite/
FACTORIE	http://www.factorie.cc

Sequence tagging tasks

- Part-of-Speech tagging
- Named Entity Recognition
- Semantic Role Labelling
- …

BIO-notation:

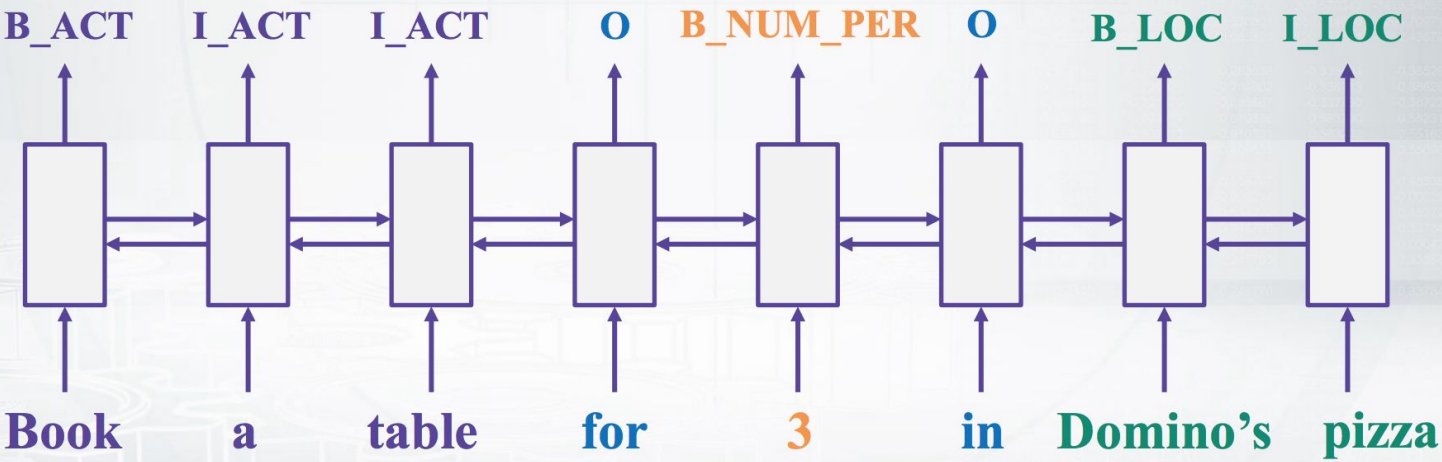
- B – beginning, I – inside, O – outside

B_ACT	I_ACT	I_ACT	O	B_NUM_PER	O	B_LOC	I_LOC
Book	a	table	for	3	in	Domino's	pizza

https://www.coursera.org/learn/language-processing

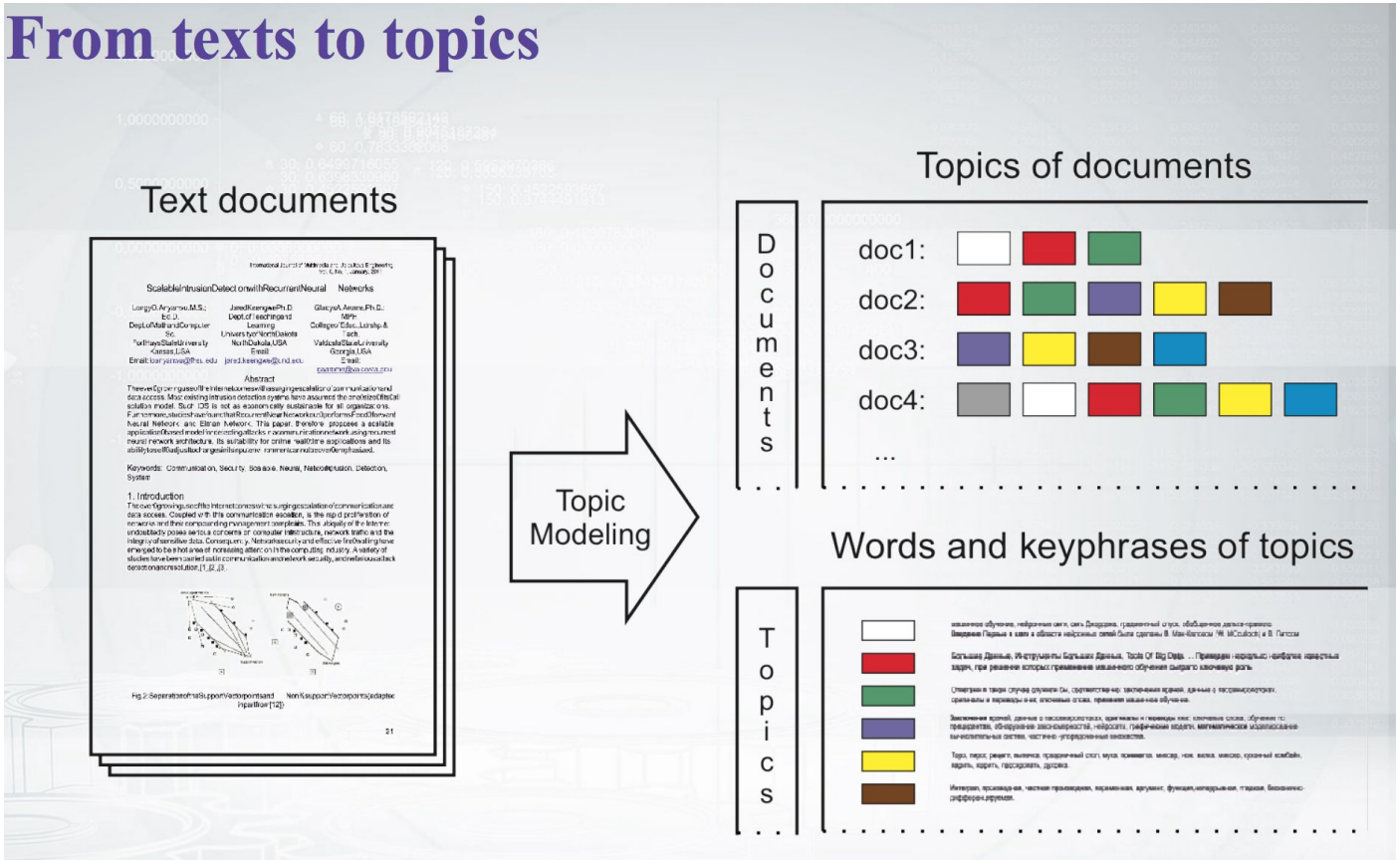
Bi-directional LSTM

- Universal approach for sequence tagging
- You can stack several layers + add linear layers on top
- Trained by cross-entropy loss coming from each position



https://www.coursera.org/learn/language-processing

Topic modeling and PLSA



https://www.coursera.org/learn/language-processing

The formal task

Given:

- Collection of texts as bags-of-words:

  n_{wd} is a count of the word w in the document d

Find:

- Probabilities of word in topics:

  $\phi_{wt} = p(w|t)$ ⟵ **Definition of a topic!**

- Probabilities of topics in documents:

  $\theta_{td} = p(t|d)$

https://www.coursera.org/learn/language-processing

Generative model of texts

Probabilistic Latent Semantic Analysis (PLSA):

$$p(w|d) = \sum_{t \in T} p(w|t,d)\, p(t|d) = \sum_{t \in T} p(w|t)\, p(t|d)$$

Law of total probability

$$p(w) = \sum_{t \in T} p(w|t)\, p(t)$$

Assumption of conditional independence

$$p(w|t,d) = p(w|t)$$

Notation:

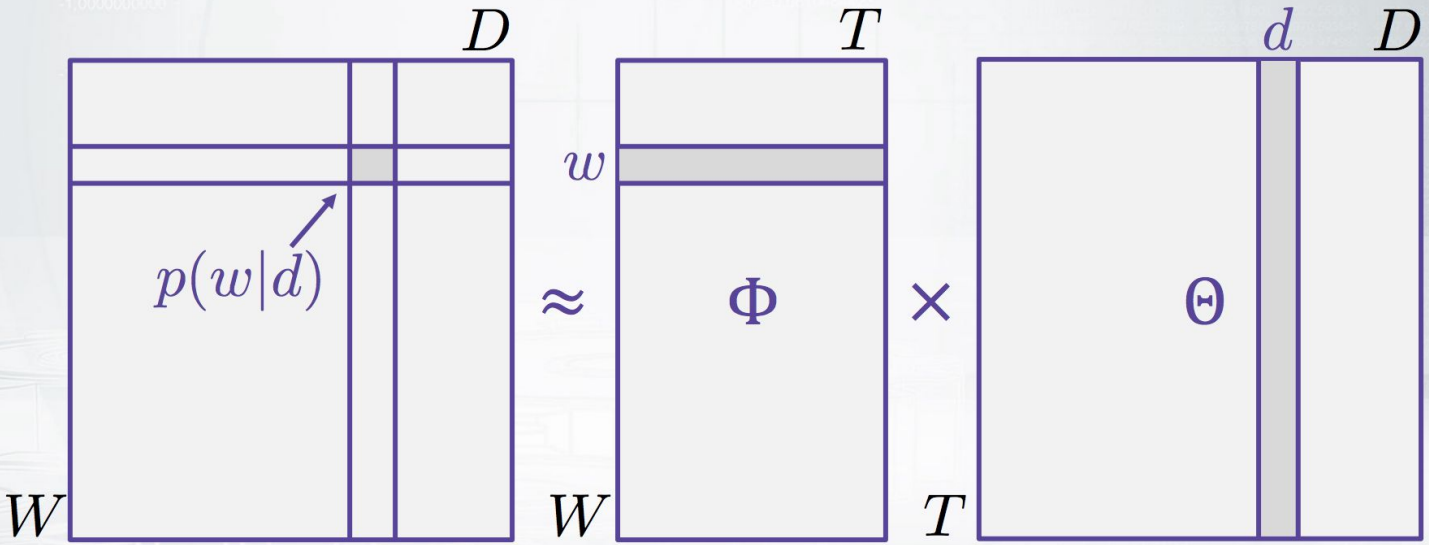
- w – *word*
- d – *document*
- t – *topic*

https://www.coursera.org/learn/language-processing

Matrix way of thinking

Probabilistic Latent Semantic Analysis:

$$p(w|d) = \sum_{t \in T} p(w|t)\, p(t|d) = \sum_{t \in T} \phi_{wt} \theta_{td}$$



https://www.coursera.org/learn/language-processing

How would you train the model?

