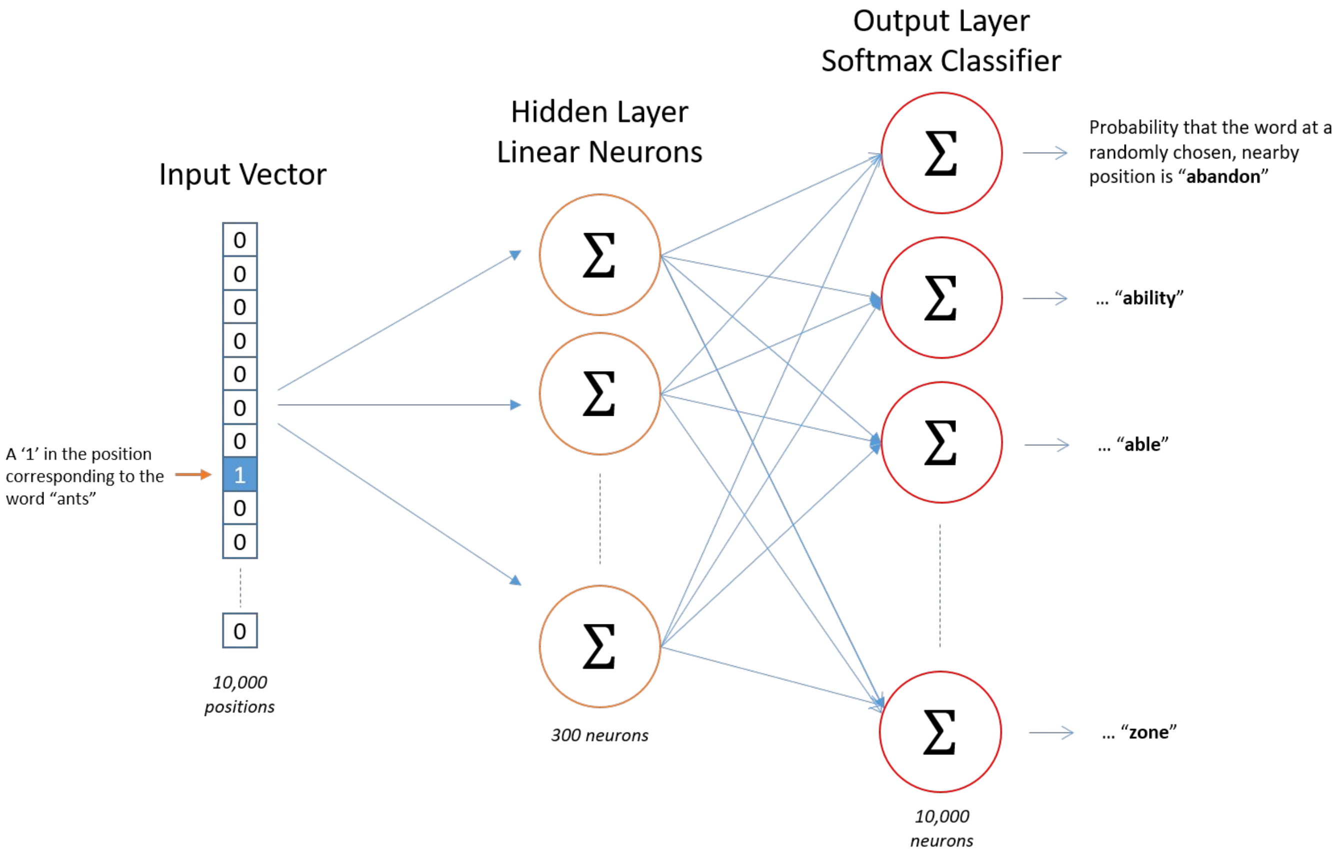
Machine Translation, Attention Mechanism

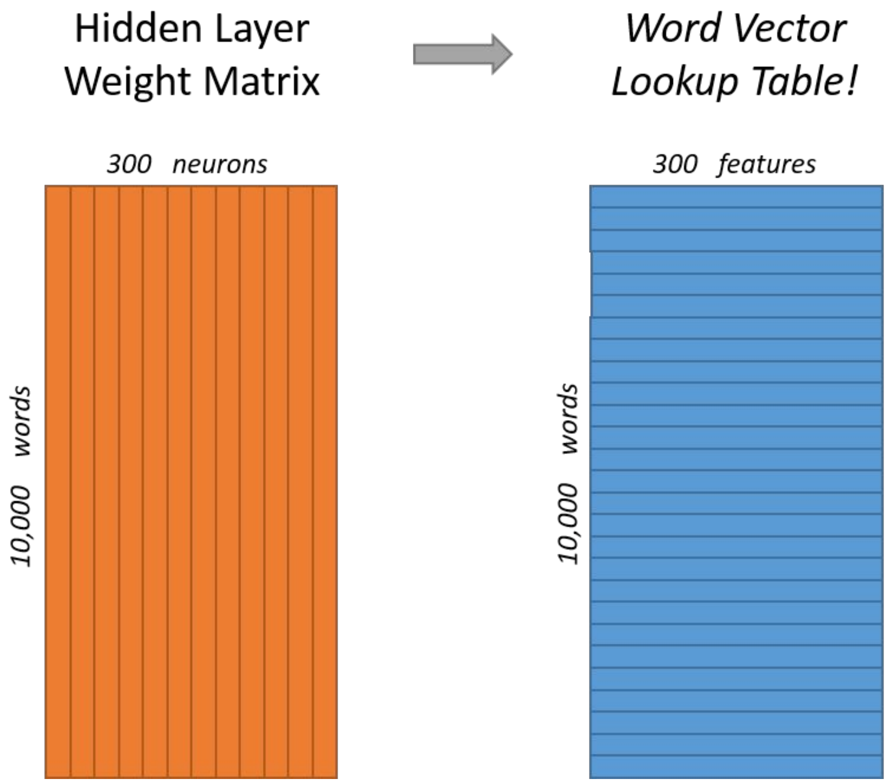
Radoslav Neychev

Embeddings: word2vec



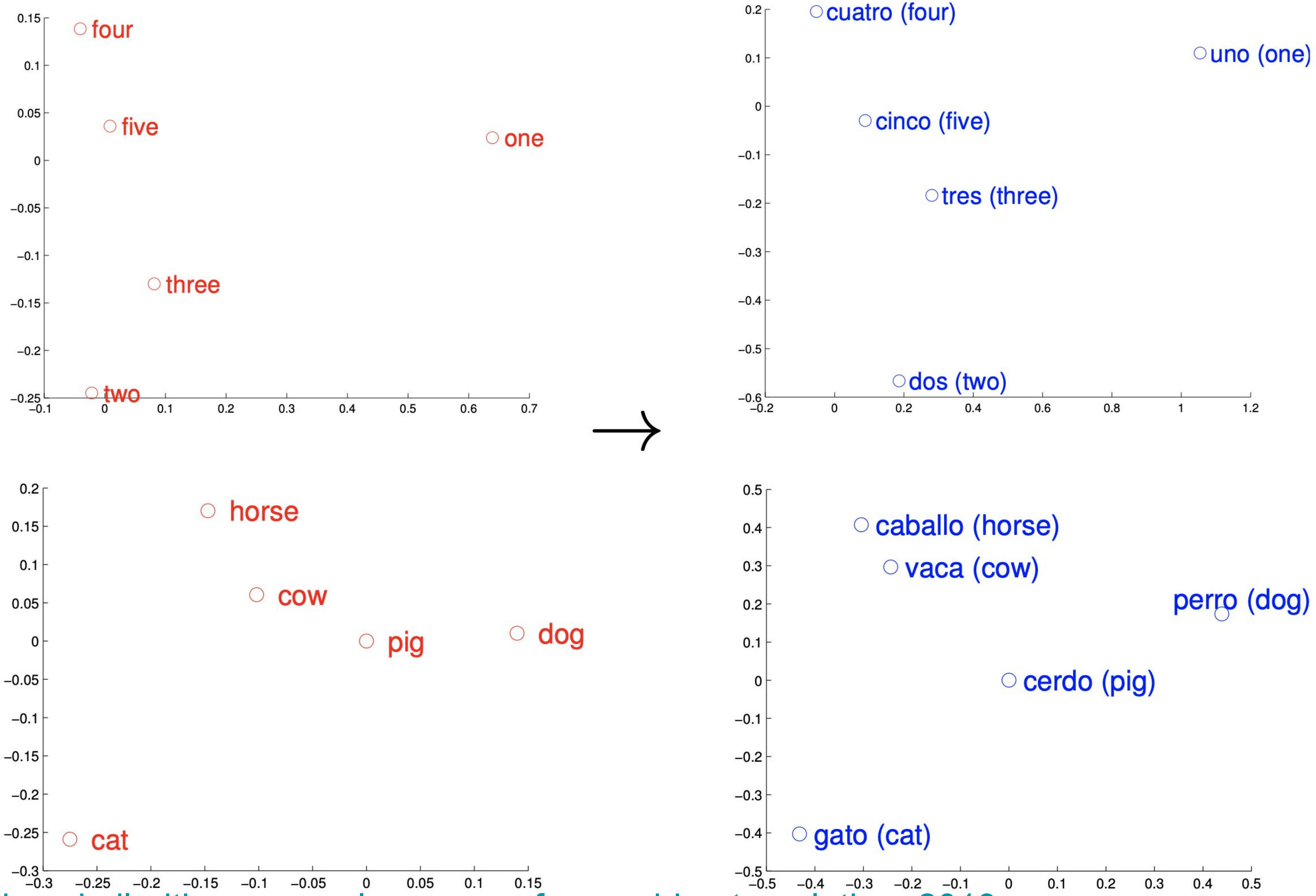
Source: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Embeddings: word2vec



Source: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Word embeddings in different languages



Source: [Exploiting similarities among languages for machine translation, 2013]

Word embeddings in different languages

- Word embeddings are quite similar for different languages
- Assume there n = 5000 word-translation pairs $\{x_i, y_i\}_{i \in \{1,n\}}$
- Learn linear mapping between the source and target spaces

$$W^\star = \underset{W \in M_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_{\mathrm{F}}$$

- The translation of source word is $t = \operatorname{argmax}_t \cos(Wx_s, y_t)$.

Sources: Exploiting similarities among languages for machine translation, 2013
Word translation without parallel data, ICLR 2018

Word embeddings in different languages

- Word embeddings are quite similar for different languages
- Assume there n = 5000 word-translation pairs $\{x_i, y_i\}_{i \in \{1,n\}}$
- Learn linear mapping between the source and target spaces

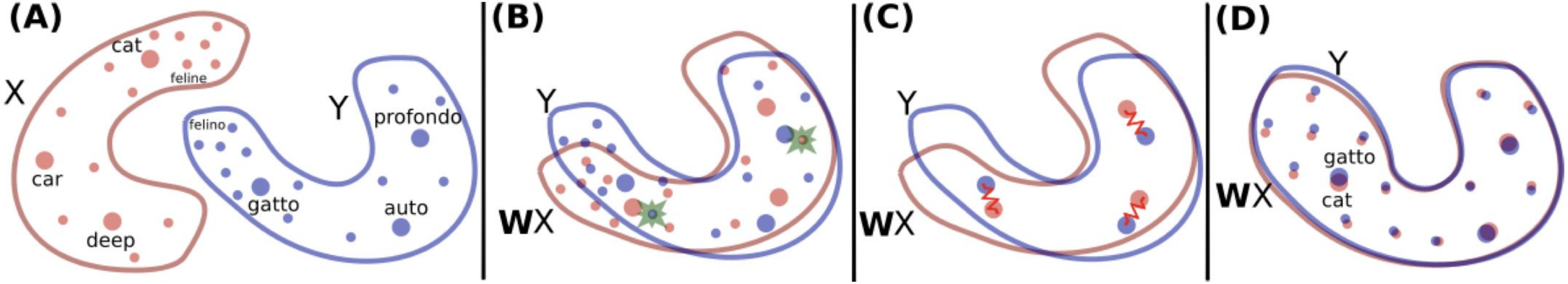
  enforcing an orthogonality constraint on W:

$$W^\star = \underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_{\mathrm{F}} = UV^T, \text{with } U\Sigma V^T = \mathrm{SVD}(YX^T).$$

- The translation of source word is $t = \operatorname{argmax}_t \cos(Wx_s, y_t)$.

Sources: [Normalized word embedding and orthogonal transform for bilingual word translation, NAACL 2015](#)
[Word translation without parallel data, ICLR 2018](#)

Word embeddings in different languages



Comment: mapping between two languages can be done completely in unsupervised manner with GANs.

We will meet later.

More info available in the mentioned paper:

Source: [Word translation without parallel data, ICLR 2018]

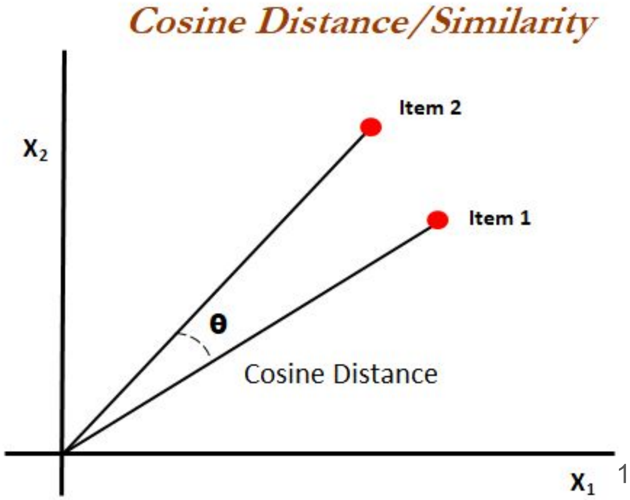
Why cosine distance/similarity?

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

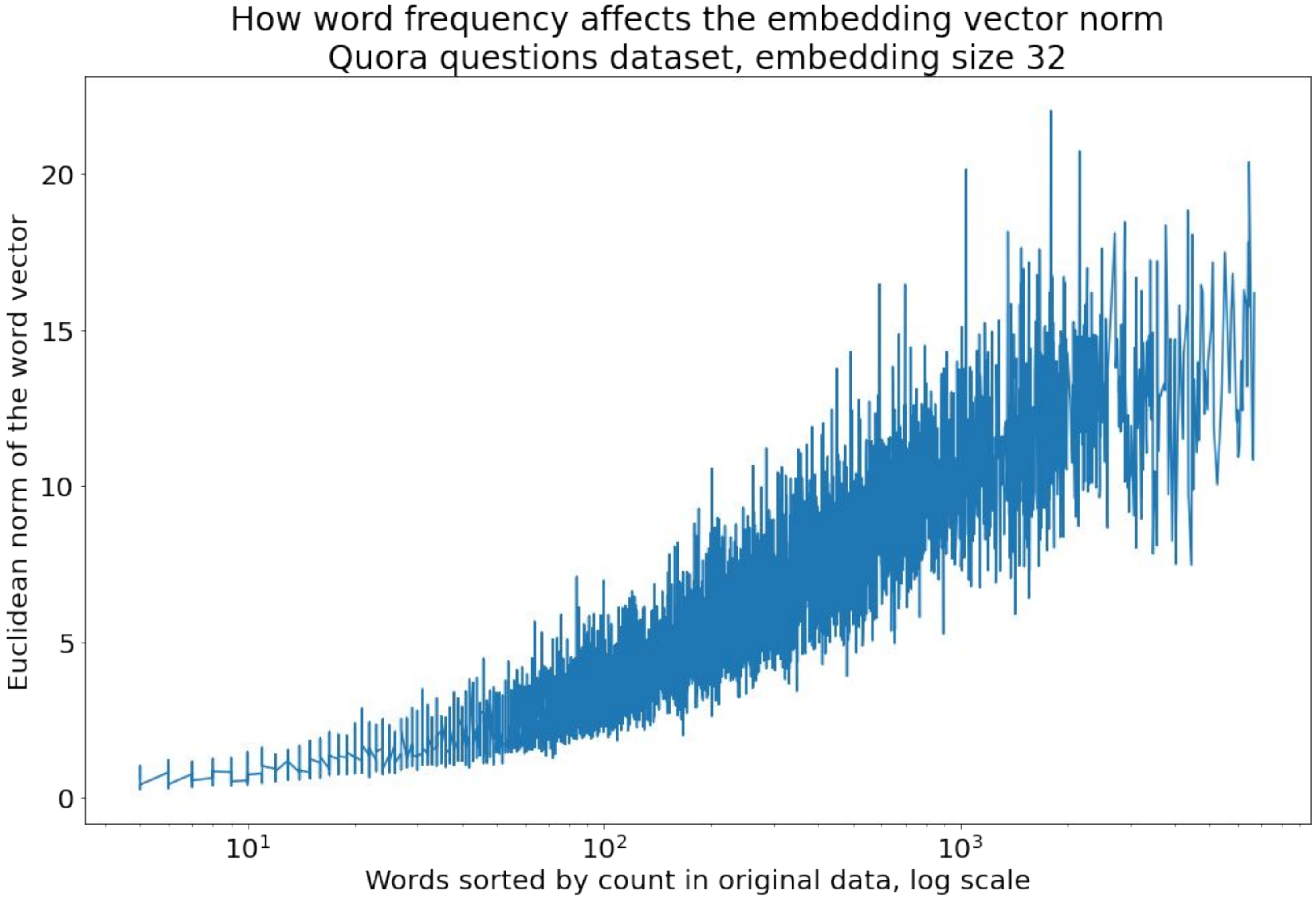
Cosine distance focuses on angle between the vectors.

With count-based approaches (e.g. BOW) it is really useful.

With word embeddings it is useful as well.



Source: [question]


How word frequency affects the embedding vector norm
Quora questions dataset, embedding size 32
Euclidean norm of the word vector
0
5
10
15
20
10^1
10^2
10^3
Words sorted by count in original data, log scale

Vector norms for words with no specific context

word	count	vector norm
overheat	11	0.81233
enormous	12	0.807057
dog	1212	11.2591
cat	1545	10.3738
laptop	1906	14.5192
phone	4124	15.7901
a	155726	11.4656
the	252068	8.47355

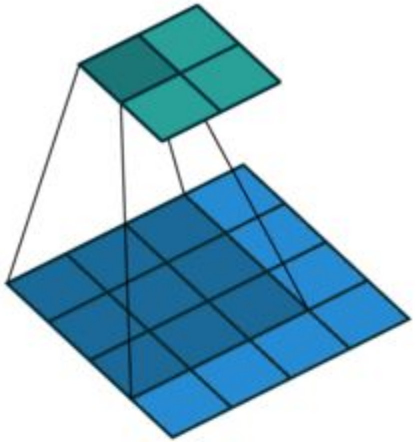
Outline

- Machine Translation historical overview
  - Statistical Machine Translation
  - Word alignments
- Neural Machine Translation (NMT)
  - Seq2Seq
  - Beam Search
- Attention mechanism

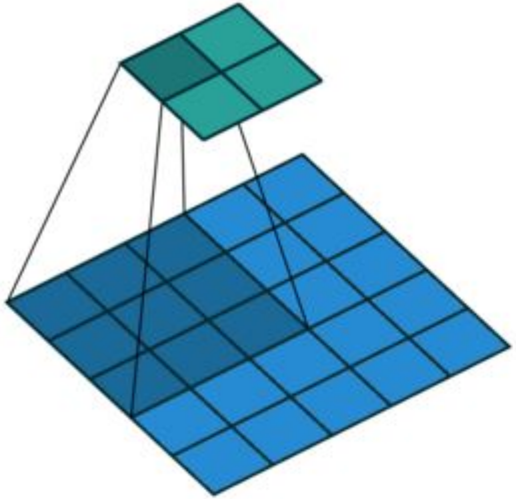
Based on: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Applying CNNs to texts

CNN simple recap



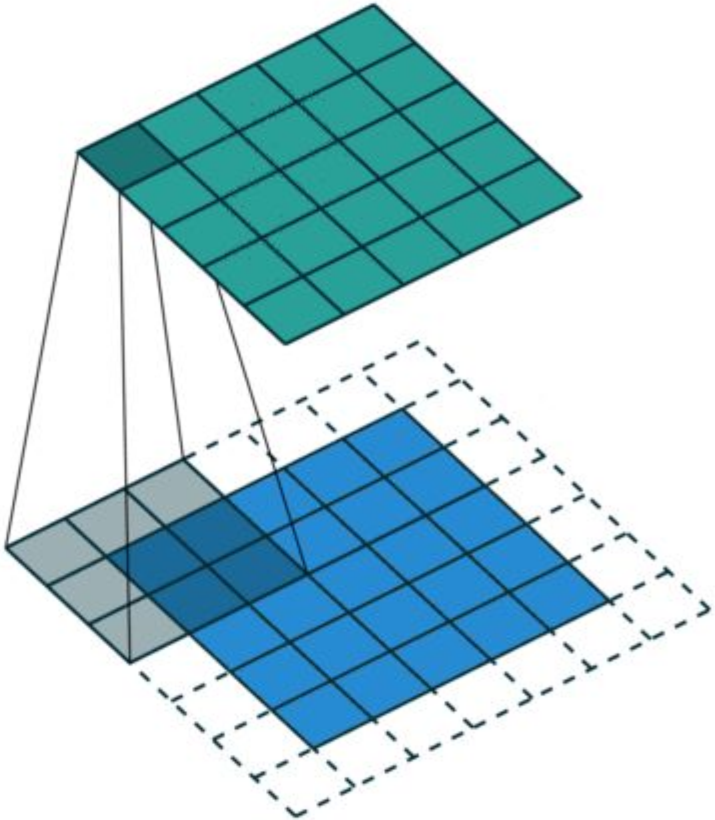
No padding,
no strides



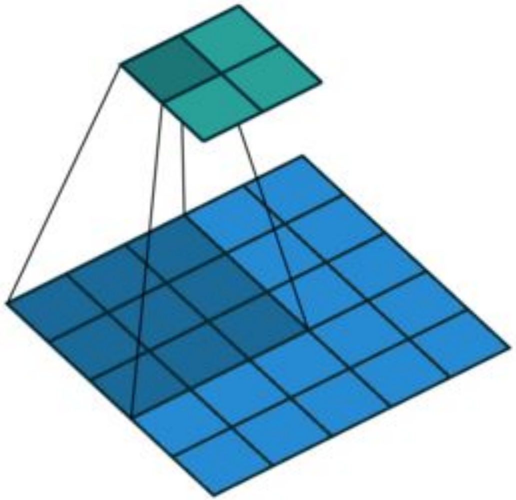
No padding,
with strides

Image source: https://github.com/vdumoulin/conv_arithmetic

CNN simple recap



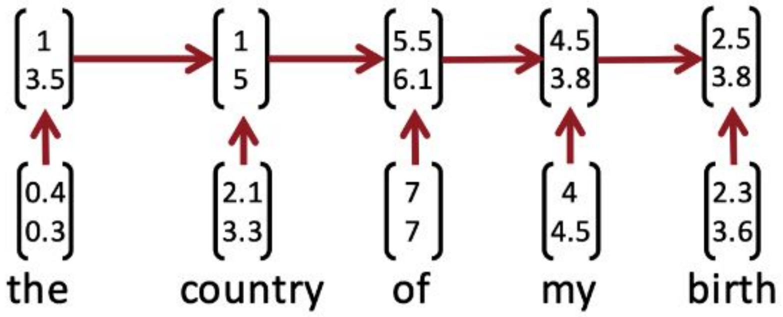
With padding,
no strides



No padding,
with strides

Image source: https://github.com/vdumoulin/conv_arithmetic

From RNN to CNN



- Recurrent neural nets can not capture phrases without prefix context and often capture too much of last words in final vector

Based on: Lecture by Richard Socher 5/12/16, http://cs224d.stanford.edu

From RNN to CNN

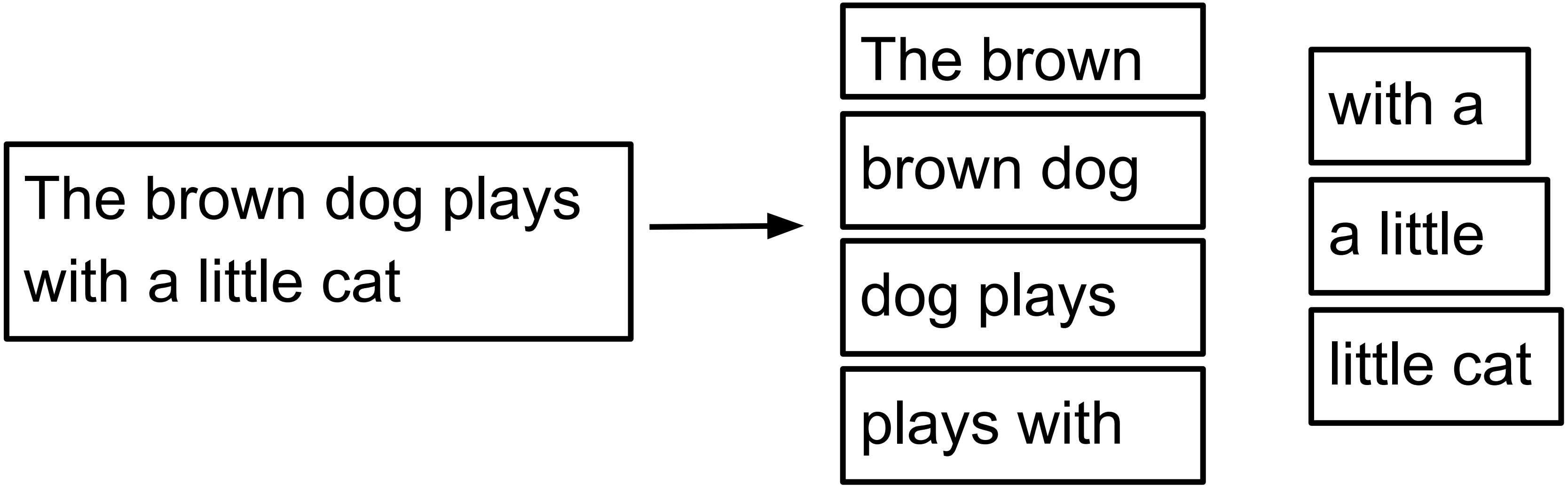
- RNN: Get compositional vectors for grammatical phrases only

- CNN: What if we compute vectors for every possible phrase?
  - Example: *"the country of my birth"* computes vectors for:
    - *the country, country of, of my, my birth, the country of, country of my, of my birth, the country of my, country of my birth*

- Regardless of whether it is grammatical
- Wouldn't need parser
- Not very linguistically or cognitively plausible

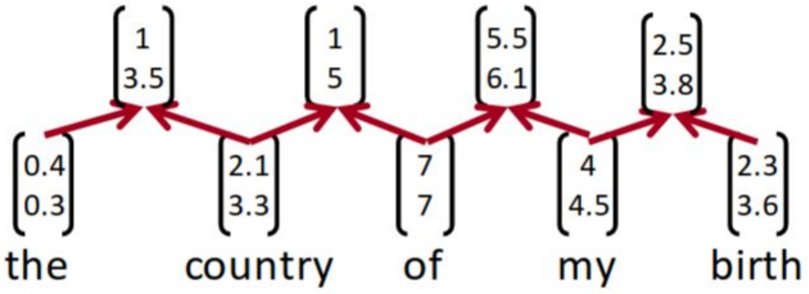
Based on: Lecture by Richard Socher 5/12/16, http://cs224d.stanford.edu

Recap: n-gramms



From RNN to CNN

- Imagine using only bigrams



- Same operation as in RNN, but for every pair

$$p = \tanh\left(W\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right)$$

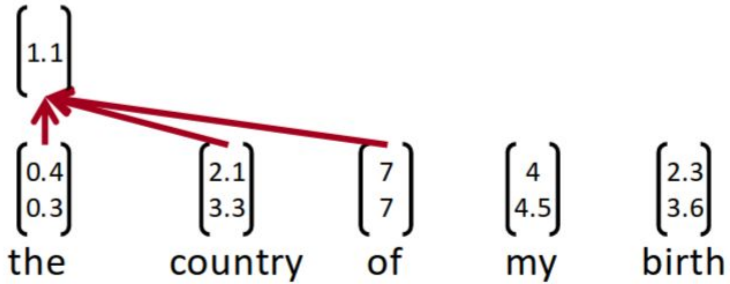
- Can be interpreted as convolution over the word vectors

Based on: Lecture by Richard Socher 5/12/16, http://cs224d.stanford.edu

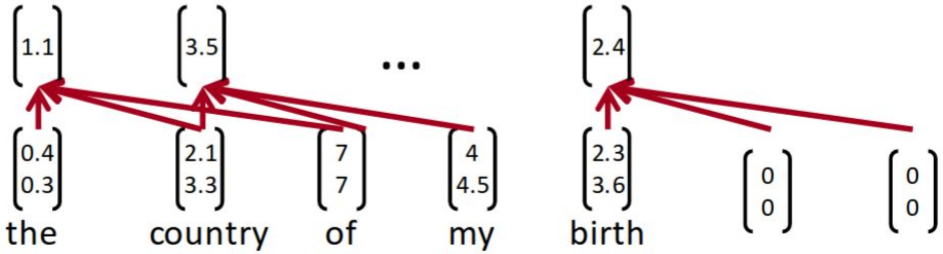
One layer CNN

- Simple convolution + pooling
- Window size may be different (2 or more)
- The feature map based on bigrams:

$$c_i = f(\mathbf{w}^T \mathbf{x}_{i:i+h-1} + b)$$



$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$$



What's next?

We need more features!

Based on: Lecture by Richard Socher 5/12/16, http://cs224d.stanford.edu

One layer CNN

- Feature representation is based on some applied filter:

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \in \mathbb{R}^{n-h+1}$$

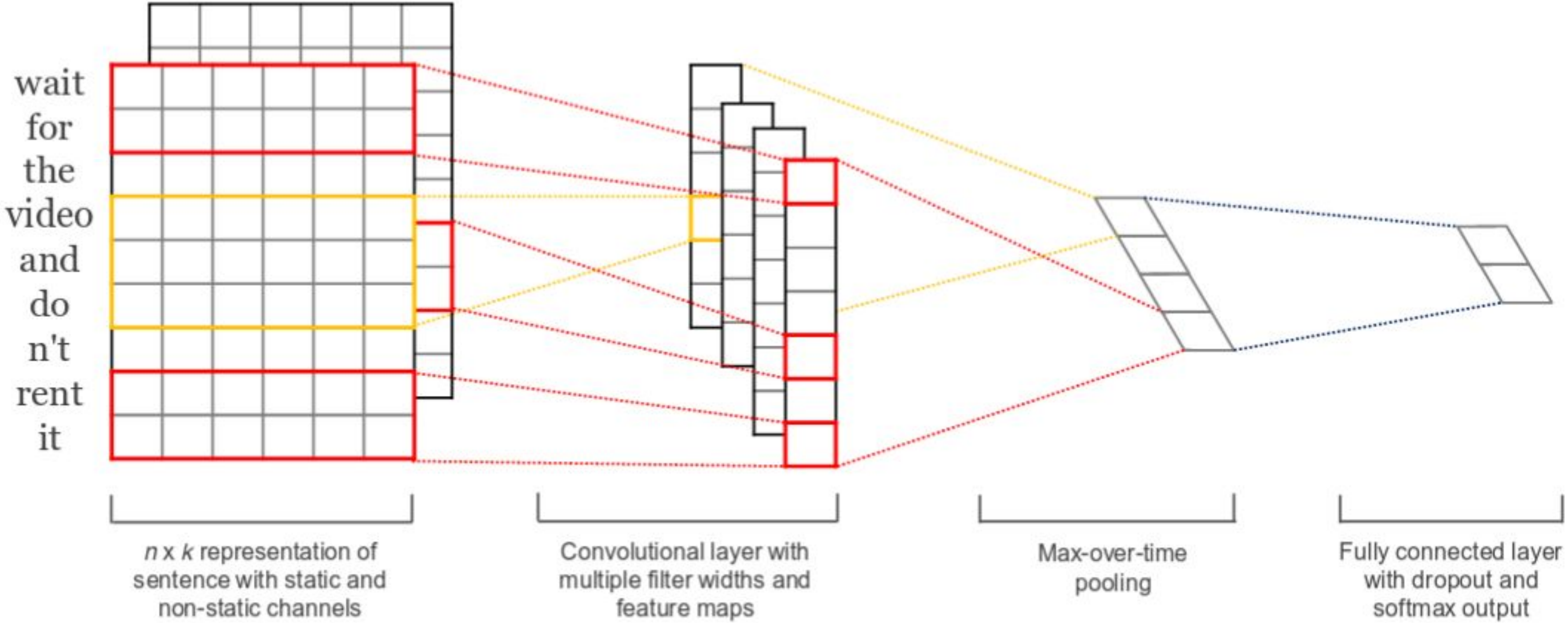
- Let's use pooling:

$$\hat{c} = \max\{\mathbf{c}\}$$

- Now the length of **c** is irrelevant!
- So we can use filters based on unigrams, bigrams, tri-grams, 4-grams, etc.