Log-likelihood optimization:

$$\log \prod_{d \in D} p(d) \prod_{w \in d} p(w|d)^{n_{dw}} \to \max_{\Phi, \Theta}$$

$$\Updownarrow$$

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \log \sum_{t \in T} \phi_{wt} \theta_{td} \to \max_{\Phi, \Theta}$$

Given non-negativity and normalization constraints:

$$\phi_{wt} \geq 0 \qquad \theta_{td} \geq 0$$

$$\sum_{w \in W} \phi_{wt} = 1 \qquad \sum_{t \in T} \theta_{td} = 1$$

https://www.coursera.org/learn/language-processing

Latent Dirichlet Allocation

Dirichlet priors for $\phi_t = (\phi_{wt})_{w \in W}$ and $\theta_d = (\theta_{td})_{t \in T}$:

$$Dir(\phi_t|\beta) = \frac{\Gamma(\beta_0)}{\prod_w \Gamma(\beta_w)} \prod_w \phi_{wt}^{\beta_w - 1} \quad \beta_0 = \sum_w \beta_w, \beta_t > 0$$

- **Inference:**
  - Variational Bayes
  - Gibbs Sampling

- **Output:**
  - Posterior probabilities for parameters (also Dirichlet!).

https://www.coursera.org/learn/language-processing

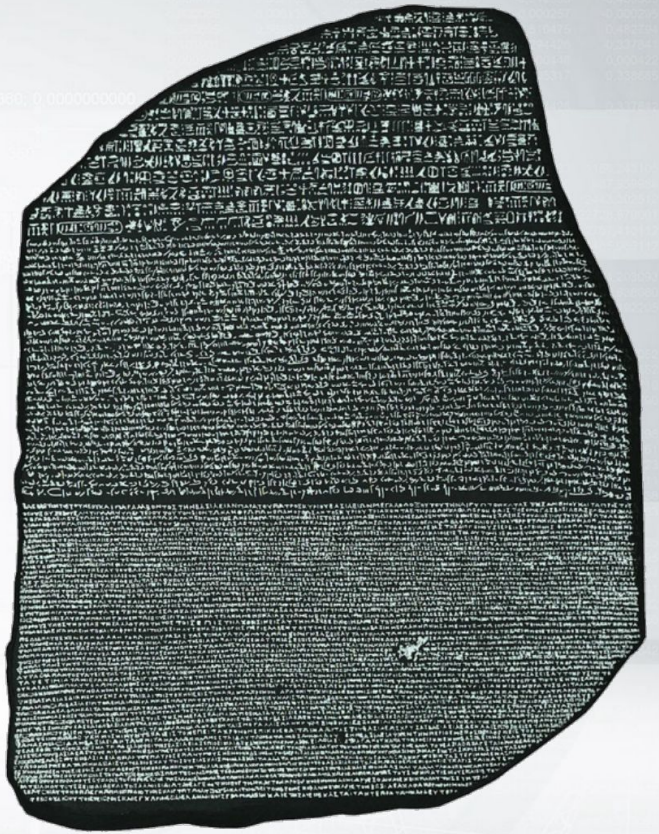
Machine Translation

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



https://www.coursera.org/learn/language-processing

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.***

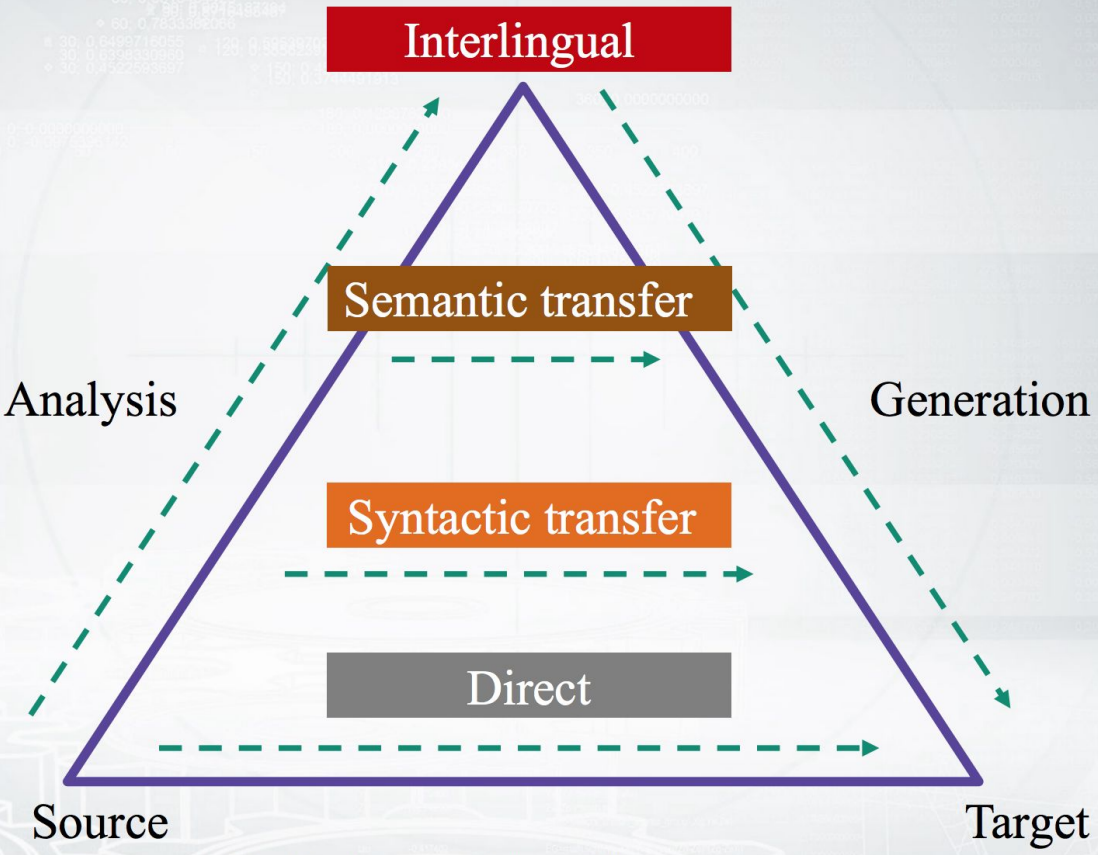
System output: ***The letter was sent on Tuesday.***

1-grams: 4 / 6
2-grams: 3 / 5
3-grams: 2 / 4
4-grams: 1 / 3
Brevity: min(1, 6/5)