Based on: Lecture by Richard Socher 5/12/16, http://cs224d.stanford.edu

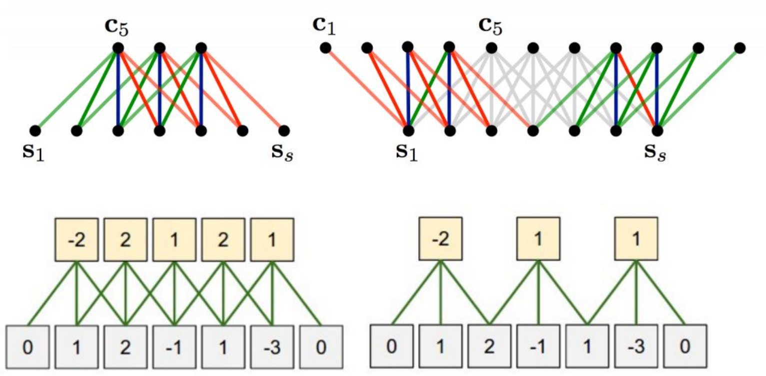
Another example from Kim (2014) paper



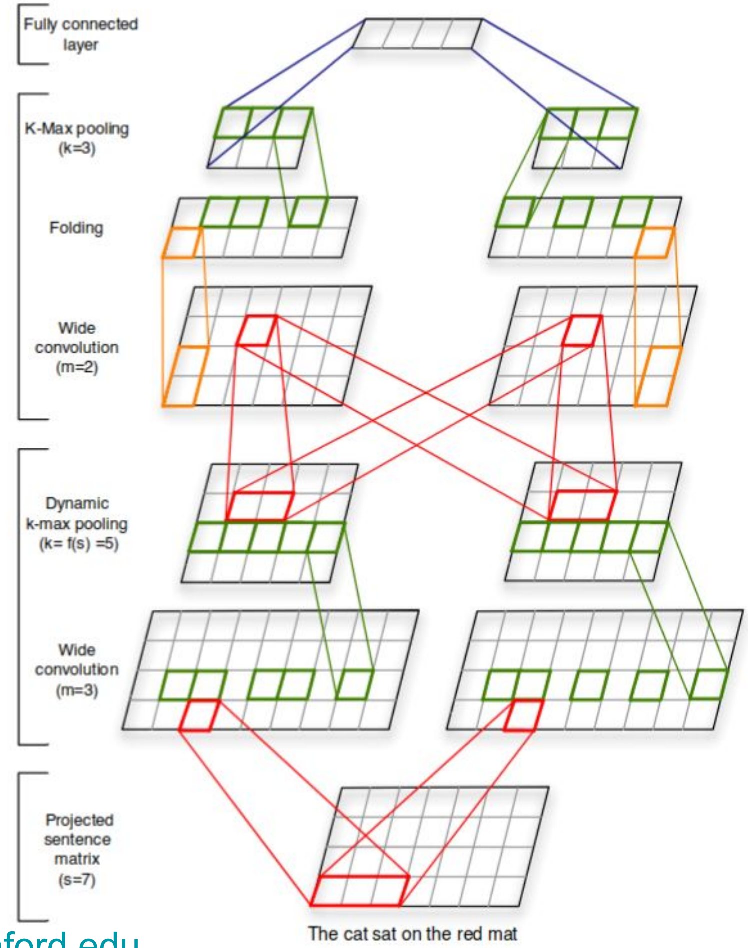
Based on: Lecture by Richard Socher 5/12/16, http://cs224d.stanford.edu

More about CNN

- Narrow vs wide convolution (stride and zero-padding)



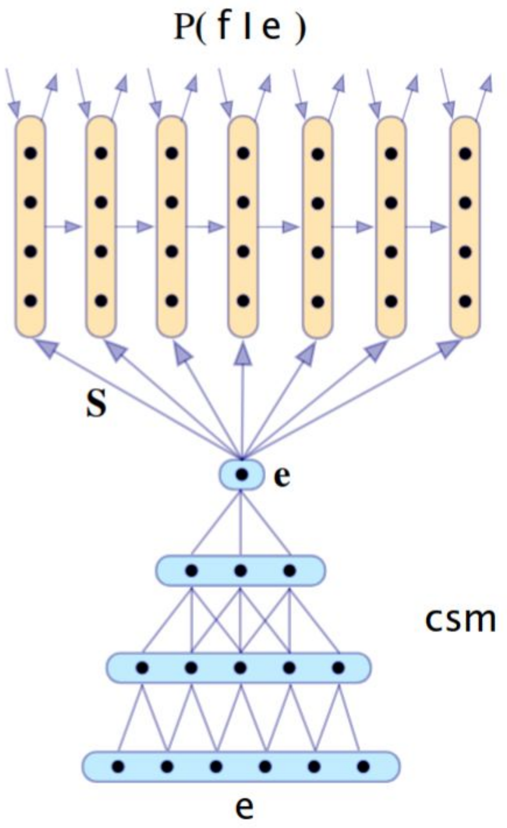
- Complex pooling schemes over sequences
- Great readings (e.g. Kalchbrenner et. al. 2014)



Based on: Lecture by Richard Socher 5/12/16, http://cs224d.stanford.edu

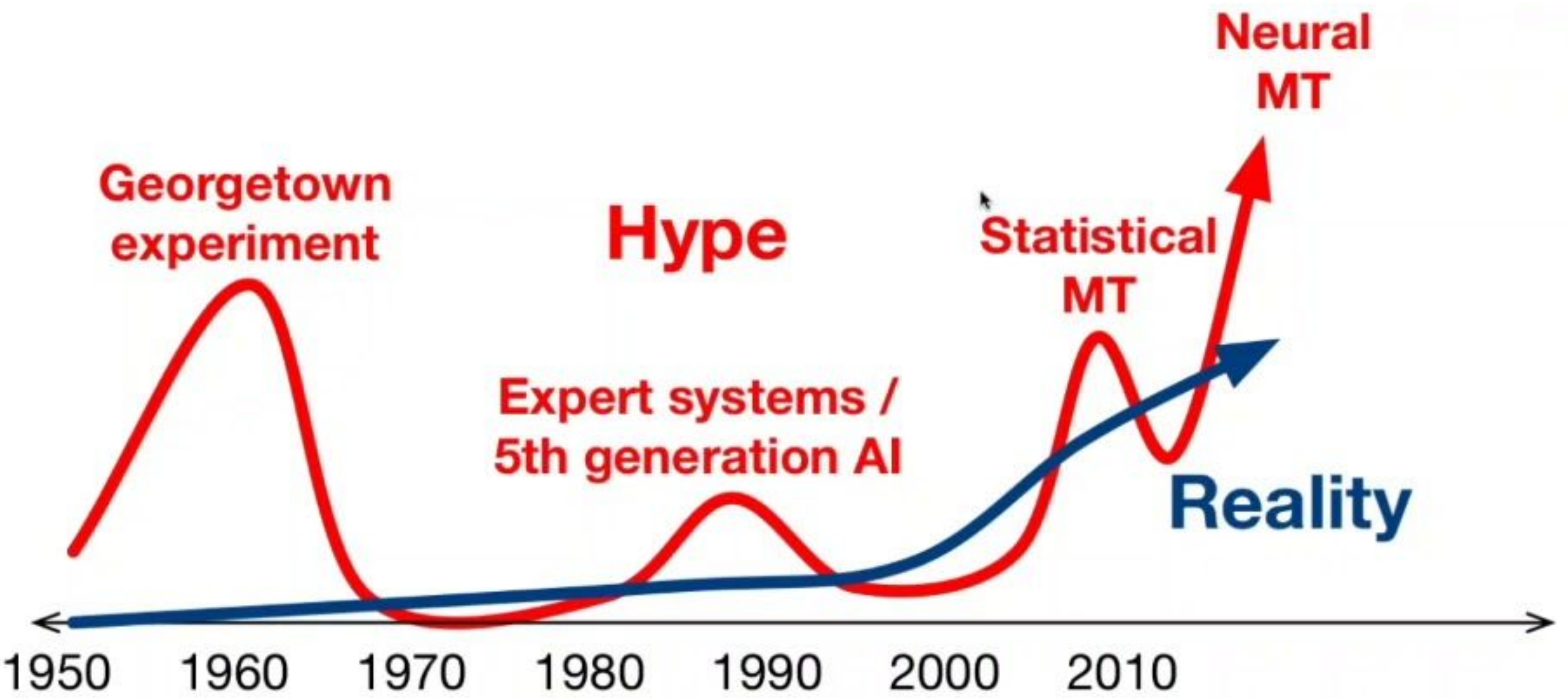
CNN applications

- Neural machine translation: CNN as encoder, RNN as decoder
- One of the first neural machine translation efforts
- Paper: [Recurrent Continuous Translation Models, Kalchbrenner and Blunsom, 2013]



Based on: Lecture by Richard Socher 5/12/16, http://cs224d.stanford.edu

Historical overview



Before Deep Learning

1950s: first Machine Translation

- Georgetown experiment (7 Jan 1954)
  - Automatic Russian-English translation of 60 sentences
  - 250 vocabulary articles
  - 6 grammar rules
  - Calculated on Mainframe IBM 701
- The same experiment in the USSR (1954 too)
  - Rule-based translation
  - Calculated on BESM

1990-2010: Statistical Machine Translation

We want to find best English sentence y, given French sentence x

Let’s use Bayes Rule to break this down into two components:

$$
\begin{aligned}
&\operatorname{argmax}_y P(y|x) \\
&= \operatorname{argmax}_y \underbrace{P(x|y)}_{\text{Translation Model}} \underbrace{P(y)}_{\text{Language Model}}
\end{aligned}
$$

Translation Model

Models how words and phrases should be translated (*fidelity*).
Learnt from parallel data.

Language Model

Models how to write good English (*fluency*).
Learnt from monolingual data.

Source: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

1990-2010: Statistical Machine Translation

How to learn translation model from the parallel corpus?

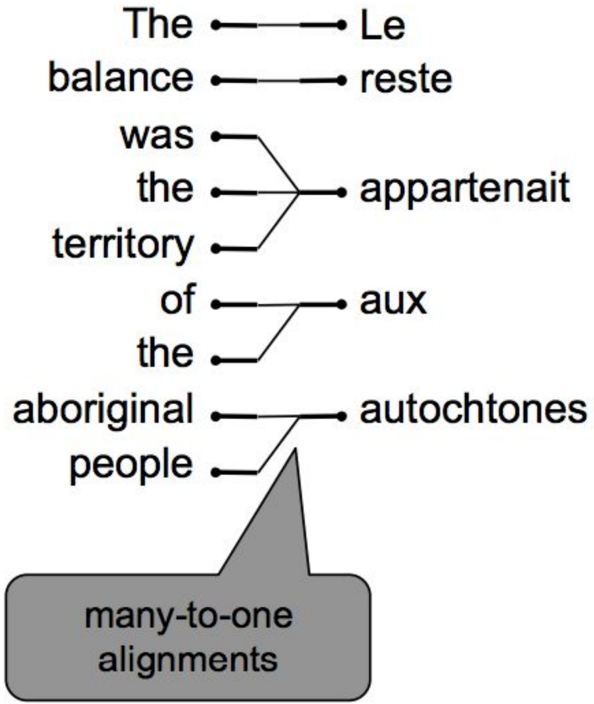
Let's calculate

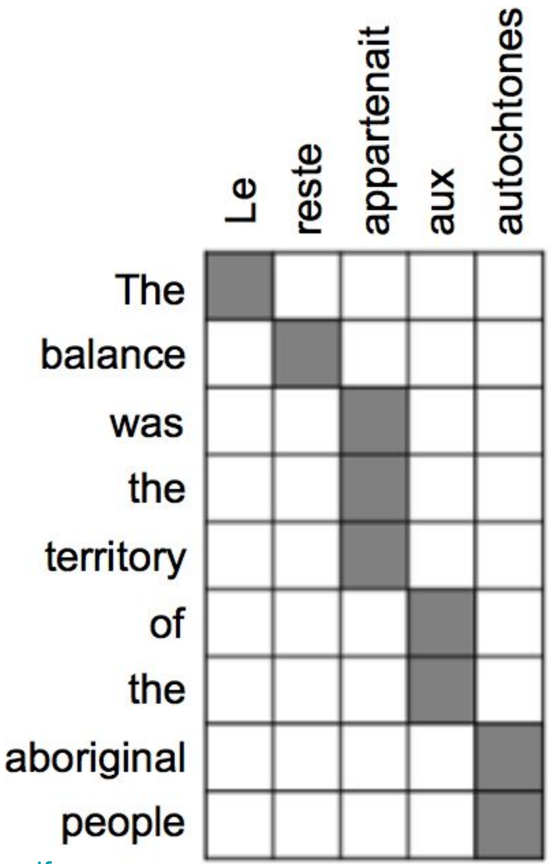
$$P(x, a|y)$$

Where **a** is an **alignment** (word-level correspondence between French sentence x and English sentence y)

1990-2010: Statistical Machine Translation

Alignment can be: **many-to-one**

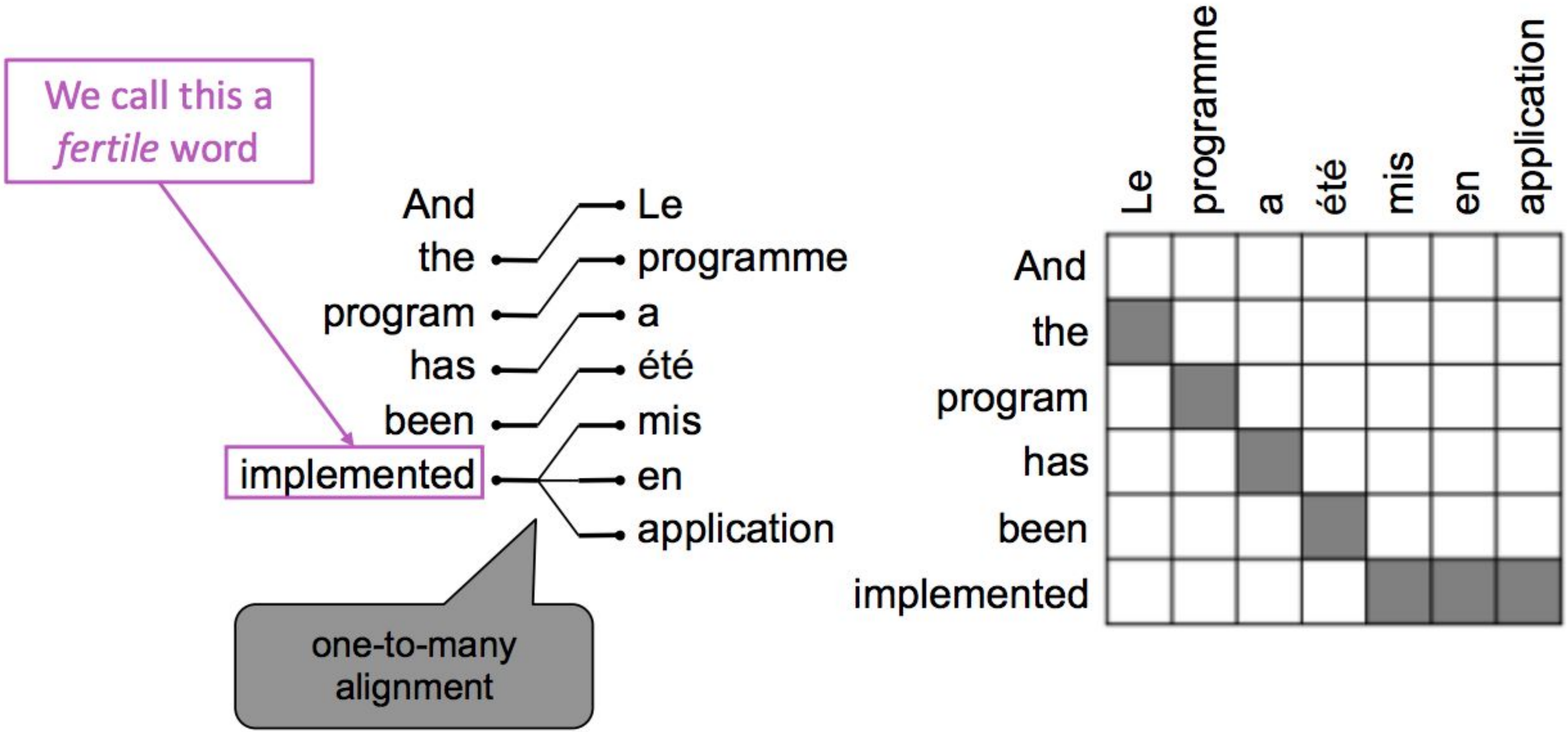




Source: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

1990-2010: Statistical Machine Translation

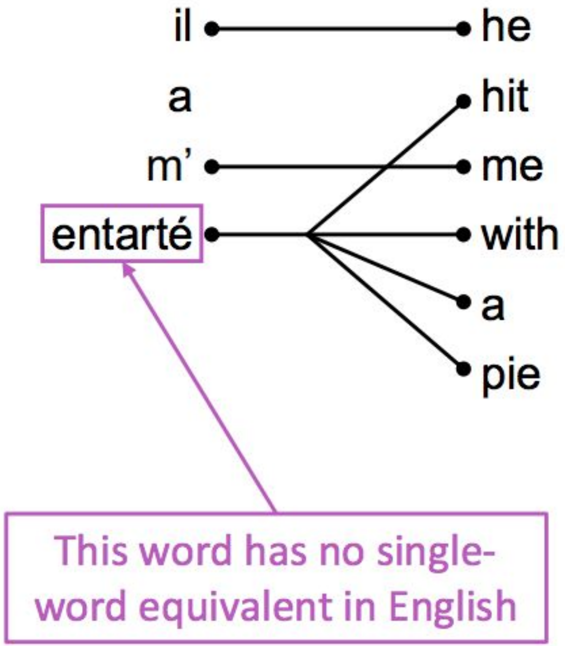
Alignment can be: **one-to-many**

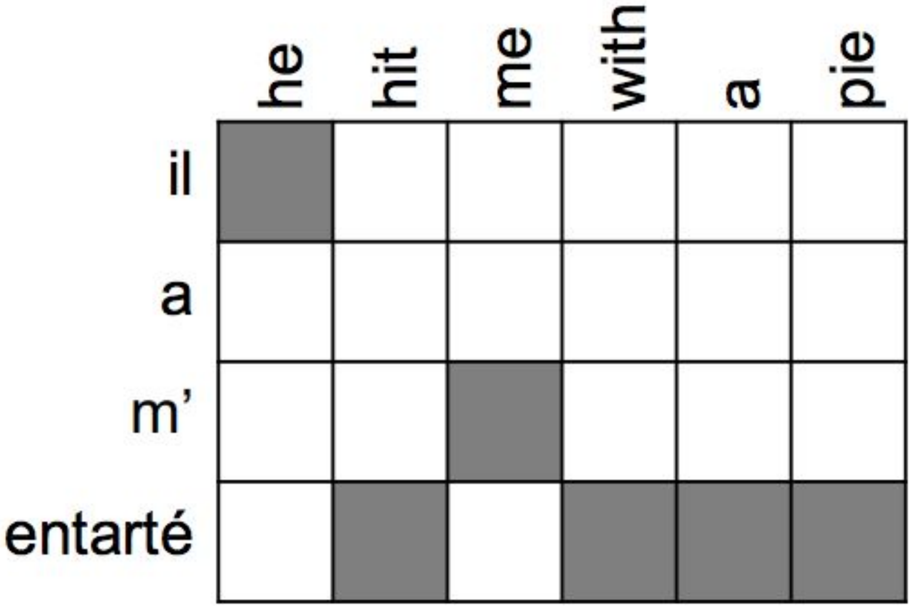


Source: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

1990-2010: Statistical Machine Translation

Some words are very fertile!

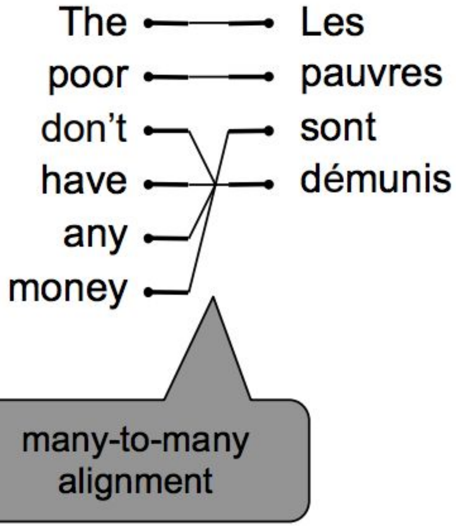


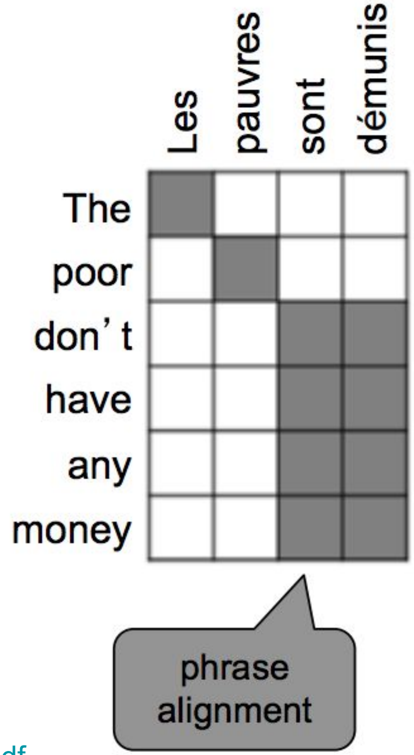


Source: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

1990-2010: Statistical Machine Translation

Alignment can be: **many-to-many**





Source: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

1990-2010: Statistical Machine Translation

$$\text{argmax}_y \underbrace{P(x|y)}_{\text{Translation Model}} \underbrace{P(y)}_{\text{Language Model}}$$

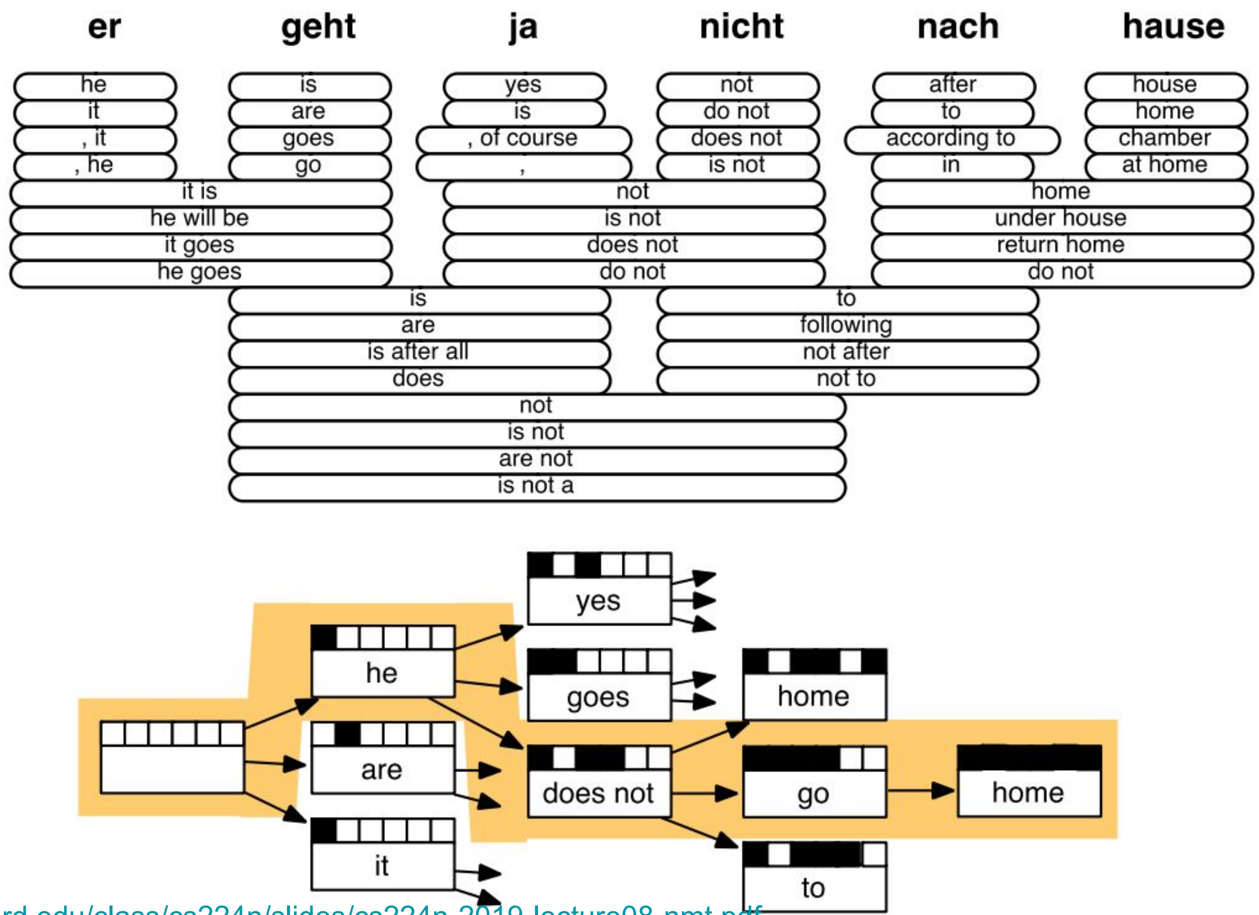
Question:
How to compute this argmax?

Enumerate every possible y and calculate the probability? No!

Use a heuristic search algorithm to search for the best translation, discarding hypotheses that are too low-probability

Source: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

1990-2010: Statistical Machine Translation



Source: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

1990-2010: Statistical Machine Translation

- Systems had many separately-designed subcomponents
- Lots of feature engineering
- Need to design features to capture particular language phenomena
- Require compiling and maintaining extra resources (tables of equivalent phrases)
- Lots of human effort to maintain
- Repeated effort for each language pair!

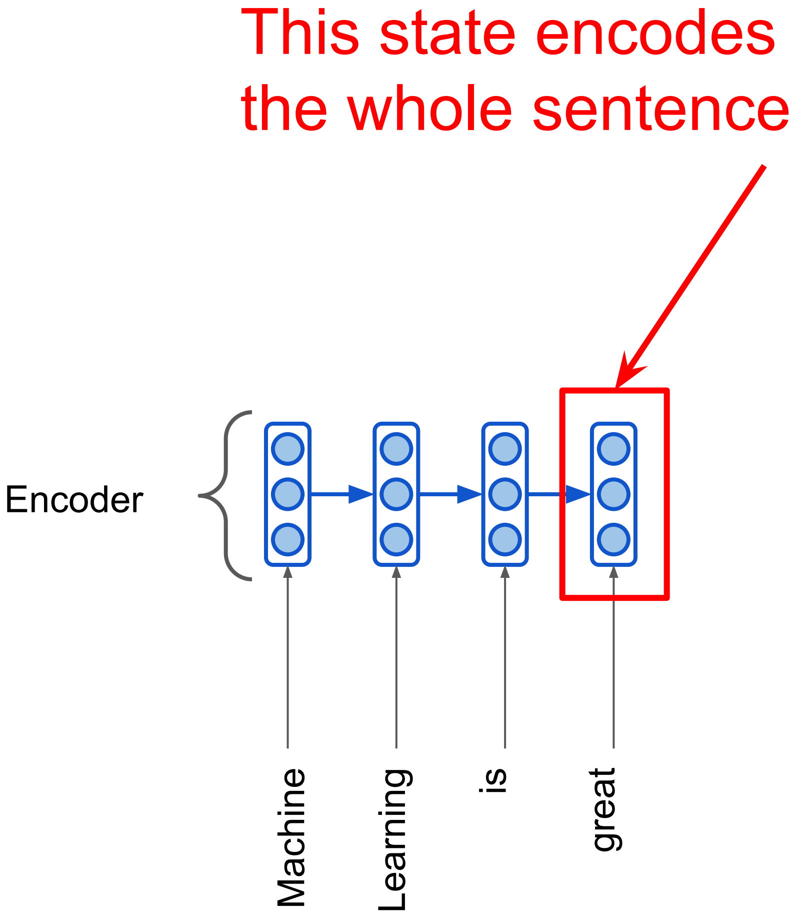
Neural Machine Translation

What is Neural Machine Translation?

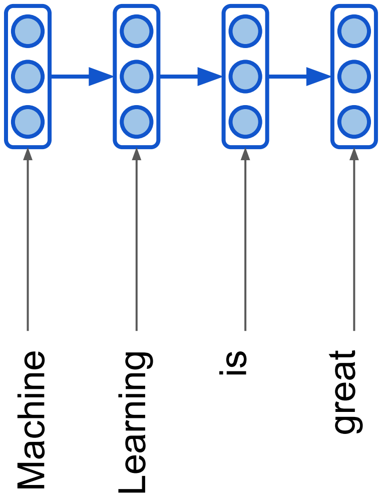
- Neural Machine Translation (NMT) is a way to do Machine Translation with a single neural network

- The neural network architecture is called sequence-to-sequence (aka **seq2seq**), it involves two **RNNs**

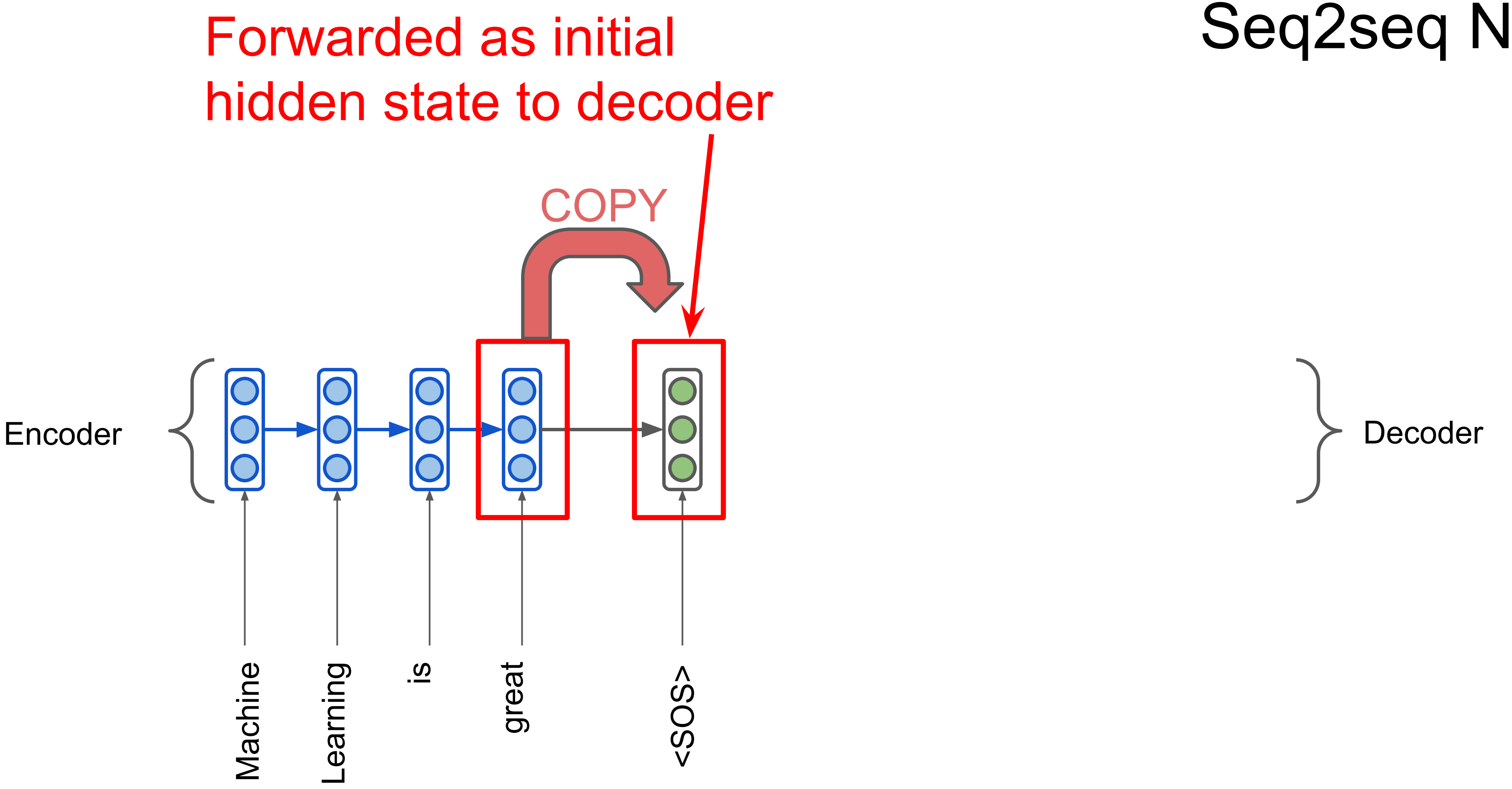
Seq2seq NMT



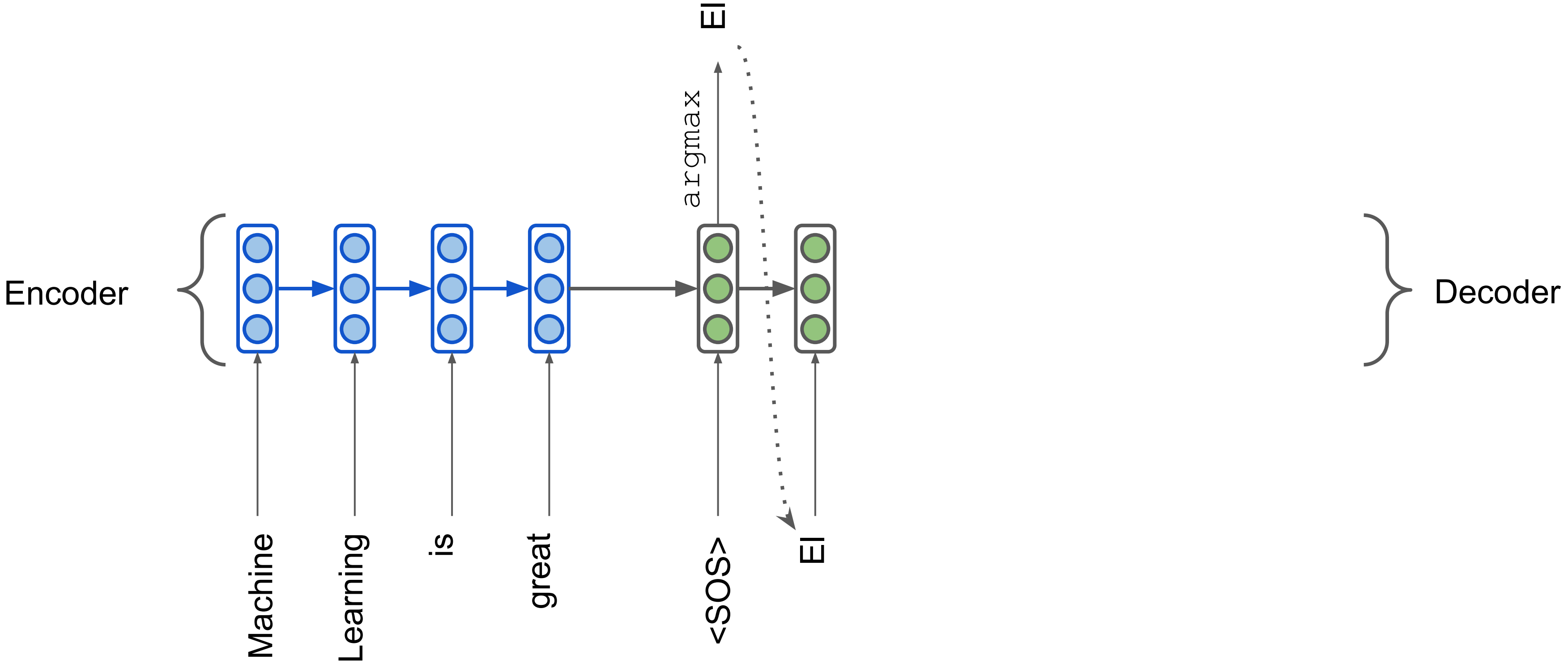
This state encodes the whole sentence
Encoder
Machine
Learning
is
great