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

https://www.coursera.org/learn/language-processing

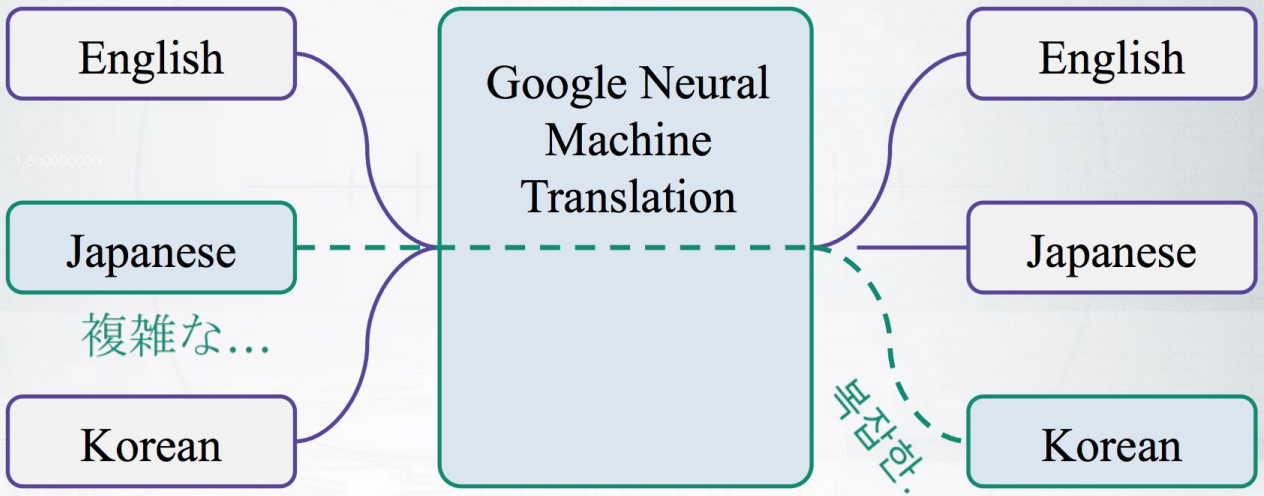
The mandatory slide



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

https://www.coursera.org/learn/language-processing

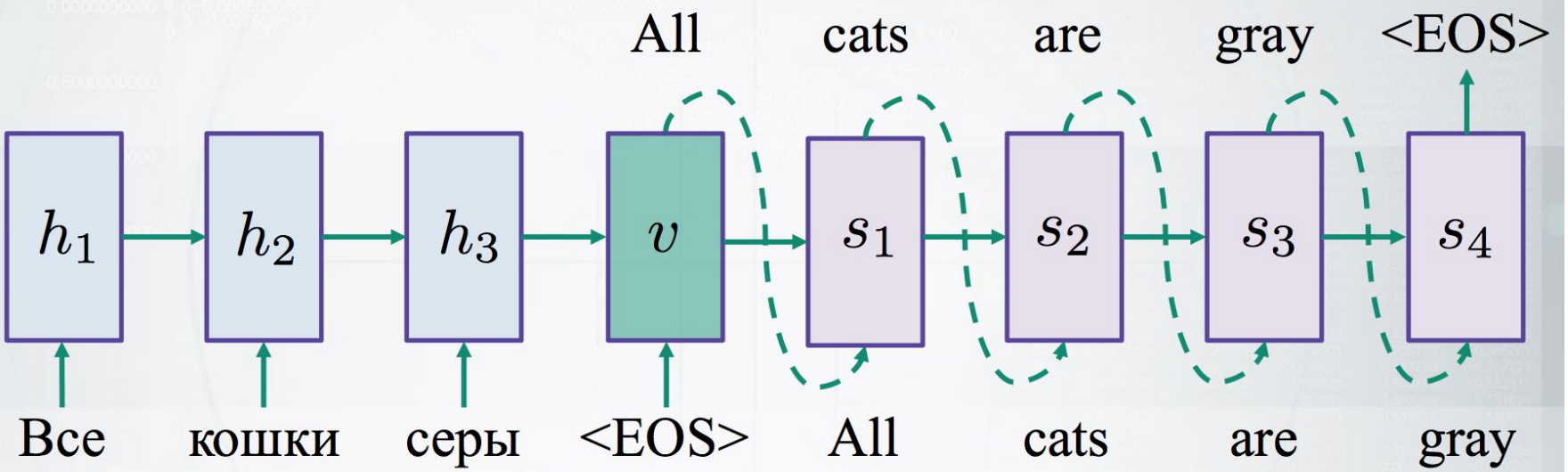
Zero-shot translation



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

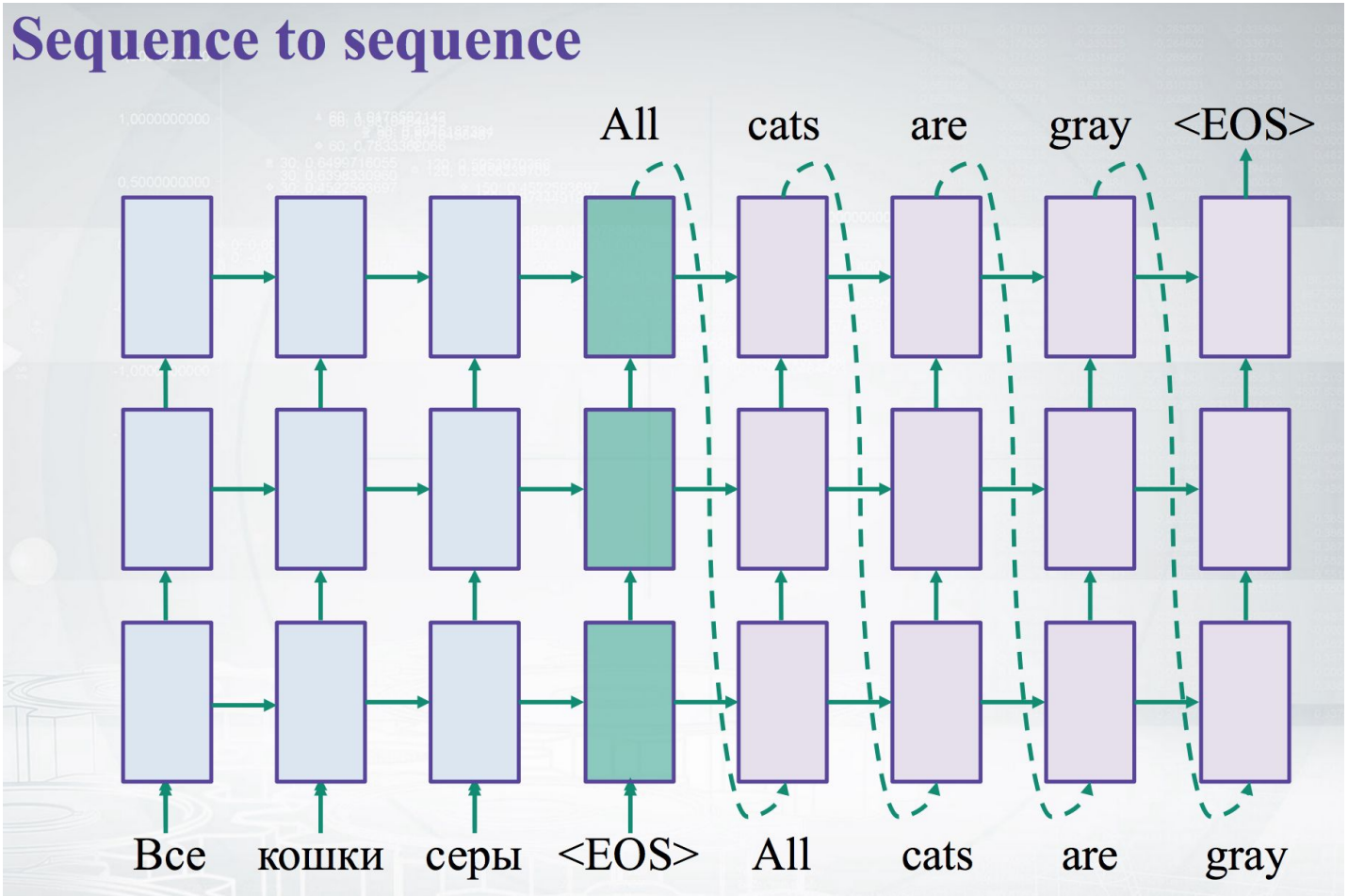
https://www.coursera.org/learn/language-processing

Sequence to sequence



https://www.coursera.org/learn/language-processing

Sequence to sequence



https://www.coursera.org/learn/language-processing

Sequence to sequence

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

- **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

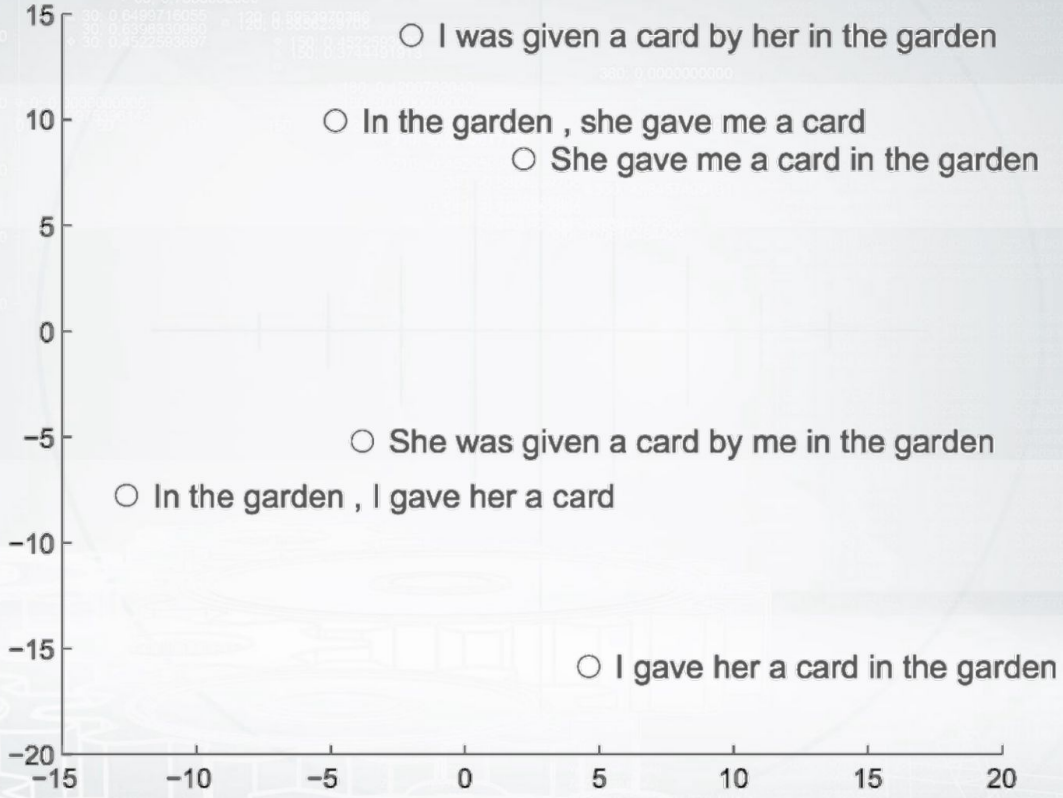
  RNN: $s_j = g(s_{j-1}, [y_{j-1}, v])$

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

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

https://www.coursera.org/learn/language-processing

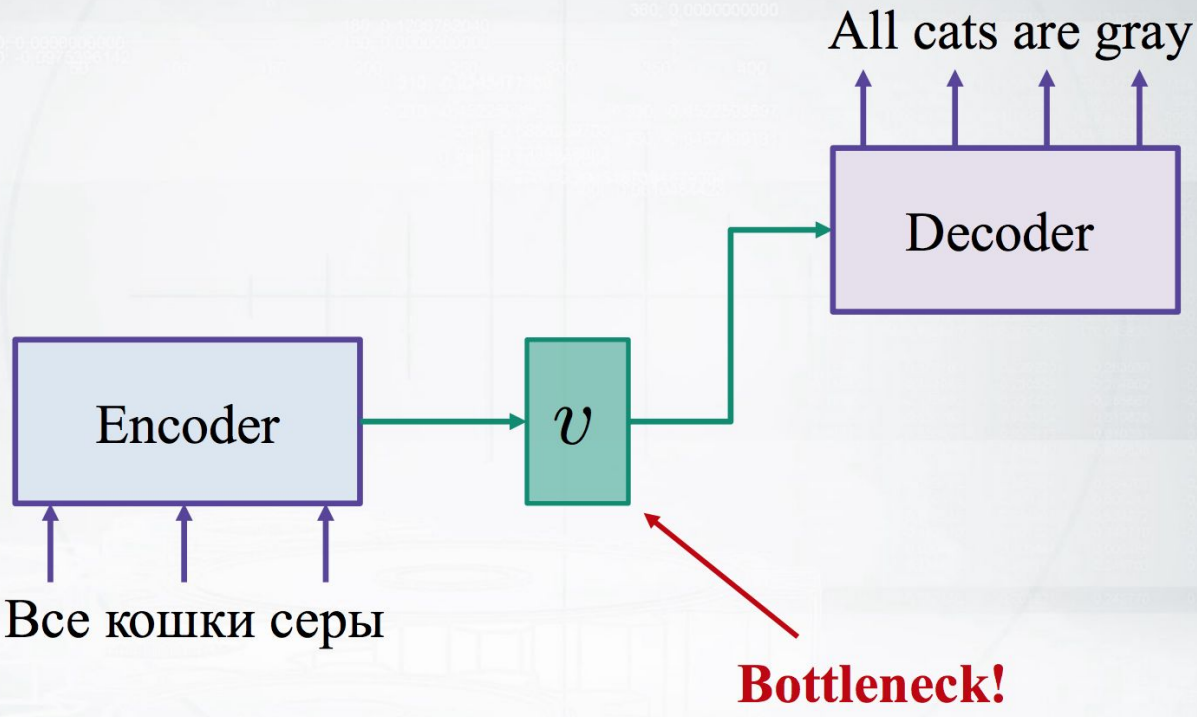
Hidden representations are good…



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

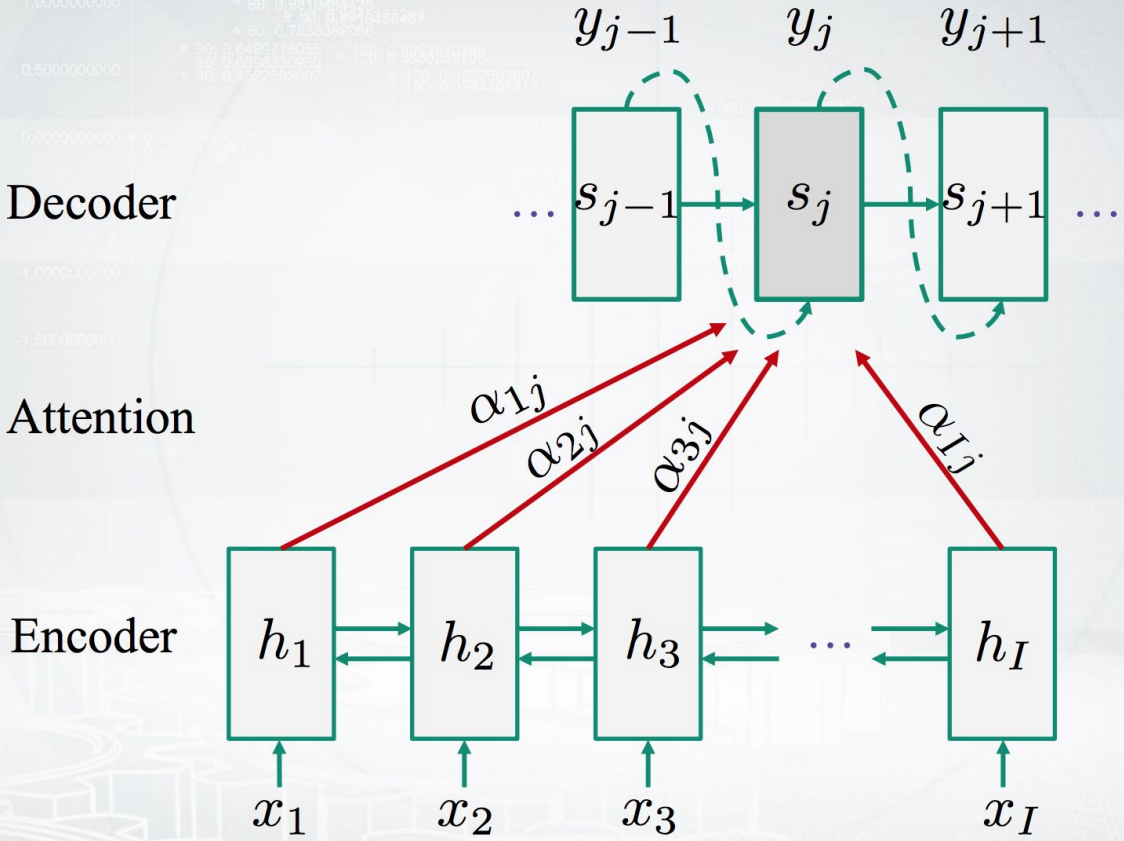
https://www.coursera.org/learn/language-processing

… but still a bottleneck



https://www.coursera.org/learn/language-processing

Attention mechanism



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

https://www.coursera.org/learn/language-processing

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:

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

https://www.coursera.org/learn/language-processing

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:**

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

https://www.coursera.org/learn/language-processing

Put all together

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

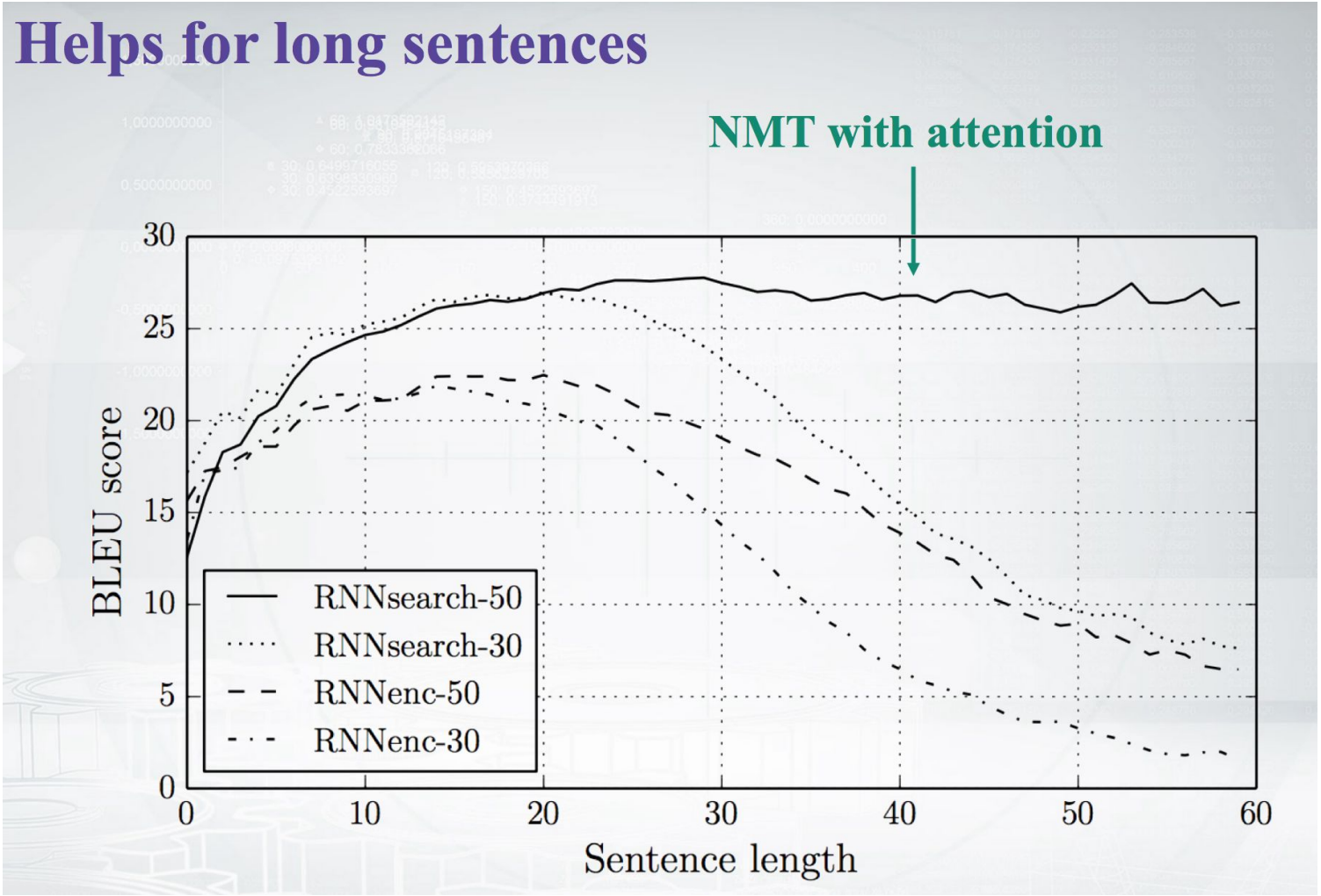
- Still encoder-decoder architecture with RNNs:

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

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

https://www.coursera.org/learn/language-processing

Helps for long sentences



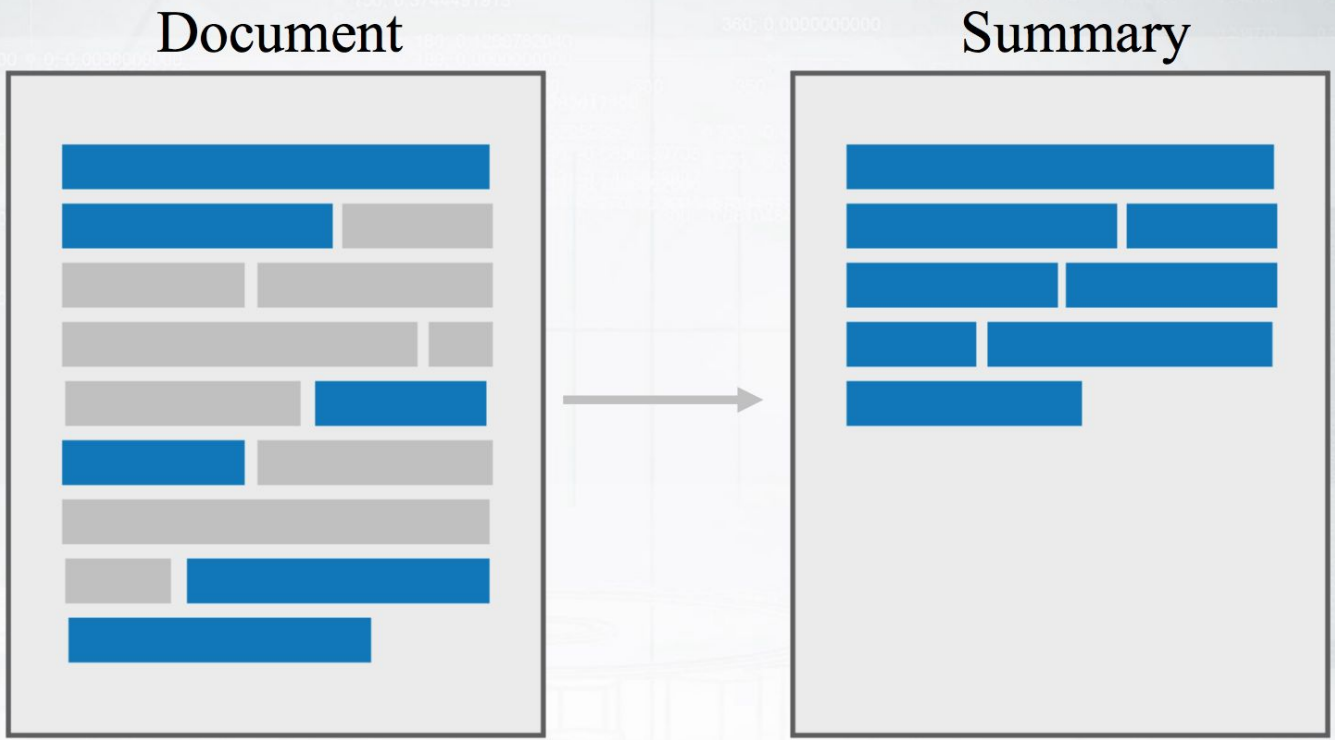
https://www.coursera.org/learn/language-processing

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
- …

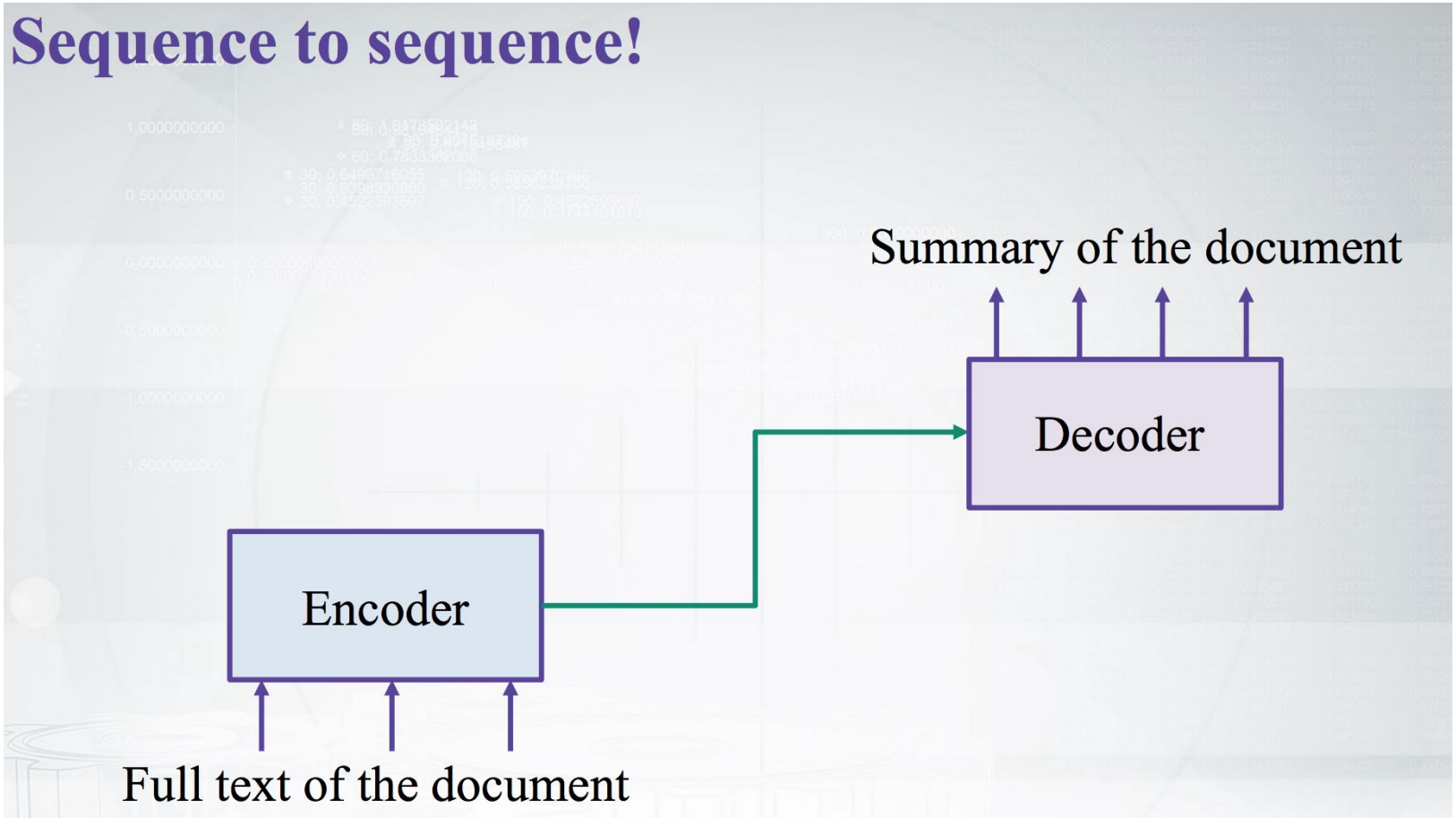
https://www.coursera.org/learn/language-processing

Summarization



https://www.coursera.org/learn/language-processing

Sequence to sequence!