Machine
Learning
is
great

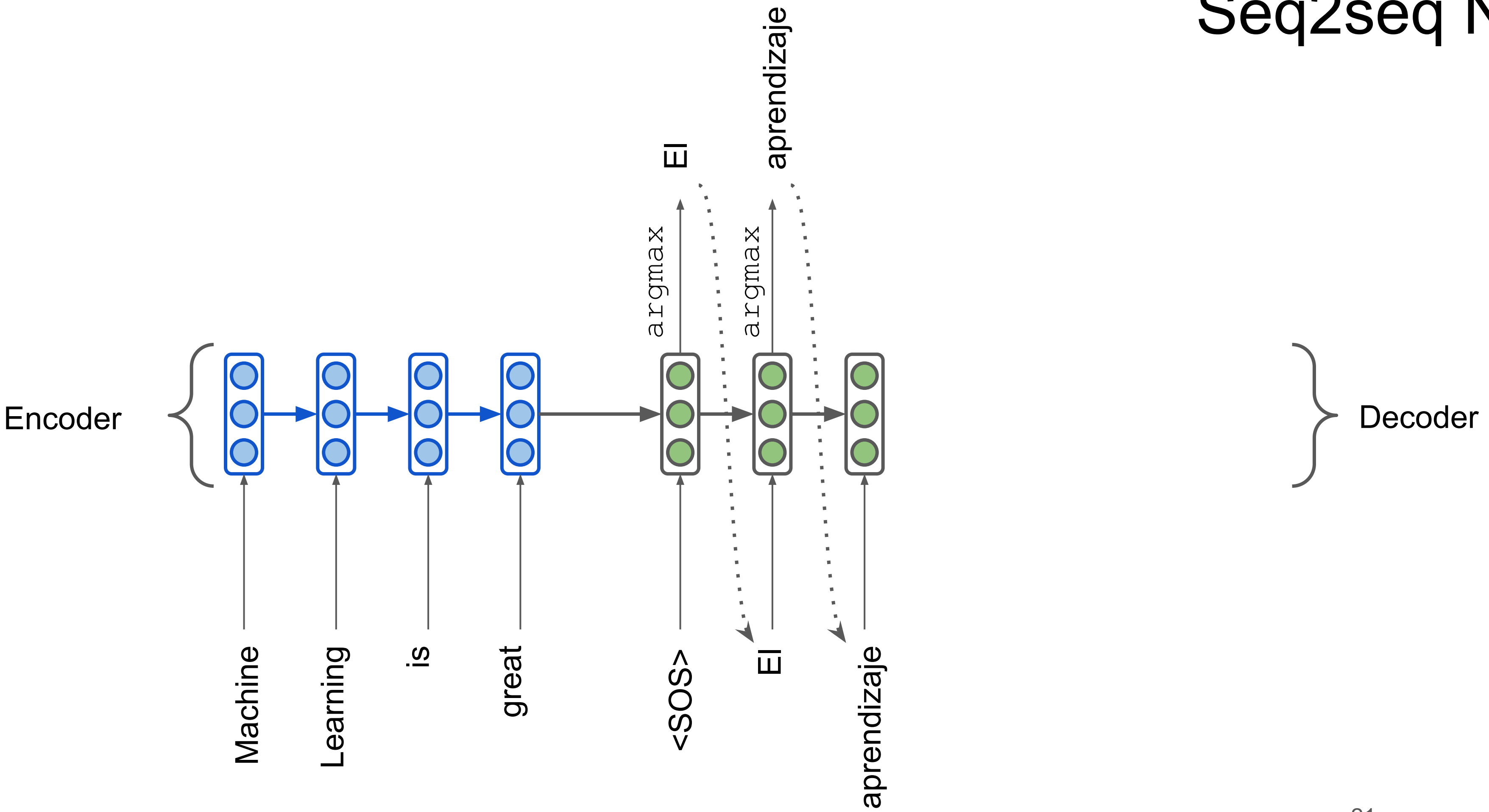
Seq2seq NMT


Forwarded as initial hidden state to decoder
COPY
Encoder
Decoder
Machine
Learning
is
great
<SOS>

Seq2seq NMT

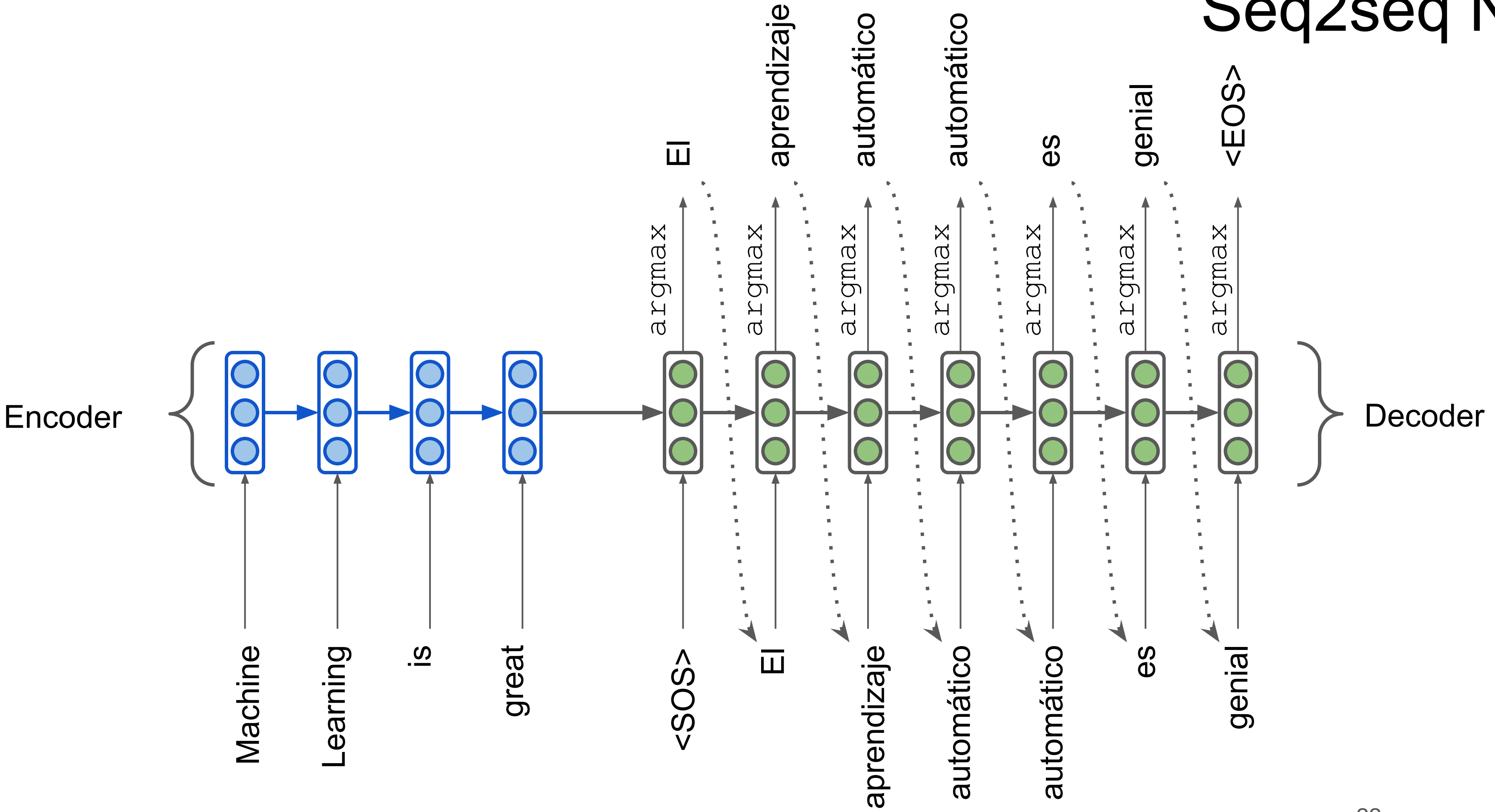


Seq2seq NMT



Encoder
Decoder
Machine
Learning
is
great
<SOS>
El
aprendizaje
argmax
argmax
El
aprendizaje

Seq2seq NMT



Encoder
Decoder
Machine
Learning
is
great
<SOS>
El
aprendizaje
automático
automático
es
genial
argmax
argmax
argmax
argmax
argmax
argmax
argmax
El
aprendizaje
automático
automático
es
genial
<EOS>

NMT: how does it work?

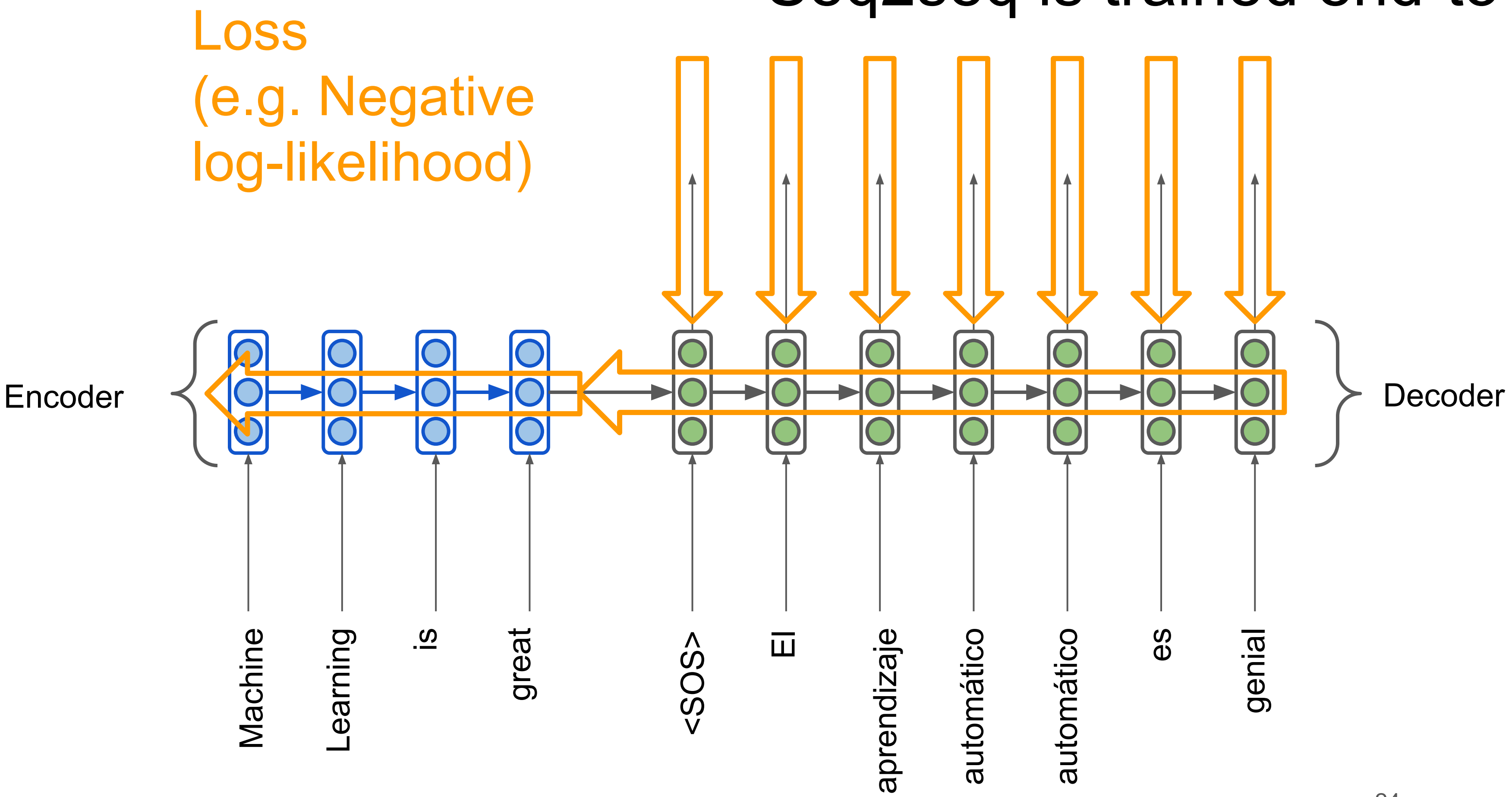
- NMT directly calculates P(y|x)
  - y – target sentence, x – source sentence

$$P(y|x) = P(y_2|y_1, x)P(y_3|y_1, y_2, x)\ldots\underbrace{P(y_T|y_1, y_2, \ldots, x)}$$

Probability of next word
in target language

- To train it we need a huge parallel corpus.

Seq2seq is trained end-to-end



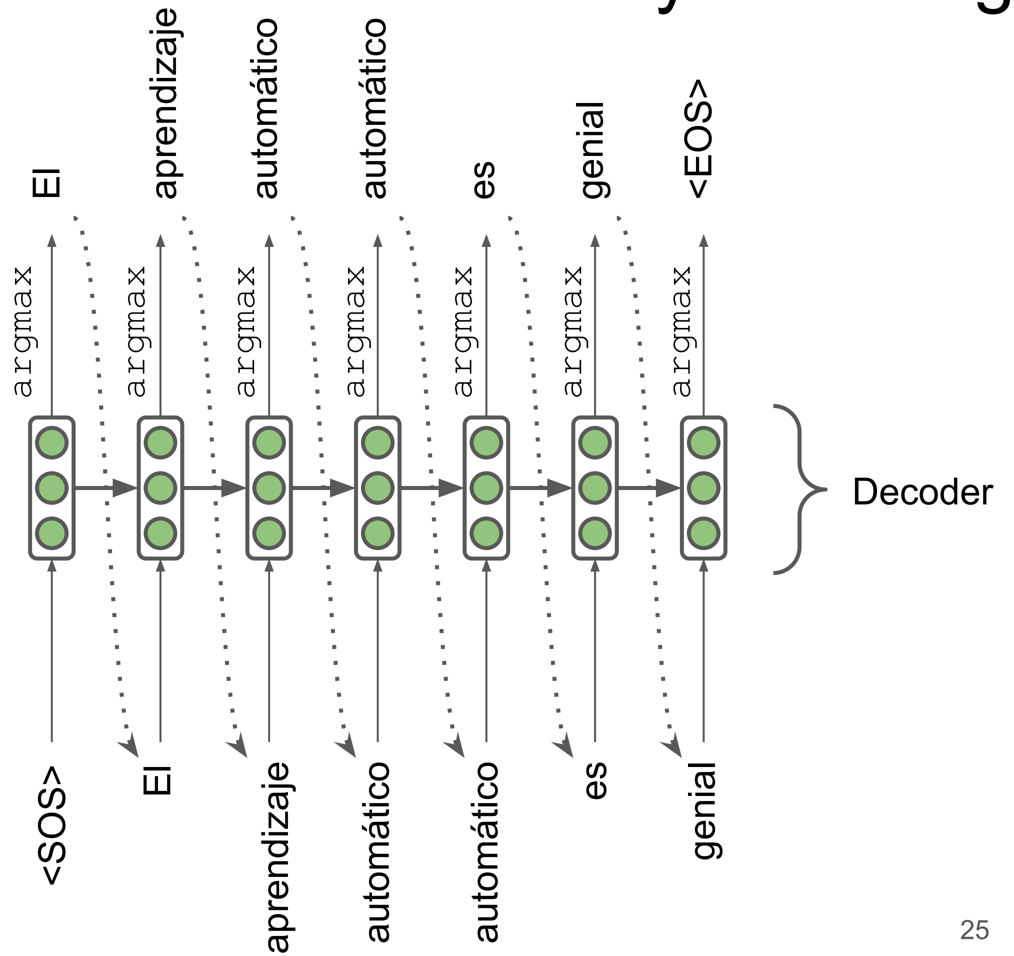
Greedy decoding

- Decoder predicts the most probable token (argmax) on each step

- The approach is **greedy**

Any problems with it?

Any mistake is treated as input on the next step!



Exhaustive search

- We want the translation that maximizes the likelihood:

$$P(y|x) = P(y_1|x) \prod_{t=2}^{T} P(y_t|y_1, \ldots y_{t-1}, x)$$

- We cannot compute all the possible sequences (exponential complexity)

Beam search

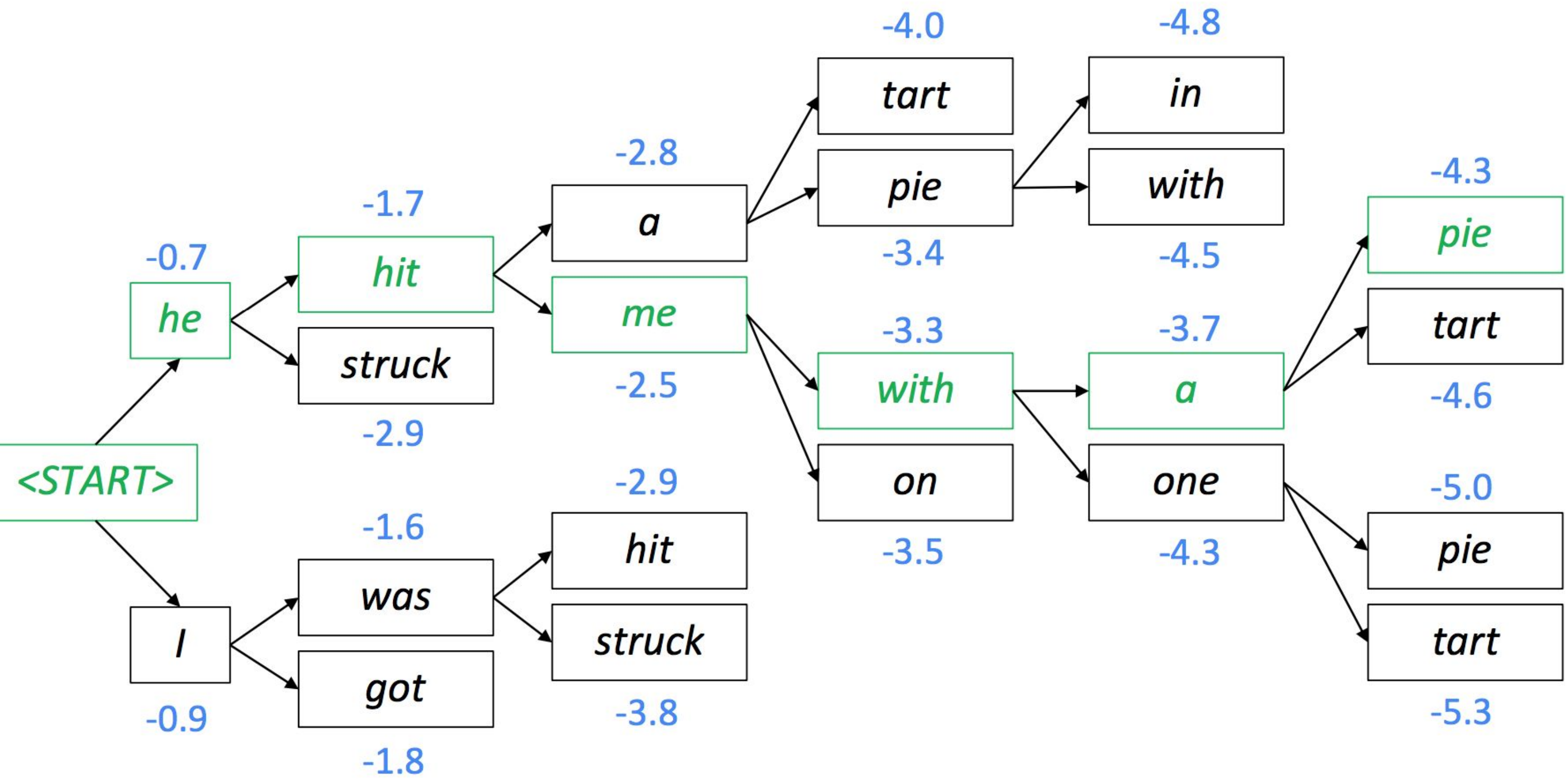
- On each step of decoder, keep track of the k most probable partial translations (which we call hypotheses)
- k is the beam size (in practice around 5 to 10)
- A hypothesis has a score which is its log probability:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