https://www.coursera.org/learn/language-processing

From Google research blog

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*

https://www.coursera.org/learn/language-processing

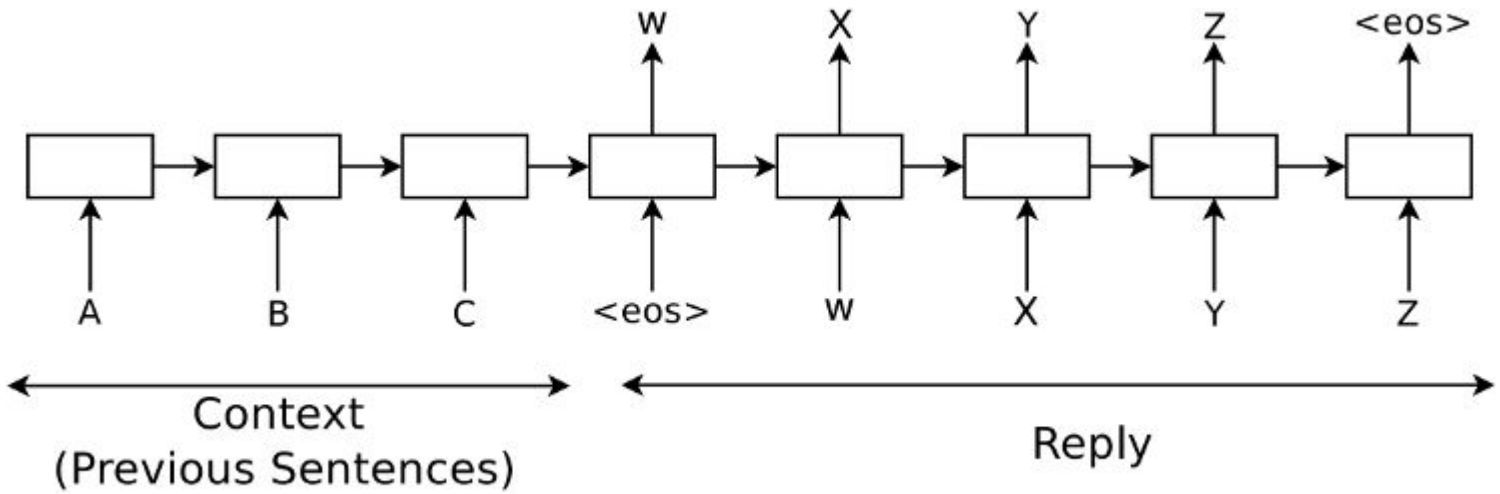
Simplification

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.

Dialog systems

General conversation models can be simply divided into two major types—generative and selective (or ranking) models.



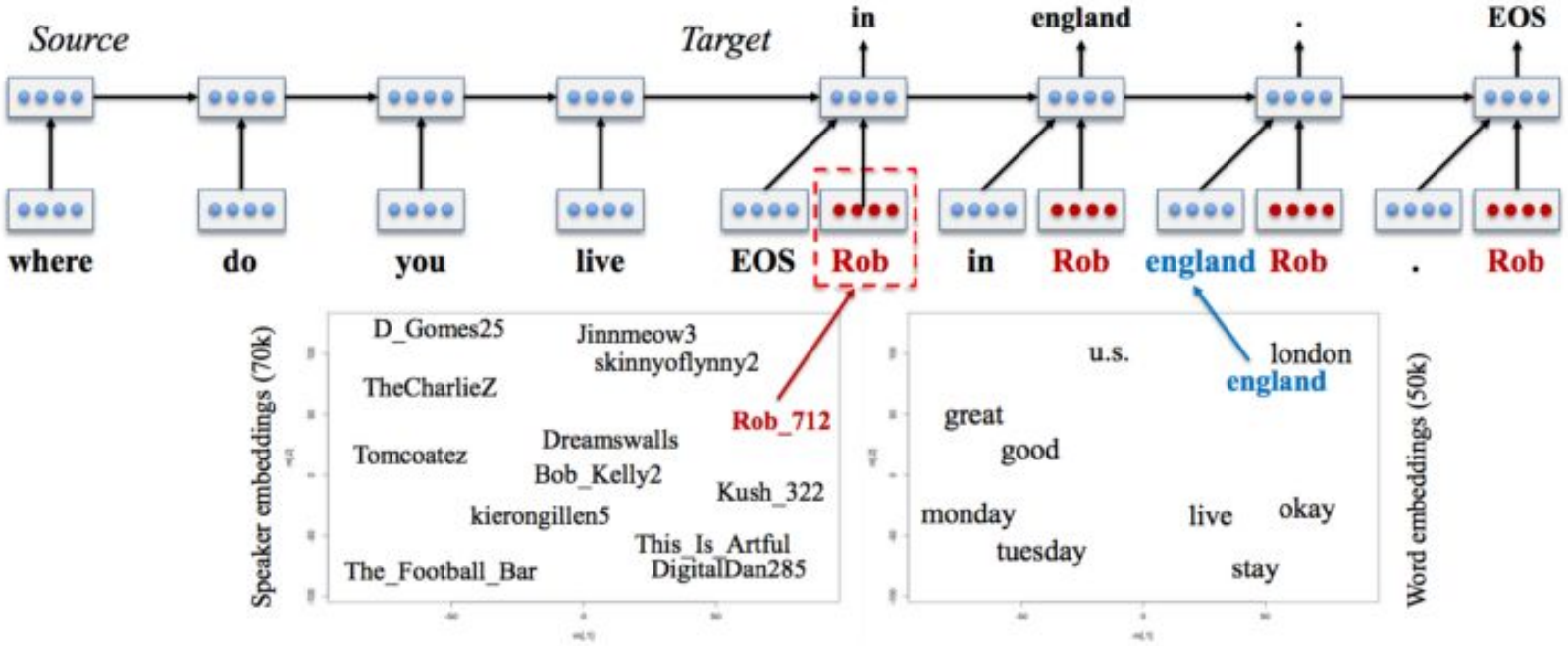
Problems of generative approach

- Generic responses: generative models trained via maximum likelihood tend to predict high probability for general replies such as “Okay,” “No,” “Yes,” and “I don’t know” for a wide range of contexts
- Reply inconsistency / how to incorporate metadata

message	Where do you live now?
response	I live in Los Angeles.
message	In which city do you live now?
response	I live in Madrid.
message	In which country do you live now?
response	England, you?
message	Where were you born?
response	I was born in Canada.
message	Where are you from?
response	England, you?
message	Where did you grow up?
response	I grew up in Texas.
message	How old are you?
response	16 and you?
message	What’s your age?
response	18.
message	What is your major?
response	I’m majoring in psychology
message	What did you study in college?
response	English lit.

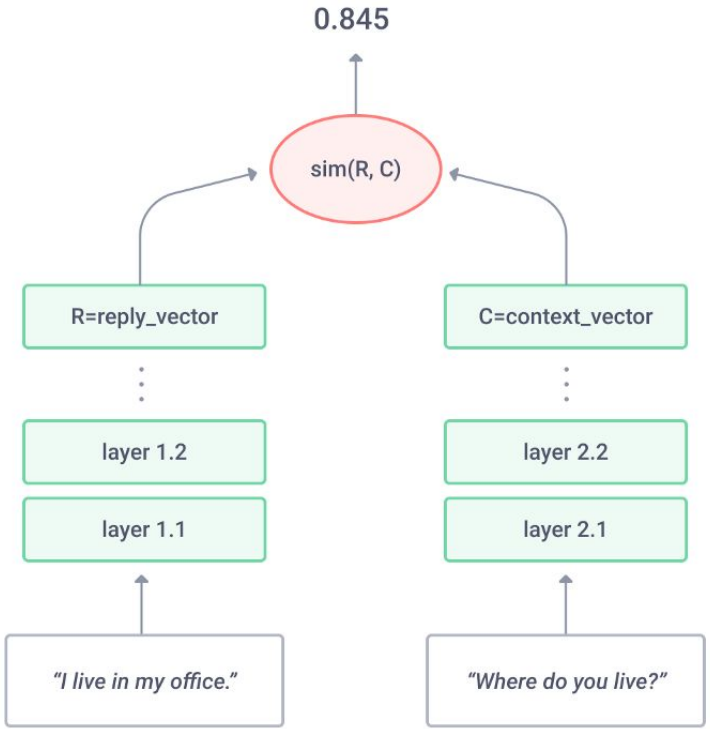
Table 1: Inconsistent responses generated by a 4-layer SEQ2SEQ model trained on 25 million Twitter conversation snippets.

The most cited work dealing with it is “A Persona-Based Neural Conversation Model.” Authors used speaker ids for each utterance in order to generate an answer, which conditioned not only on encoder state, but also on speaker embedding. Speaker embeddings are learned from scratch along with the model.



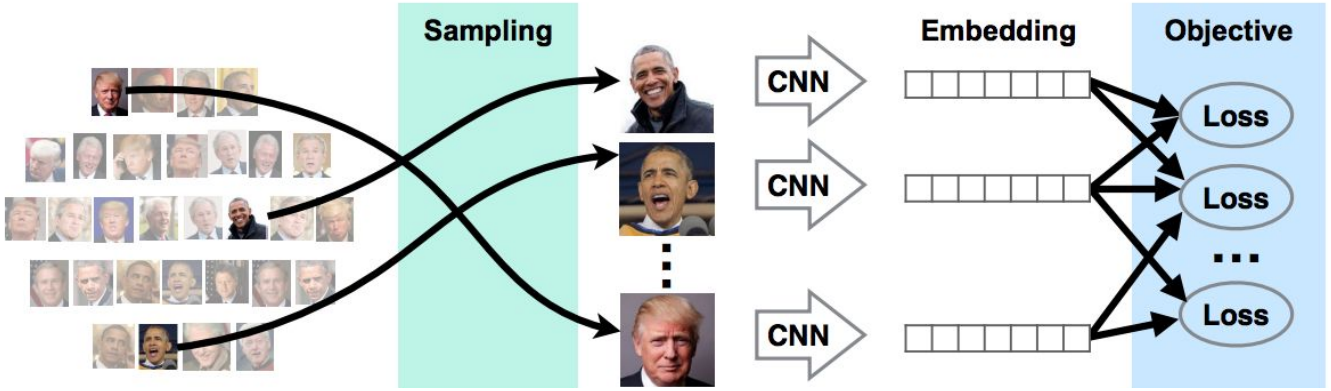
Selective models

Instead of estimating probability p(reply | context; w), selective models learn similarity function—sim(reply, context; w), where a reply is one of the elements in a predefined pool of possible answers.



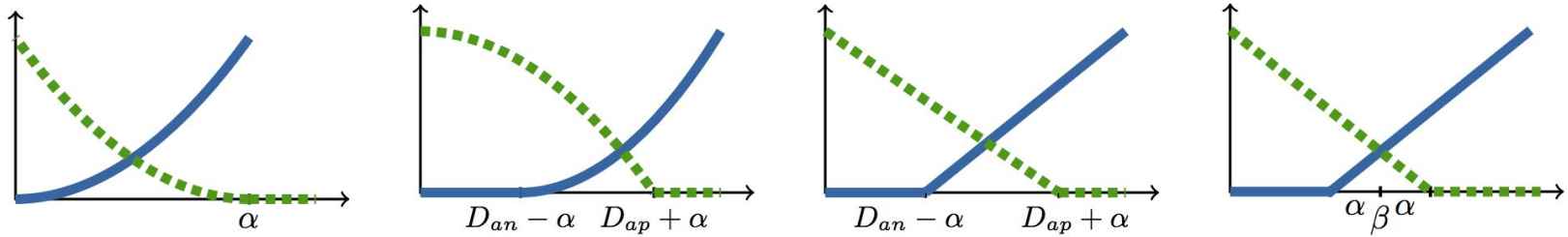
$$L = \max\left(0, sim(\texttt{ctx}, \texttt{reply}_{wrong}) - sim(\texttt{ctx}, \texttt{reply}_{correct}) + \alpha\right) \to \min$$

Triplet loss



$$\ell^{\text{triplet}}(a,p,n) := \left[D_{ap}^2 - D_{an}^2 + \alpha\right]_+$$

$$\ell^{\text{margin}}(i,j) := \left(\alpha + y_{ij}(D_{ij} - \beta)\right)_+$$



(a) Contrastive loss [11] (b) Triplet loss ℓ_2^2 [25] (c) Triplet loss ℓ_2 (d) Margin based loss

Figure 4: Loss vs. pairwise distance. The solid blue lines show the loss function for positive pairs, the dotted green for negative pairs. Our loss finds an optimal boundary β between positive and negative pairs, and α ensures that they are separated by a large margin.

Generative vs selective: pros and cons

Generative models	Selective models
+ Can possibly generate arbitrary answer (more similar to general AI) + Can generate answer in correct grammar form (e.g. with correct speaker gender) - Can generate answer with incorrect grammar/syntax - Prone to “general answer” problem - Difficult to impose properties on model replies (e.g. no obscene words, speak like some specific person), but possible!	- Restricted pool of answers which can not cover all dialogue topics - For context “What is your name, girl?” can select “My name is Stephen.” (inconsistency) + Predefined answers have good grammar/syntax + Less prone to “general answer” problems + You can customize answers for your own needs (without obscenities, kind answers)

Evaluation

One of the most important questions is how to evaluate neural conversational models. There are many automatic metrics which are used to evaluate chatbots with machine learning:

- Precision/recall/accuracy for selective models
- Perplexity/loss value for generative models
- BLEU/METEOR scores from machine translation

Task-oriented dialog system

You can talk to a personal assistant:

- Apple Siri
- Google Assistant
- Microsoft Cortana
- Amazon Alexa
- …

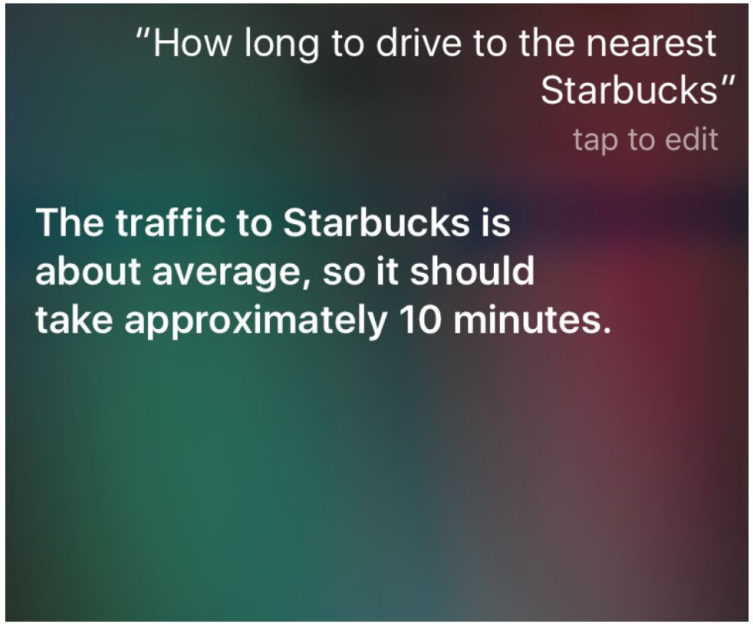
You can solve these tasks:

- Set up a reminder
- Find photos of your pet
- Find a good restaurant
- Send a message
- …

https://www.coursera.org/learn/language-processing

Intent classification

- What does the user want?
- Which predefined scenario is the user trying to execute?

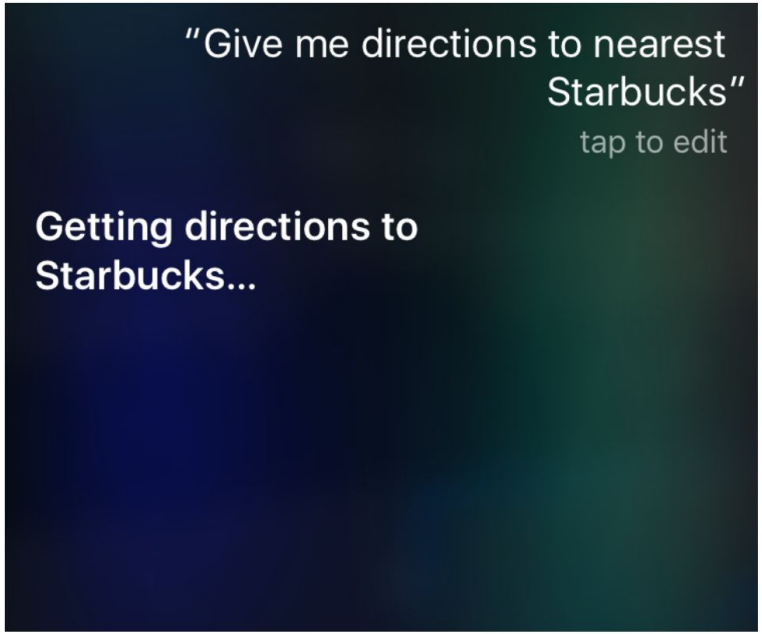


Intent: **nav.time.closest**

https://www.coursera.org/learn/language-processing

There’re many intents

- And you need to classify them to give correct answers
- This is a classification task and you can measure **accuracy**



Intent: **nav.directions.closest**

https://www.coursera.org/learn/language-processing

Form filling approach to dialog management

- Think of an intent as a **form** that a user needs to fill in.
- Each intent has a set of fields (**slots**) that must be filled in to execute the request.

- Example: **nav.directions** intent
  - **@FROM** slot: defaults to current geolocation
  - **@TO** slot: required

- We need a **slot tagger** to extract slots from utterance.

Slot filling/tagging

- Example:
  - User: Show me the way to History Museum
  - Slot tagger: Show me the way to @TO{History Museum}
- This is token classification task in **BIO** scheme:
  - **B** corresponds to the word at the **beginning** of the slot
  - **I** corresponds to the word **inside** the slot (excluding beginning)
  - **O** corresponds to words **outside** of slots
- Example of **BIO** scheme tagging:

Tokens	Show	me	the	way	to	History	Museum
Tags	O	O	O	O	O	B-TO	I-TO

https://www.coursera.org/learn/language-processing

Form filling dialog manager (single turn)

- User: **Give me directions to San Francisco**
  - Intent classifier: nav.directions
  - Slot tagger: @TO{San Francisco}
  - Dialog manager: *all slots are filled, here's the route*
- Agent (assistant): **Here's the route**

Google Maps

Form filling dialog manager (multi-turn)

- User: **Give me directions from Los Angeles**
  - Intent classifier: nav.directions
  - Slot tagger: @FROM{Los Angeles}
  - Dialog manager: *required slot is missing, where to?*
- Agent (assistant): **Where do you want to go?**
- User: **San Francisco**
  - Intent classifier: nav.directions
  - Slot tagger: @TO{San Francisco}
  - Dialog manager: *okay, here's the route*
- Agent (assistant): **Here's the route**

Google Maps

Form filling dialog manager (multi-turn)

- User: **Give me directions from Los Angeles**
  - Intent classifier: nav.directions
  - Slot tagger: @FROM{Los Angeles}
  - Dialog manager: *required slot is missing, where to?*
- Agent (assistant): **Where do you want to go?**
- User: **San Francisco**
  - Intent classifier: nav.directions
  - Slot tagger: @TO{San Francisco}
  - Dialog manager: *okay, here's the route*
- Agent (assistant): **Here's the route**

We need context here

Google Maps

https://www.coursera.org/learn/language-processing

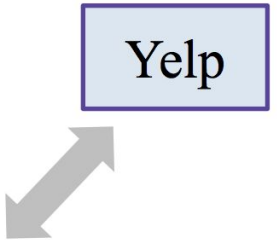
How to track context (an easy way)

- Both intent classifier and slot tagger need context (what happened before)

- Let's add simple **features** to both of them:
  - Previous utterance intent as a categorical feature
  - Slots filled in so far with binary feature for each possible slot
- Improves slot tagger F1 by 0.5%
- Reduces intent classifier error by 6.7%

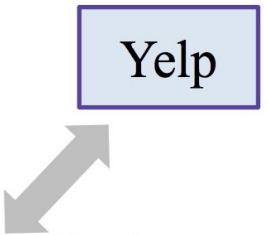
How to track a form switch

- User: **Give me directions from Los Angeles**
  - Intent classifier: nav.directions
  - Slot tagger: @FROM{Los Angeles}
  - Dialog manager: *required slot is missing, where to?*
- Agent (assistant): **Where do you want to go?**
- User: **Forget about it, let's eat some sushi first**
  - Intent classifier: nav.find
  - Slot tagger: @CATEGORY{sushi}
  - Dialog manager: *okay, let's start a new form and find some sushi*
- Agent (assistant): **Okay, here are nearby sushi places**

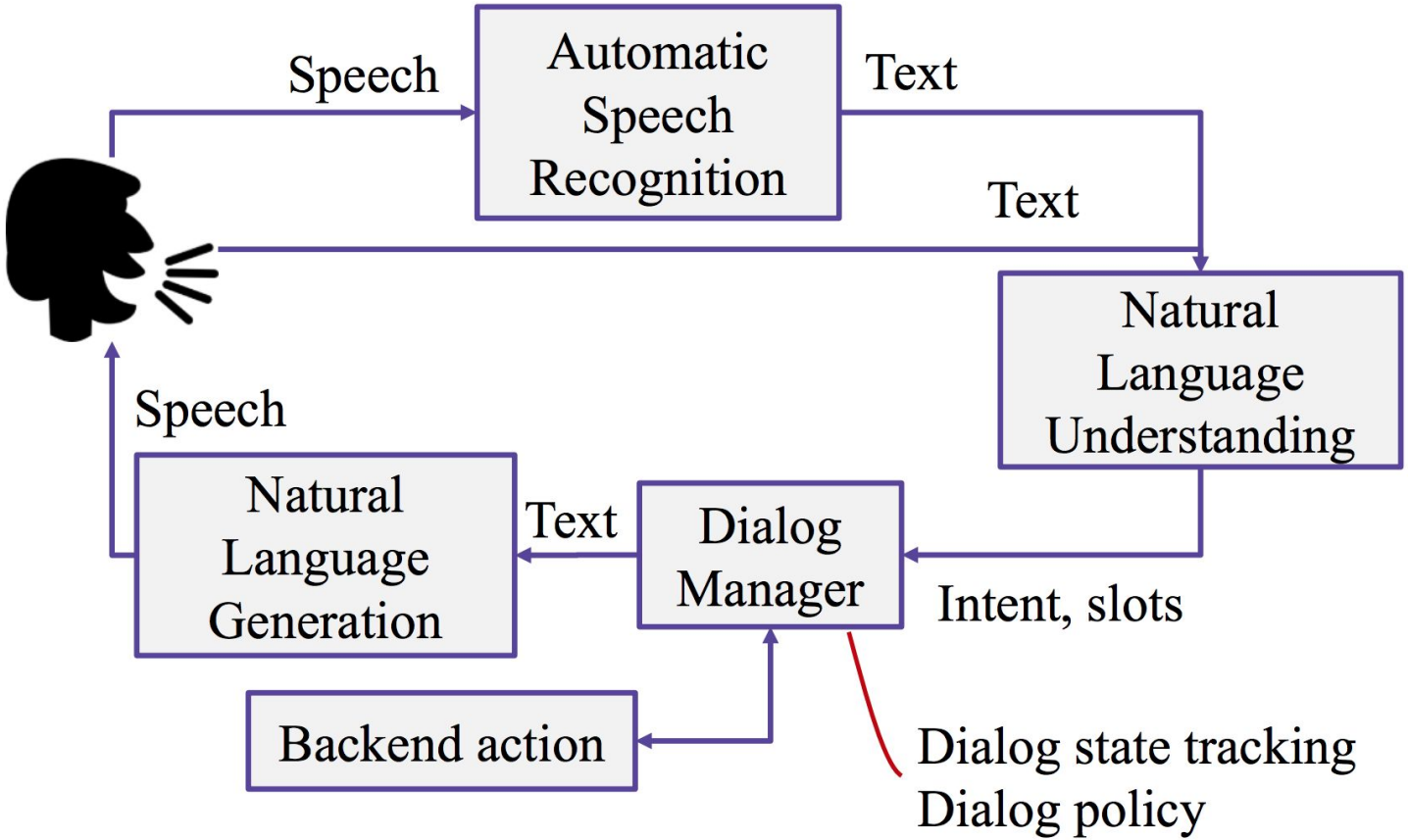


How to track a form switch

- User: **Give me directions from Los Angeles**
  - Intent classifier: nav.directions
  - Slot tagger: @FROM{Los Angeles}
  - Dialog manager: *required slot is missing, where to?*
- Agent (assistant): **Where do you want to go?**
- User: **Forget about it, let's eat some sushi first**
  - Intent classifier: nav.find
  - Slot tagger: @CATEGORY{sushi}
  - Dialog manager: *okay, let's start a new form and find some sushi*
- Agent (assistant): **Okay, here are nearby sushi places**