- We search for high-scoring hypotheses, tracking top k on each step
- Beam search does not guarantee finding optimal solution

Beam search decoding: example

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Source: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Beam search decoding: stopping criterion

- In **greedy decoding**, usually we decode until the model produces <EOS> token
- In **beam search decoding**, different hypotheses may produce <EOS> tokens on different timesteps
  - When a hypothesis produces <EOS>, that hypothesis is complete.
  - Place it aside and continue exploring other hypotheses via beam search.
- Usually we continue beam search until:
  - We reach pre-defined timestep T
  - We have at least n completed hypotheses

Beam search decoding: finishing up

- How to select top one with highest score?
- Each hypothesis on our list has a score:

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

- **Problems?**

Longer hypotheses have lower scores

- **Fix:** Normalize by length. Use this to select top one instead:

$$\frac{1}{t} \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

NMT: Quality Evaluation

BLEU (Bilingual Evaluation Understudy) compares the machine-written translation to human-written translation, and computes a similarity score based on:

- n-gram precision
- penalty for too-short system translations (brevity penalty)

$$BLEU = \text{brevity penalty} \cdot \left(\prod_{i=1}^{n} \text{precision}_i \right)^{1/n} \cdot 100\%$$

$$\text{brevity penalty} = min\left(1, \frac{\text{output length}}{\text{reference length}}\right)$$

BLEU

BLEU (Bilingual Evaluation Understudy) compares the machine-written translation to human-written translation, and computes a similarity score based on:

- n-gram precision
- brevity penalty

SYSTEM A: Israeli officials responsibility of airport safety
2-GRAM MATCH 1-GRAM MATCH

REFERENCE: Israeli officials are responsible for airport security

SYSTEM B: airport security Israeli officials are responsible
2-GRAM MATCH 4-GRAM MATCH

Metric	System A	System B
precision (1gram)	3/6	6/6
precision (2gram)	1/5	4/5
precision (3gram)	0/4	2/4
precision (4gram)	0/3	1/3
brevity penalty	6/7	6/7
BLEU	0%	52%

$$BLEU = \text{brevity penalty} \cdot \left(\prod_{i=1}^{n} \text{precision}_i \right)^{1/n} \cdot 100\%$$

BLEU is imperfect:

- There are many valid ways to translate a sentence
- So a good translation may get a poor BLEU score just because of low n-gram overlap with the human translation

Other ways to estimate translation quality

- **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation)
- **METEOR** (Metric for Evaluation of Translation with Explicit ORdering)
  - Uses synonyms from WordNet
- **NIST** (от US National Institute of Standards and Technology)
  - More weight to rare n-grams, less punishment for short texts
- **TER**
  - Uses the number of changes that should be made to get to the reference translation

NMT: advantages

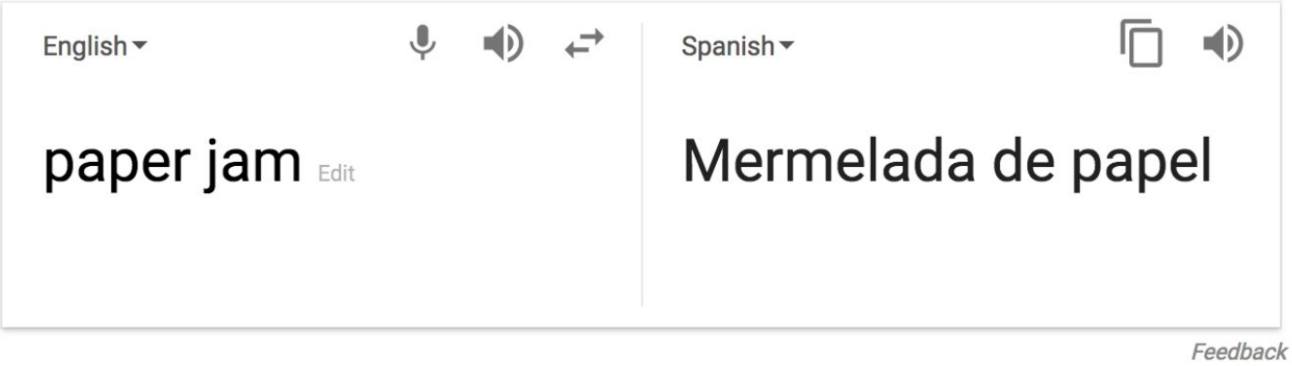
- Better performance
  - More fluent
  - Better use of context
  - Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No subcomponents to be individually optimized
- Requires much less human engineering effort
  - No feature engineering
  - Same method for all language pairs

NMT: disadvantages

- NMT is less interpretable
  - Hard to debug

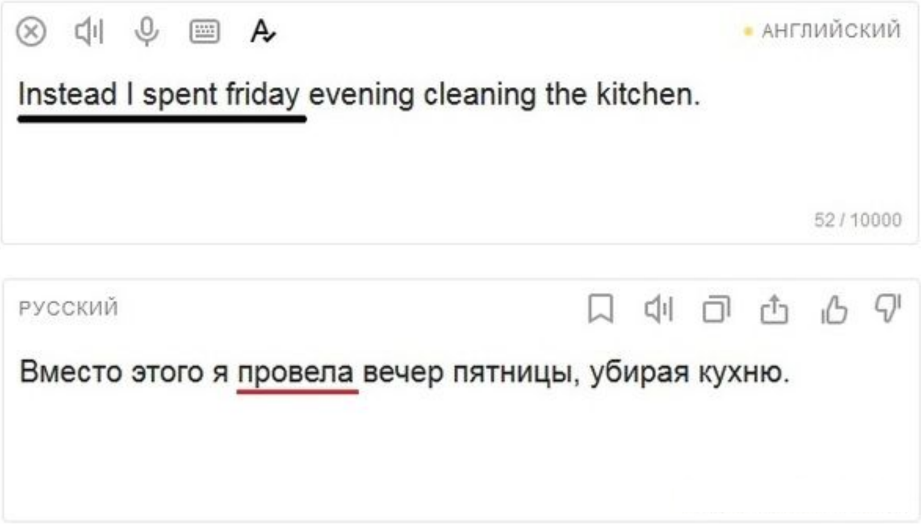
- NMT is difficult to control
  - For example, can't easily specify rules or guidelines for translation
  - Safety concerns!

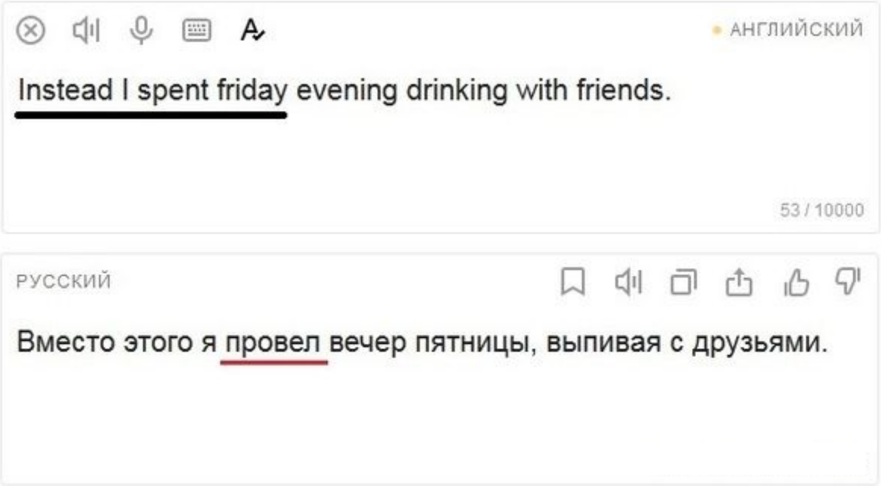
NMT: disadvantages



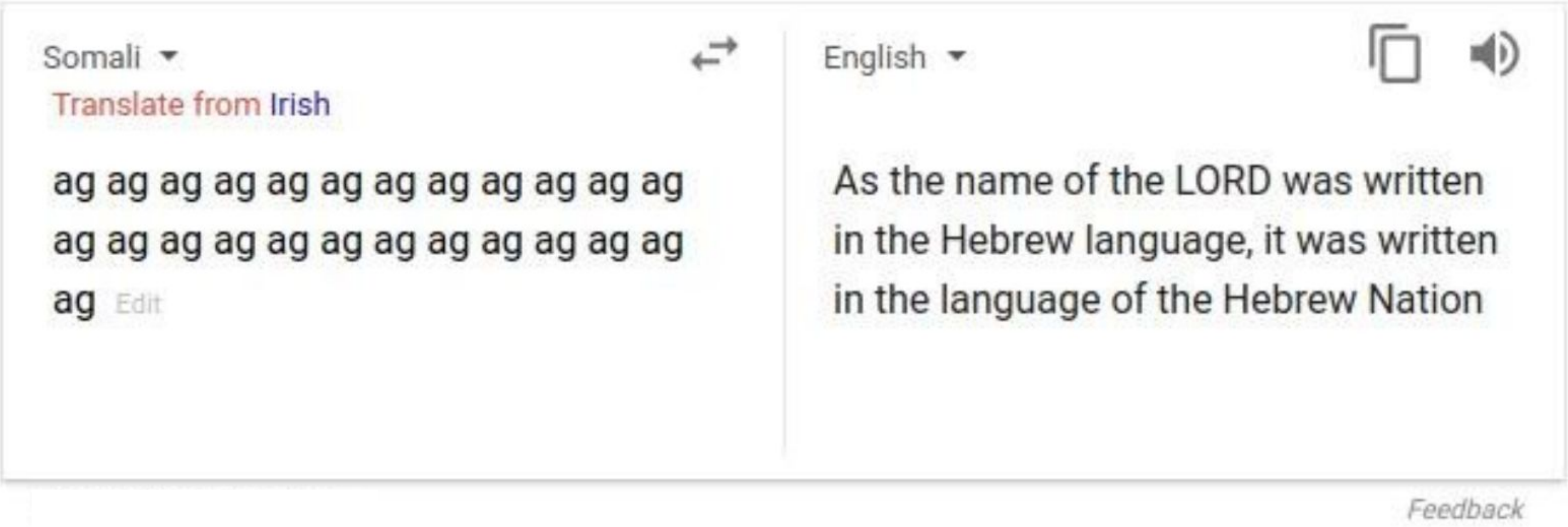
?

NMT: disadvantages

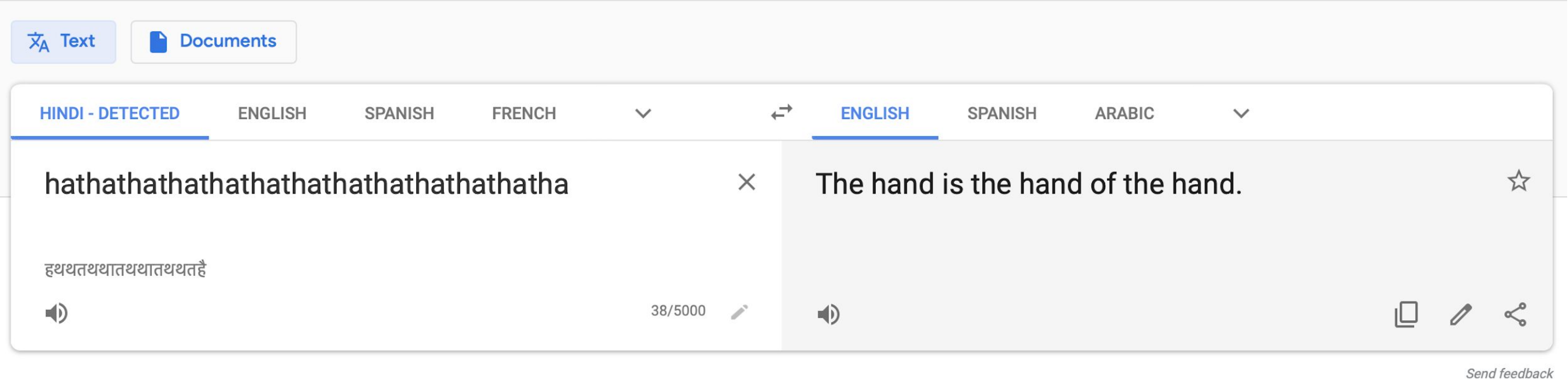




NMT: disadvantages



NMT: disadvantages



Is Machine Translation solved?

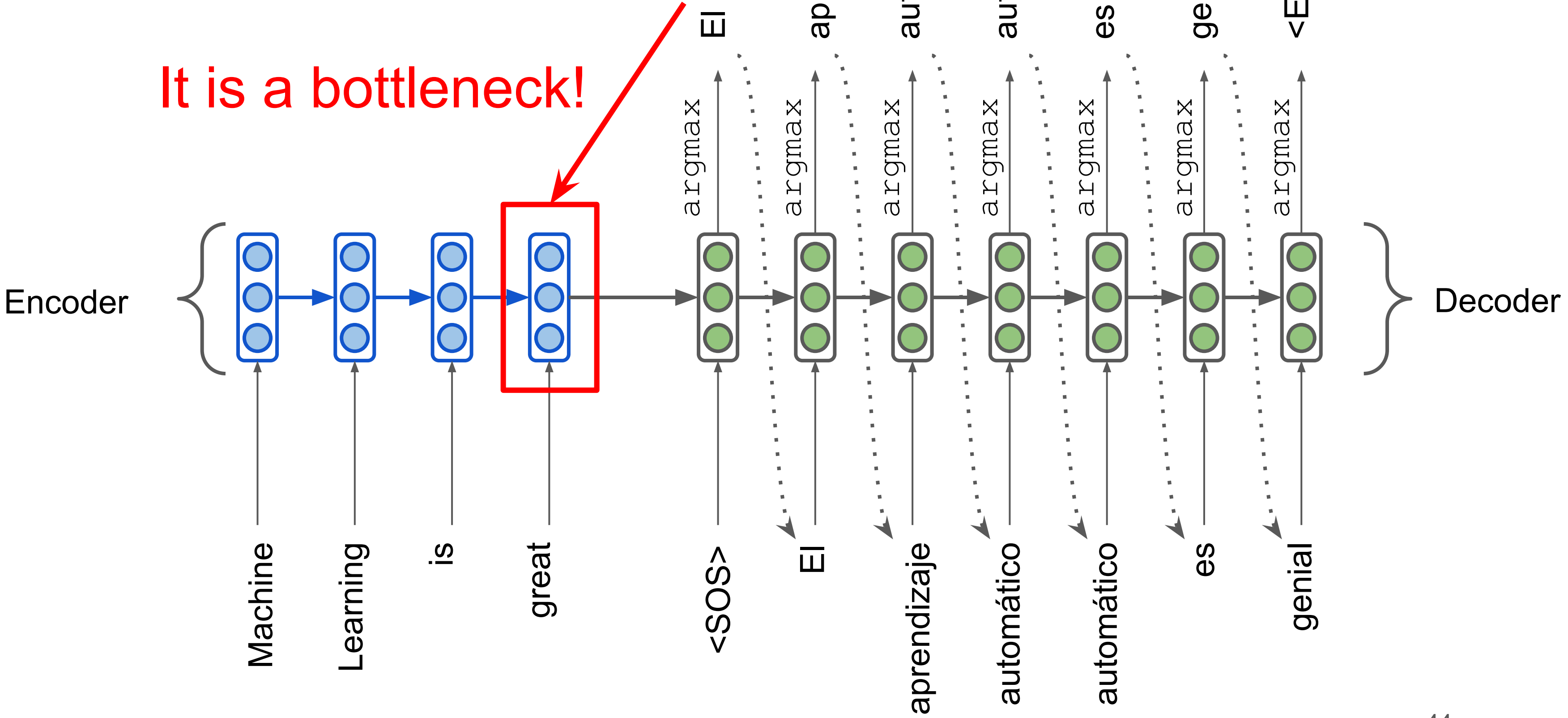
- Many difficulties remain:
  - Out-of-vocabulary words
  - Domain mismatch between train and test data
  - Maintaining context over long texts
  - Low-resource language pairs (no big parallel corpora)

Attention

Seq2seq NMT

This state encodes the whole sentence

It is a bottleneck!