Task-oriented dialog system overview



https://www.coursera.org/learn/language-processing

Intent classifier

- What you can do:
  - Any model on BOW with n-grams and TF-IDF
  - RNN (LSTM, GRU, …)
  - CNN (1D convolutions)

- CNNs can perform better on datasets where the task is essentially a key phrase recognition task as in some sentiment detection datasets.

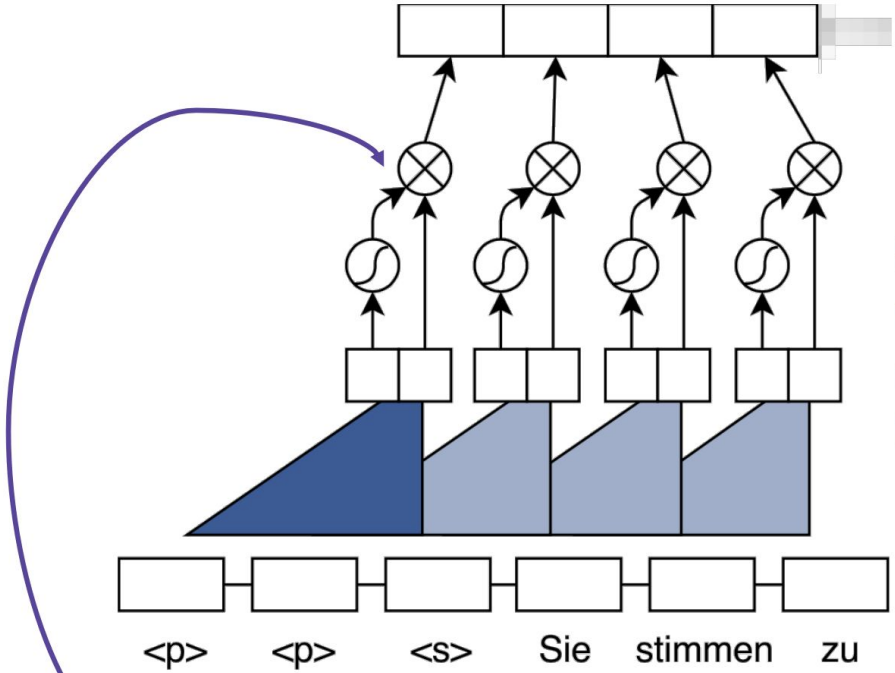
https://www.coursera.org/learn/language-processing

Slot tagger

- What you can do:
  - Handcrafted rules like regular expressions
  - CRF
  - RNN seq2seq
  - **CNN seq2seq**
  - Any seq2seq with attention

https://www.coursera.org/learn/language-processing

CNN for sequences: Gated Linear Unit



Stacking 6 layers with kernel size 5 results in an input field of 25 elements

$$h_l(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c})$$

https://arxiv.org/pdf/1612.08083.pdf https://arxiv.org/pdf/1705.03122.pdf

https://www.coursera.org/learn/language-processing

CNN for sequences: results

- They can sometimes beat LSTM in **language modeling**:

Model	Test PPL	Hardware
LSTM-1024 (Grave et al., 2016b)	48.7	1 GPU
GCNN-8	44.9	1 GPU
GCNN-14	37.2	4 GPUs

Table 3. Results for single models on the WikiText-103 dataset.

- … and **machine translation**:

WMT'14 English-French	**BLEU**
Wu et al. (2016) GNMT (Word 80K)	37.90
Wu et al. (2016) GNMT (Word pieces)	38.95
Wu et al. (2016) GNMT (Word pieces) + RL	39.92
ConvS2S (BPE 40K)	40.51

https://arxiv.org/pdf/1612.08083.pdf https://arxiv.org/pdf/1705.03122.pdf

CNN for sequences: speed benefit

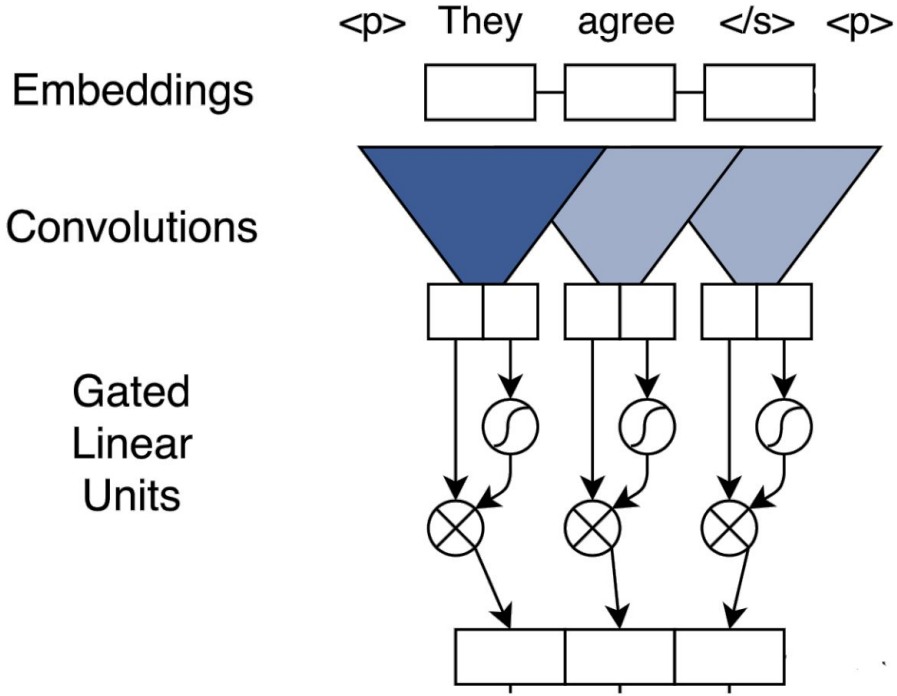
- They work faster than RNN:
  - During **training** we can process all time steps in parallel
  - During **testing** encoder can do the same
  - During **testing** we get higher throughput thanks to convolution optimizations in GPUs

	BLEU	Time (s)
GNMT GPU (K80)	31.20	3,028
GNMT CPU 88 cores	31.20	1,322
GNMT TPU	31.21	384
ConvS2S GPU (K40) $b = 1$	33.45	327
ConvS2S GPU (M40) $b = 1$	33.45	221
ConvS2S GPU (GTX-1080ti) $b = 1$	33.45	142
ConvS2S CPU 48 cores $b = 1$	33.45	142

Translation generation speed during testing

https://arxiv.org/pdf/1705.03122.pdf

CNN for sequences: how encoder looks like



- Bi-directional encoder is easy
- Works in parallel for all time steps

https://arxiv.org/pdf/1705.03122.pdf

https://www.coursera.org/learn/language-processing

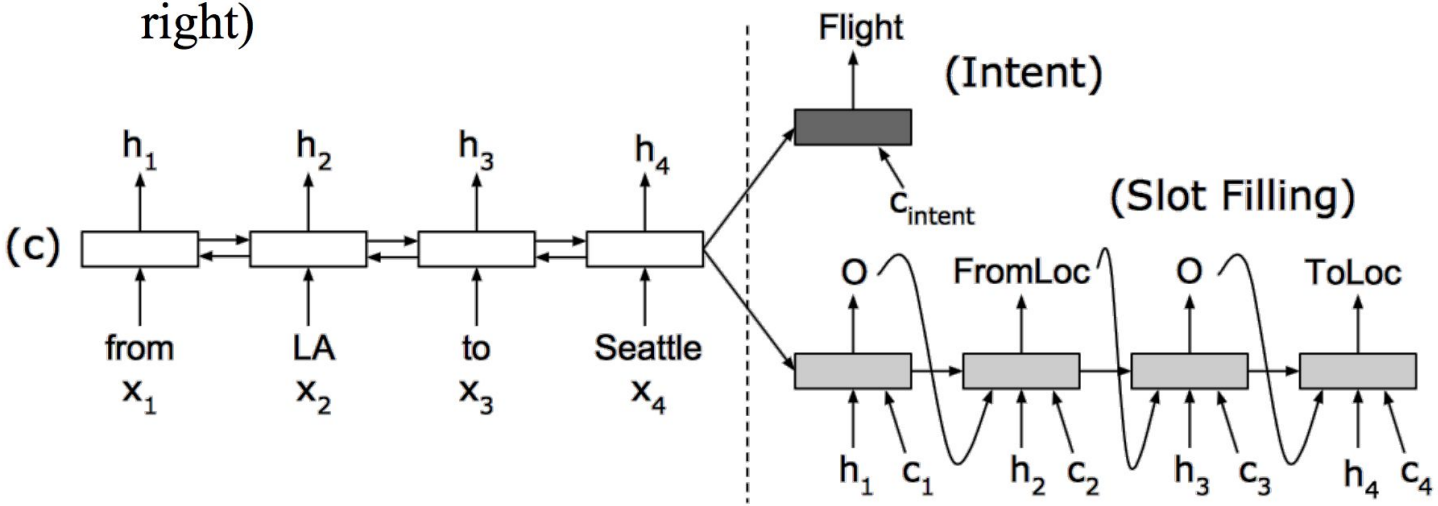
ATIS dataset

- Airline Travel Information System
- Collected in 90s
- 4978 context independent utterances
- 17 intents, 127 slot labels
- State-of-the-art: 1.79% intent error, 95.9 slots F1

Utterance	show	flights	from	Seattle	to	San	Diego	tomorrow
Slots	O	O	O	B-fromloc	O	B-toloc	I-toloc	B-depart_date
Intent	Flight							

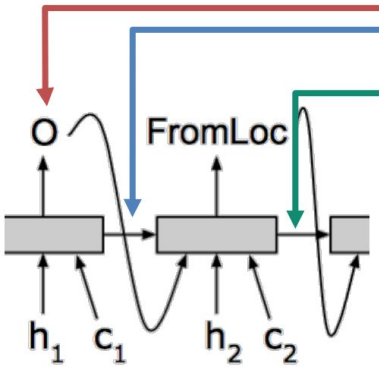
Joint training of intent classifier and slot tagger

- Encoder-decoder architecture for joint intent detection and slot filling
- Encoder is a bi-directional LSTM
- With aligned inputs (h_i on the right) and attention (c_i on the right)



https://www.coursera.org/learn/language-processing

Attention in decoder



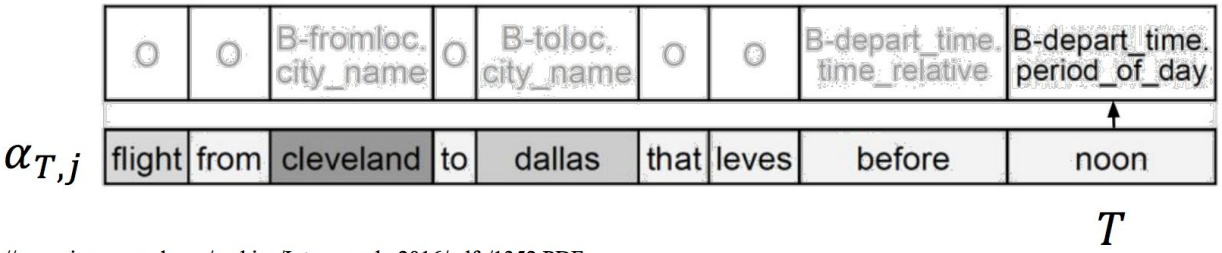
Decoder state s_i: $s_i = f(s_{i-1}, y_{i-1}, h_i, c_i)$

Attention c_i (g is a feed forward network):

$$c_i = \sum_{j=1}^{T} \alpha_{i,j} h_j \quad \alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{T} \exp(e_{i,k})}$$

$$e_{i,k} = g(s_{i-1}, h_k)$$

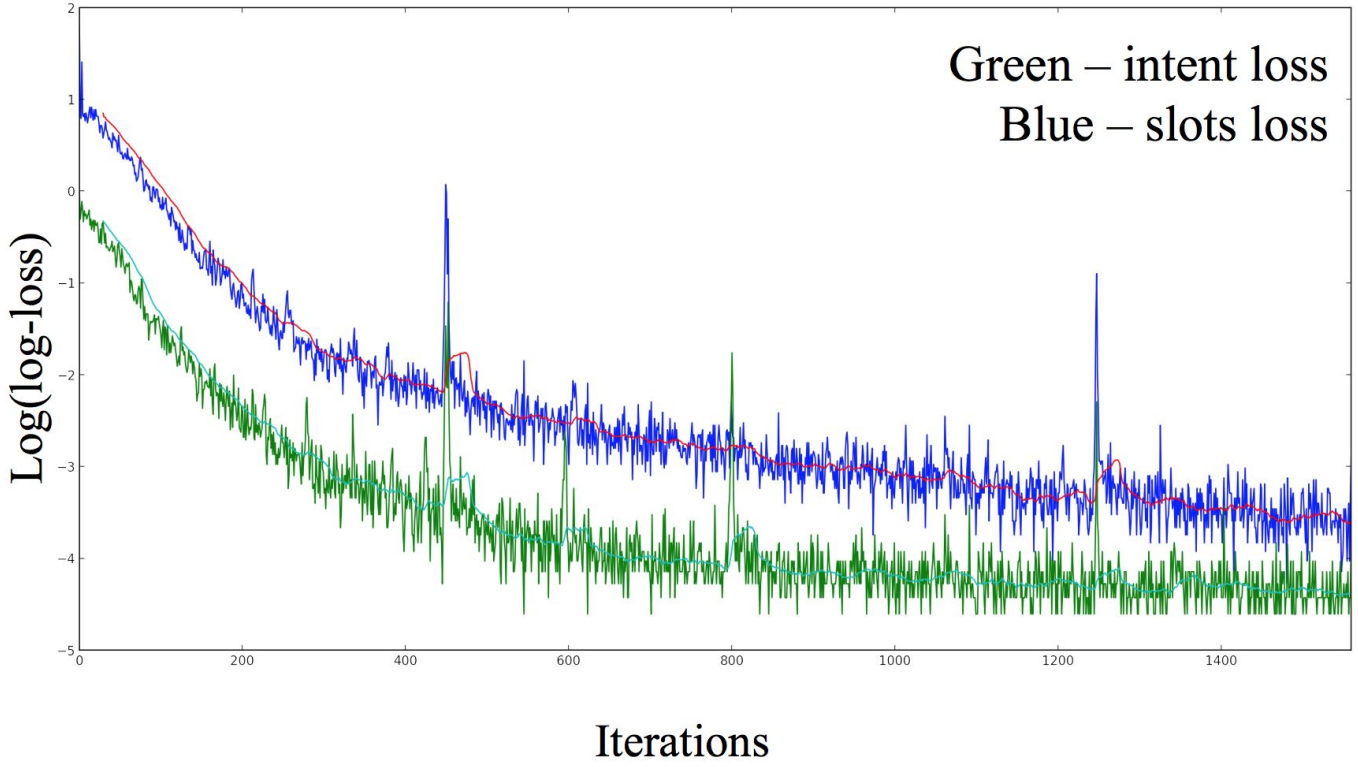
Attention weights (the darker the higher) when predicting the slot label for the last word "noon":



http://www.isca-speech.org/archive/Interspeech_2016/pdfs/1352.PDF

Joint training loss

- Final training loss is a sum of losses for intent and slots



https://www.coursera.org/learn/language-processing

Joint training results

- Better performance on ATIS dataset:

Training	Slots F1	Intent % error
Independent training for slot filling	95.78	-
Independent training for intent detection	-	2.02
Joint training for slot filling and intent detection	**95.87**	**1.57**

- Works faster than two separate models

Why do we want to utilize lexicon?

- Let's take ATIS dataset
- It has finite set of cities in training
- Will the model work for a new city?
- We have **a list of all cities**, why not use it?

- Another example
- Imagine you need to fill a slot "music artist"
- We have **all music artists** in the database like musicbrainz.org
- How can we use it?

Let's add lexicon features to input words

- Let's **match every n-gram** of input text against entries in our lexicon

Take me to San Francisco

- A match is successful when **the n-gram matches the prefix or postfix** of an entry and is at least half the length of the entry

Matches:

"San" → "San Antonio"
"San" → "San Francisco"
"San Francisco" → "San Francisco"

- When there are multiple **overlapping matches**:
  - Prefer **exact** matches over partial
  - Prefer **longer** matches over shorter
  - Prefer **earlier** matches in the sentence over later

https://arxiv.org/pdf/1511.08308v4.pdf

Matches encoding

We will use **BIOES** coding (Begin, Inside, Outside, End, Single)

- B – if token matches the beginning of some entity
- B, I – if two tokens match as prefix
- I, E – if two tokens match as postfix
- S – if matched single token entity
- …

Example for 4 lexicon dictionaries:

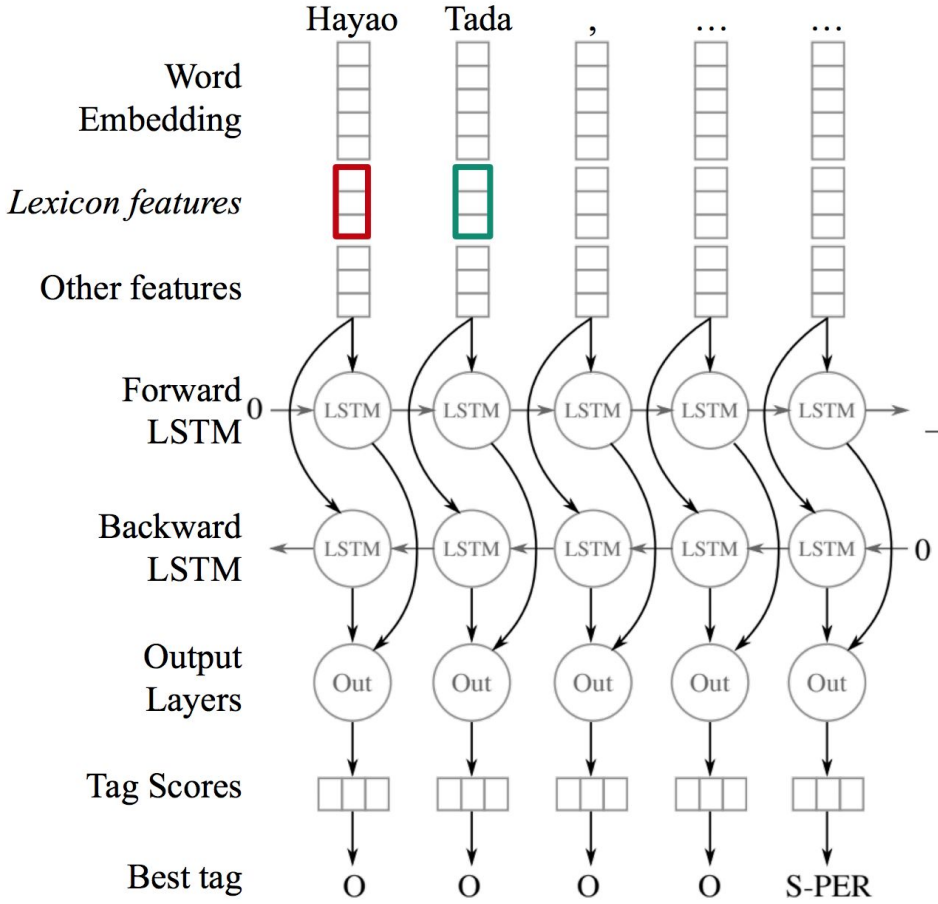
Text	Hayao	Tada	,	commander	of	the	Japanese	North	China	Area	Army
LOC	–	–	–	–	–	B	I	–	S	–	–
MISC	–	–	–	S	B	B	I	S	S	S	S
ORG	–	–	–	–	–	B	I	B	I	I	E
PERS	B	E	–	–	–	–	–	–	S	–	–

B, I, O, E, S are later encoded as one-hot vectors

https://arxiv.org/pdf/1511.08308v4.pdf

https://www.coursera.org/learn/language-processing

Adding these features to our model



These're our lexicon features

Text	Hayao	Tada	,
LOC	–	–	–
MISC	–	–	–
ORG	–	–	–
PERS	B	E	–

https://arxiv.org/pdf/1511.08308v4.pdf

https://www.coursera.org/learn/language-processing

Does lexicon help?

CoNLL-2003 Named Entity Recognition task:

Model	CoNLL-2003		
	Prec.	Recall	F1
FFNN + emb + caps + lex	89.54	89.80	89.67 (± 0.24)
BLSTM	80.14	72.81	76.29 (± 0.29)
BLSTM-CNN	83.48	83.28	83.38 (± 0.20)
BLSTM-CNN + emb	90.75	91.08	90.91 (± 0.20)
BLSTM-CNN + emb + lex	91.39	**91.85**	**91.62** (± 0.33)

Yes, it does!

Links

- https://habr.com/ru/company/oleg-bunin/blog/352614/
- https://cube.dev/blog/chatbots-machine-learning/
- https://cube.dev/blog/machine-learning-translation/
- https://cube.dev/blog/recommendation-system-algorithms/
- https://arxiv.org/pdf/1707.07435.pdf
- https://github.com/huggingface/pytorch-transformers
- https://arxiv.org/pdf/1704.04368.pdf
- https://arxiv.org/pdf/1603.06155.pdf
- https://arxiv.org/abs/1603.08023
- https://arxiv.org/abs/1703.09439
- https://arxiv.org/pdf/1706.07567.pdf