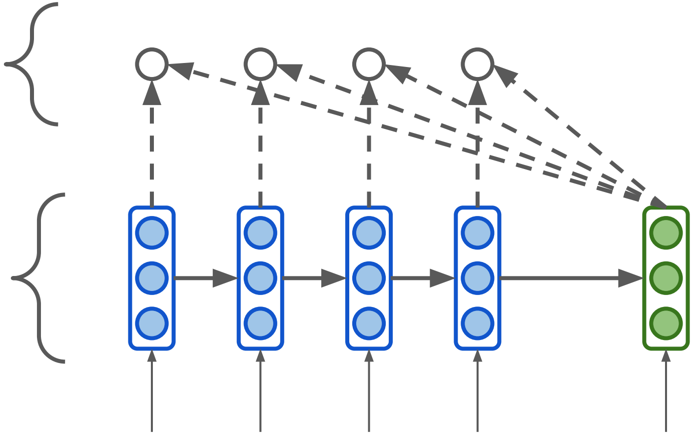
Main idea:

on each step of the **decoder**, use **direct connection to the encoder** to focus on a particular part of the source sequence

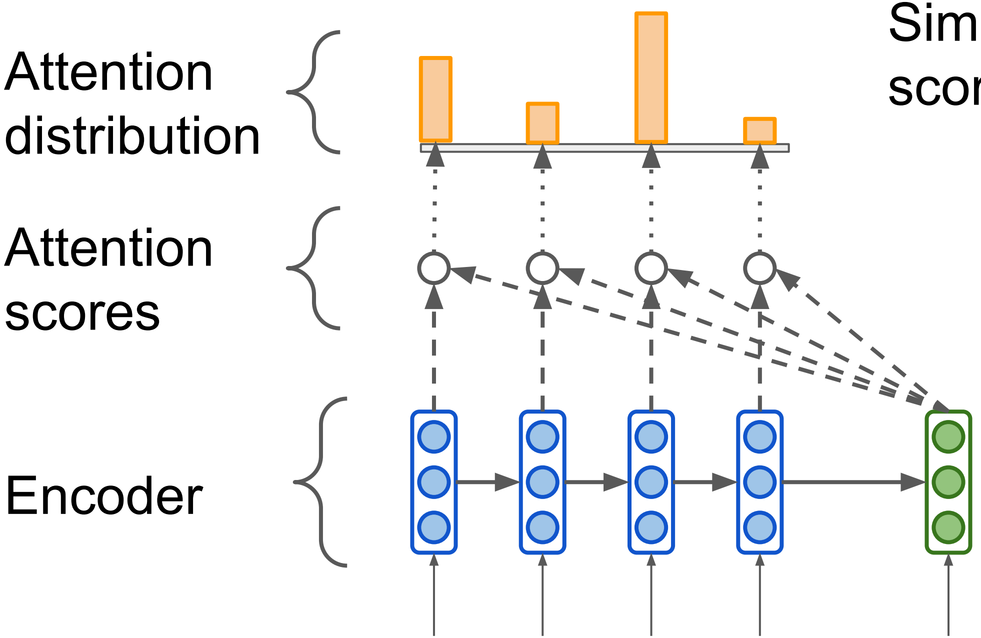
Seq2seq with attention

Attention scores

Encoder

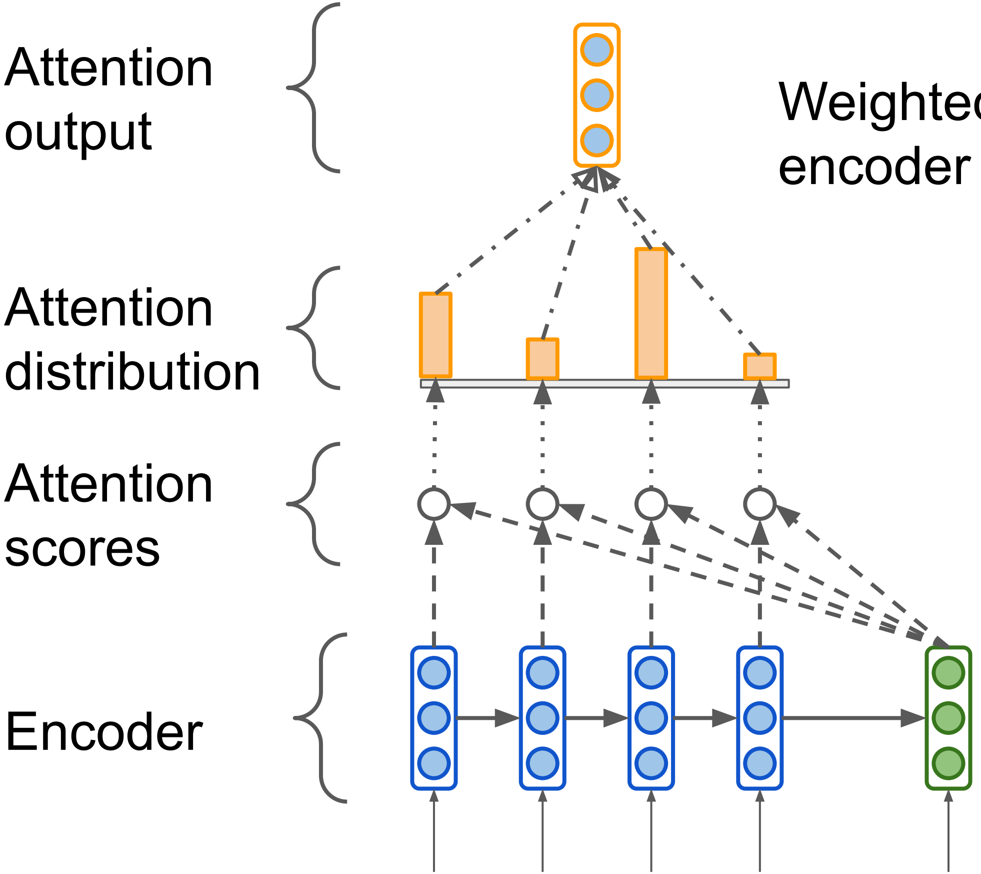


Seq2seq with attention



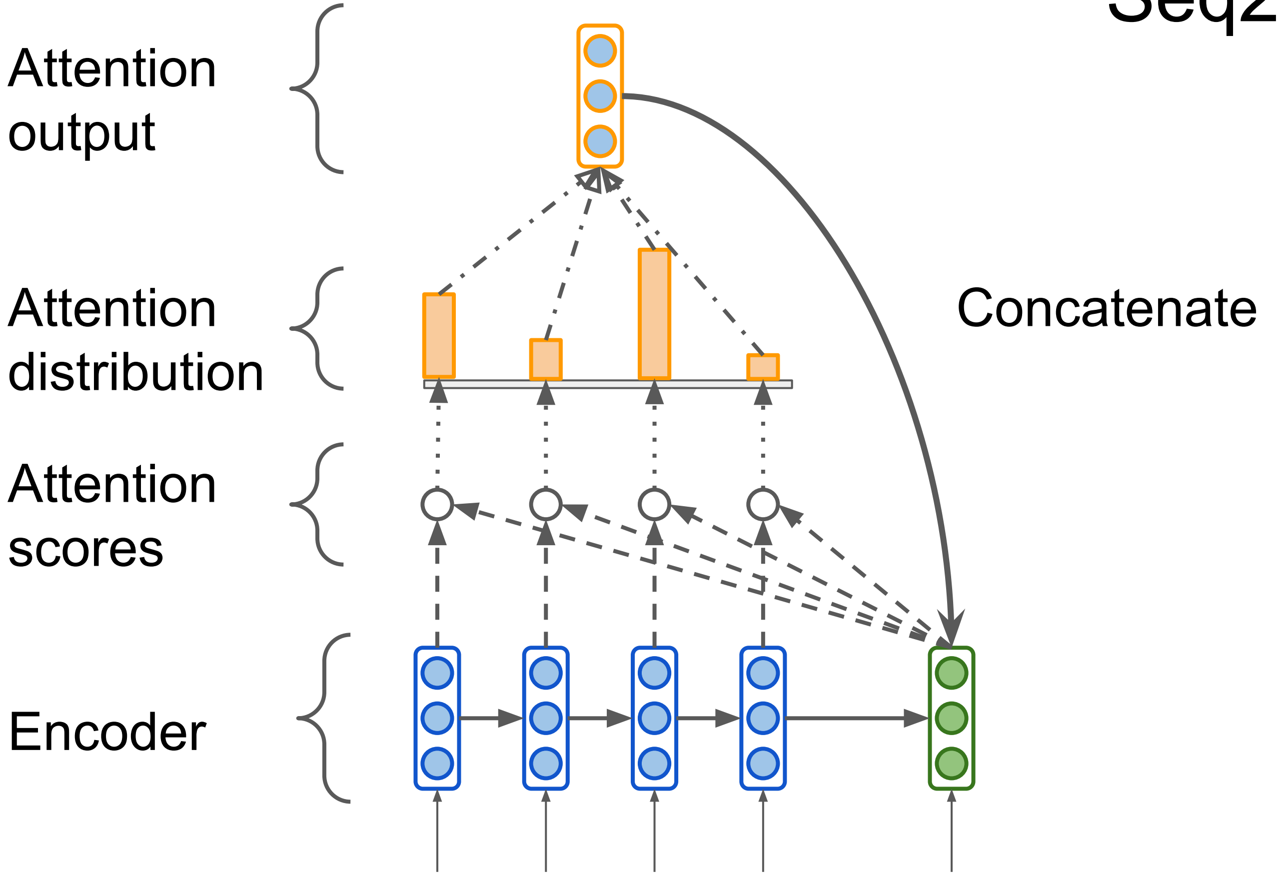
Simply apply softmax to scores

Seq2seq with attention

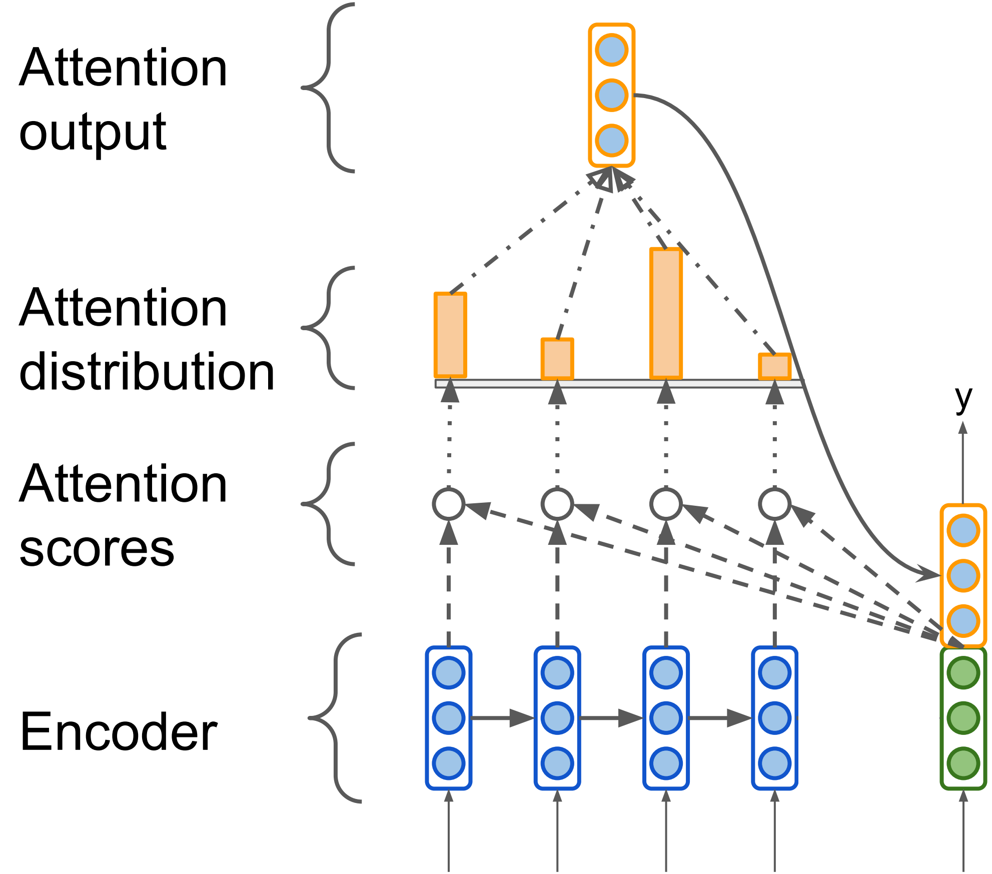


Weighted sum of all encoder states

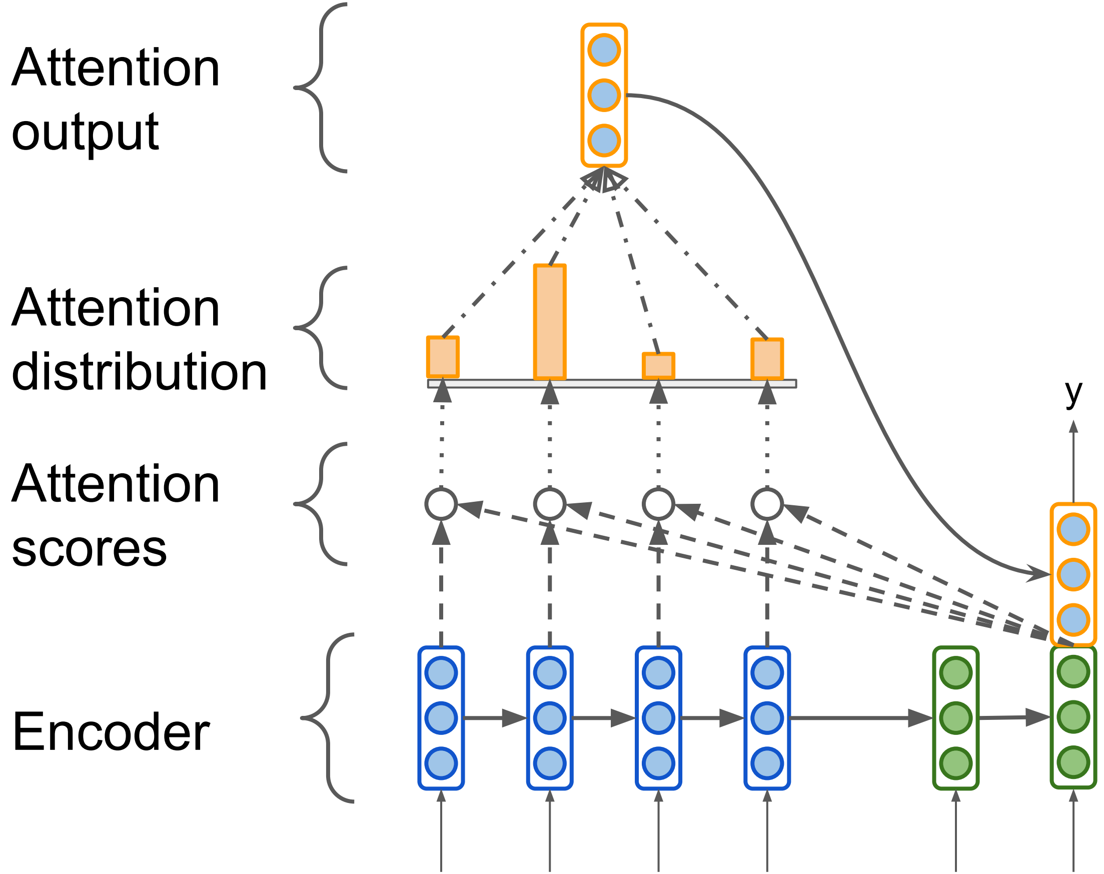
Seq2seq with attention



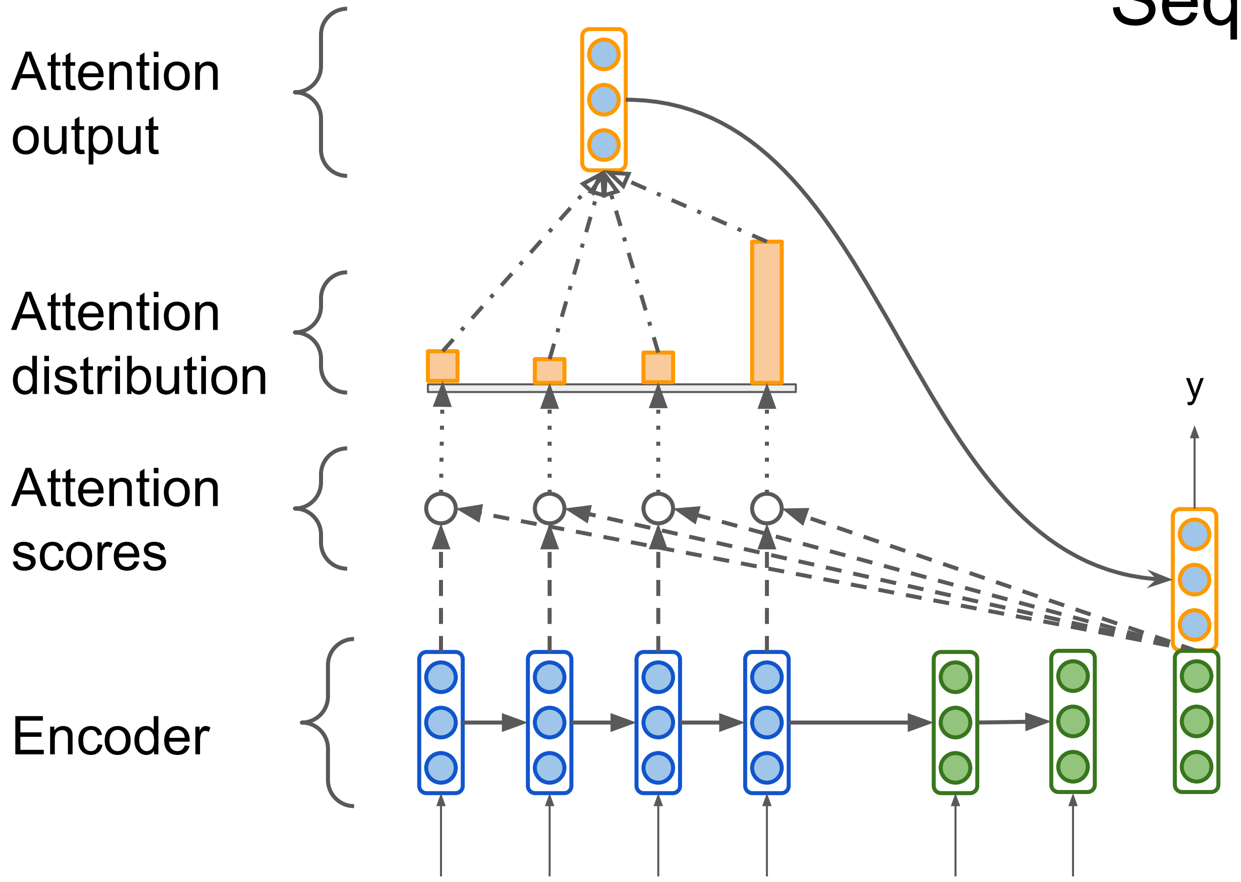
Seq2seq with attention



Seq2seq with attention



Seq2seq with attention



Attention in equations

Denote encoder hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_N \in \mathbb{R}^k$

and decoder hidden state at time step t $\mathbf{s}_t \in \mathbb{R}^k$

The attention scores $\mathbf{e}^t$ can be computed as dot product

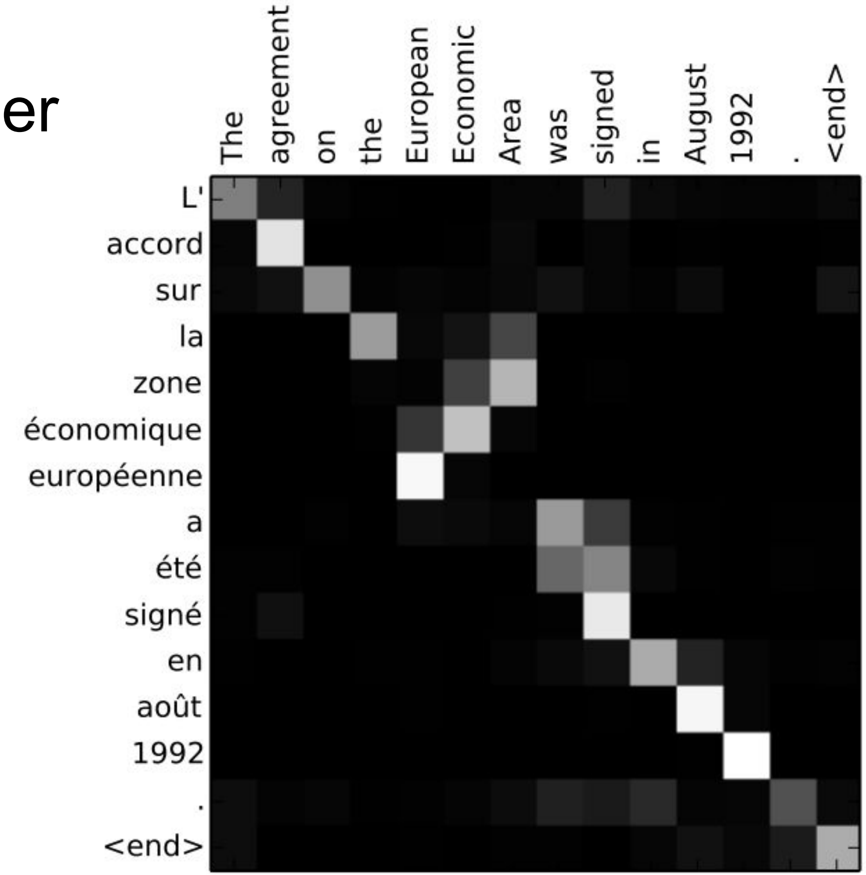
$$\mathbf{e}^t = [\mathbf{s}^T\mathbf{h}_1, \ldots, \mathbf{s}^T\mathbf{h}_N]$$

Then the attention vector is a linear combination of encoder states

$$\mathbf{a}_t = \sum_{i=1}^{N} \boldsymbol{\alpha}_i^t \mathbf{h}_i \in \mathbb{R}^k \text{, where } \boldsymbol{\alpha}_t = \mathrm{softmax}(\mathbf{e}_t)$$

Attention provides interpretability

- We may see what the decoder was focusing on
- We get word alignment for free!

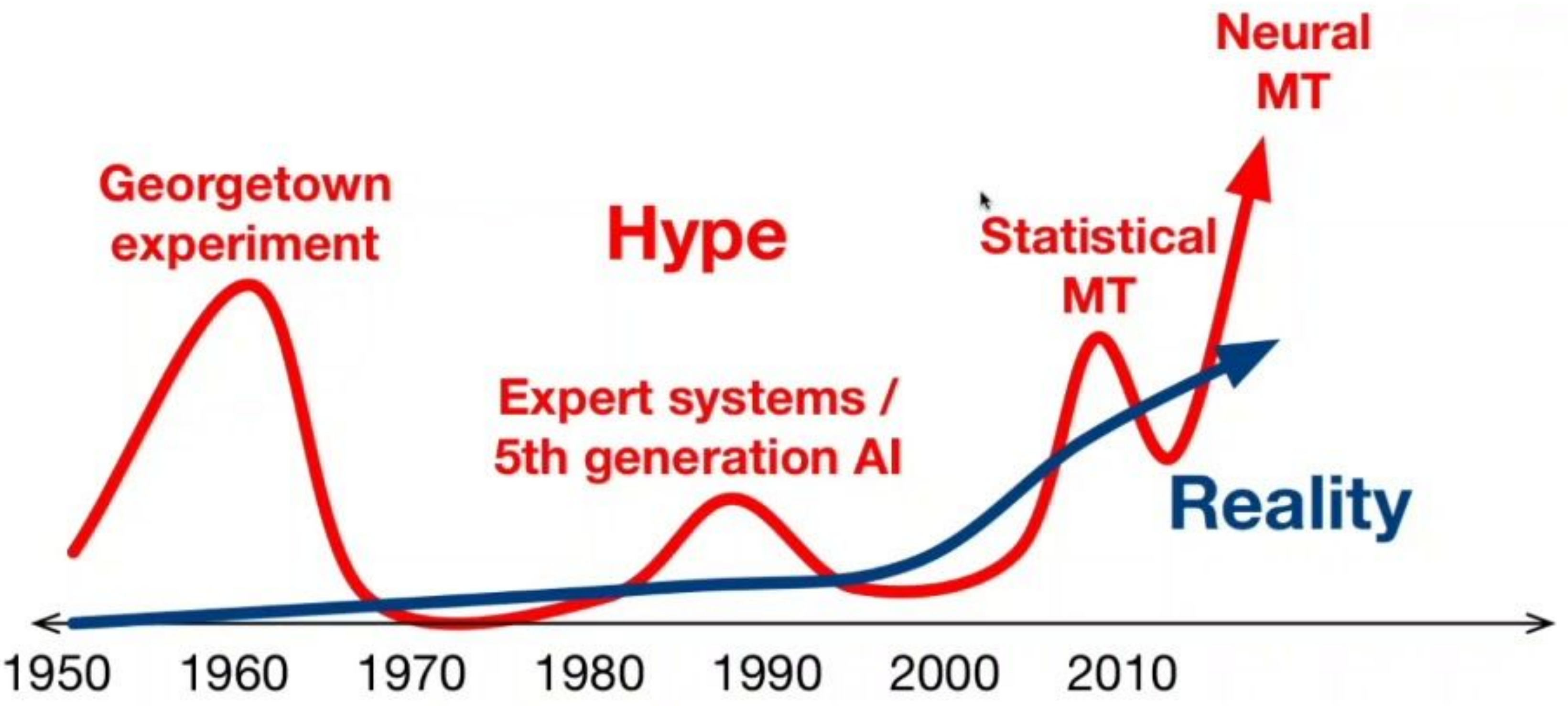


Bahdanau et al. "Neural Machine Translation by Jointly Learning to Align and Translate", 2014

Attention variants

- Basic dot-product (the one discussed before): $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
  - $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ - weight matrix
- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$
  - $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}, \boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ - weight matrices
  - $\boldsymbol{v} \in \mathbb{R}^{d_3}$ - weight vector

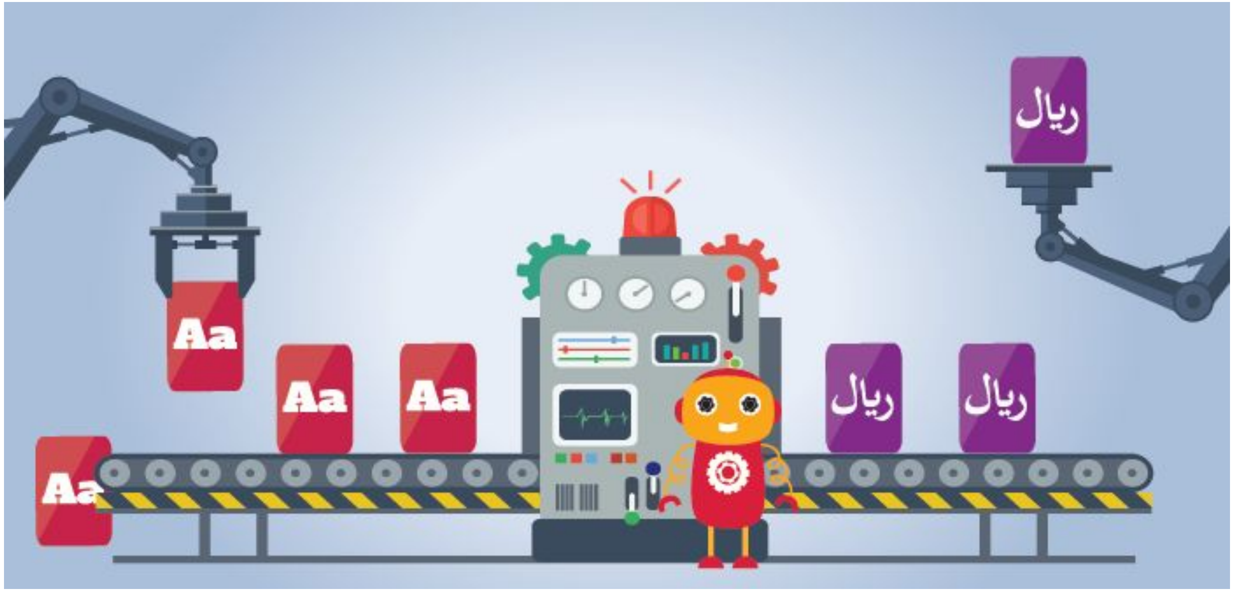
Summary



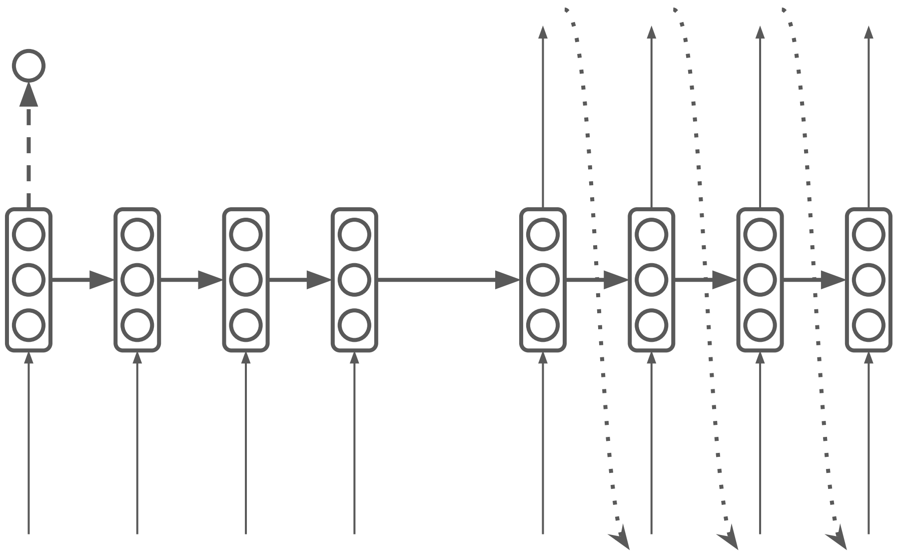
Neural
MT
Georgetown
experiment
Hype
Statistical
MT
Expert systems /
5th generation AI
Reality
1950
1960
1970
1980
1990
2000
2010

Summary

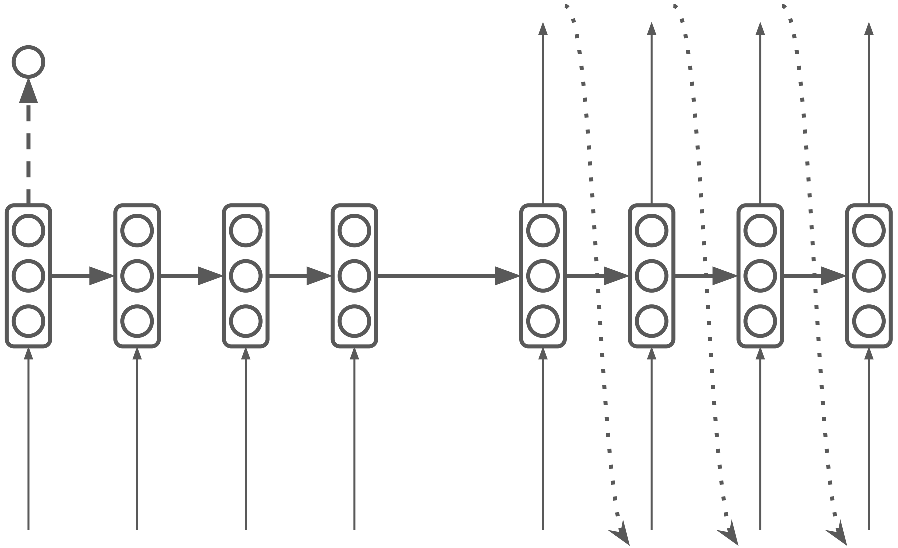
- Seq2seq is an architecture for NMT (2 RNNs)
- Attention is a way to focus on particular parts of the input



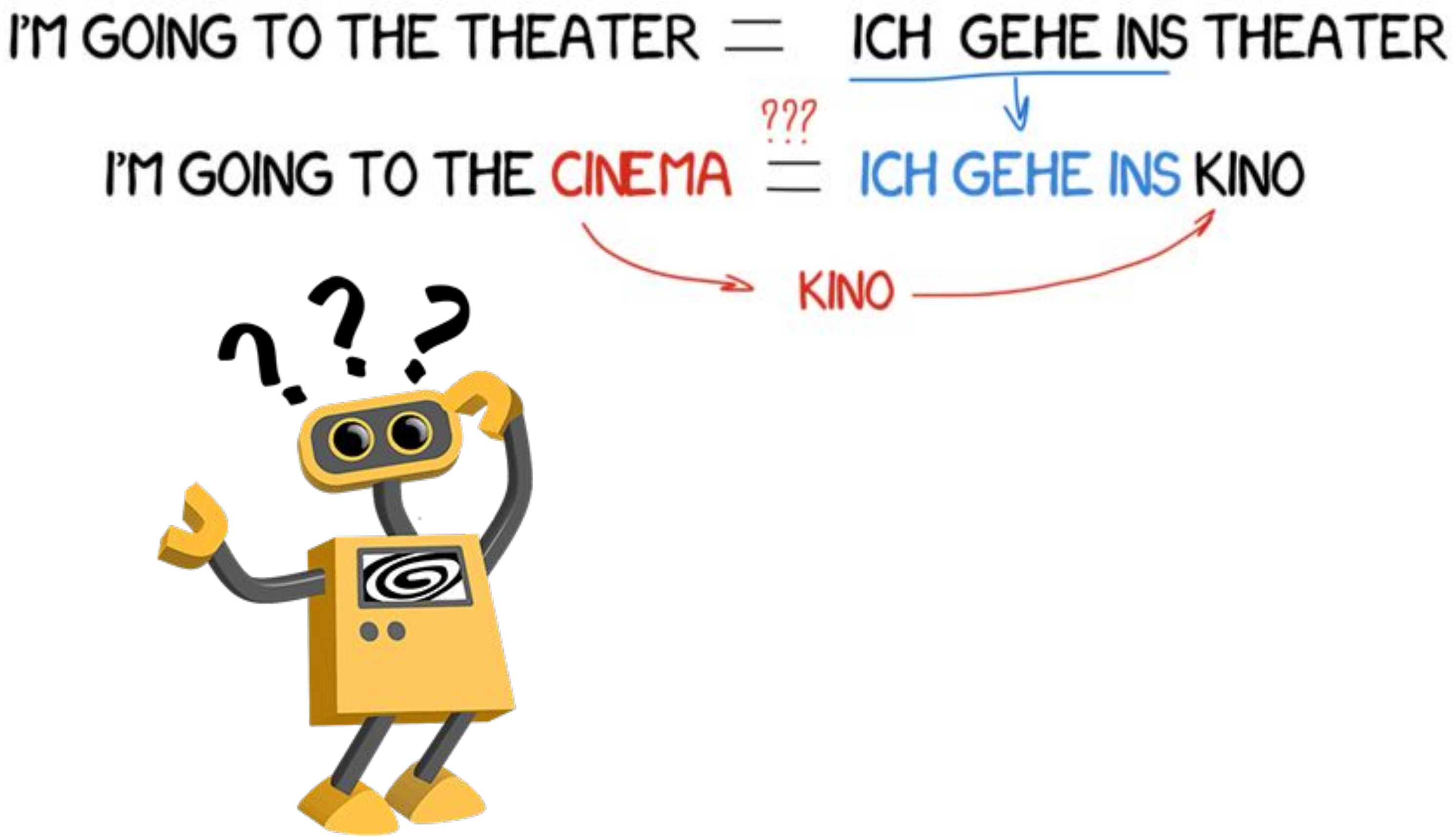
Seq2seq with attention



Seq2seq with attention



Machine Translation



Quality evaluation: Perplexity

$$PP(W) = P(w_1, w_2, .., w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, .., w_N)}} = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(w_i|w_1, .., w_{i-1})}}$$

WER (Word Error Rate)

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- C is the number of correct words,
- N is the number of words in the reference (N = S + D + C)

ROUGE

- **ROUGE** Recall-Oriented Understudy for Gisting Evaluation
- Recall in the context of ROUGE means how much of the reference summary is the system summary recovering or capturing
- **BLEU** is focusing on **precision:**
  - overlapping_words / total_words_in_**system**_summary
- **ROUGE** is focusing on **recall:**
  - overlapping_words / total_words_in_**reference**_summary

ROUGE

ROUGE - Recall-Oriented Understudy for Gisting Evaluation

- **ROUGE-N:** Overlap of N-grams between the system and reference summaries.
- **ROUGE-L:** Longest Common Subsequence (LCS) based statistics. Longest common subsequence problem takes into account sentence level structure similarity naturally and identifies longest co-occurring in sequence n-grams automatically.
- **ROUGE-W:** Weighted LCS-based statistics
- etc.