Machine Learning course,
advanced track

Machine Translation

Anastasia Ianina

MIPT
20.09.2019

Outline

1. Problem Statement
2. Before Deep Learning
3. Neural Machine Translation (NMT)
    - Seq2Seq
    - BeamSearch
4. Attention
5. Q & A

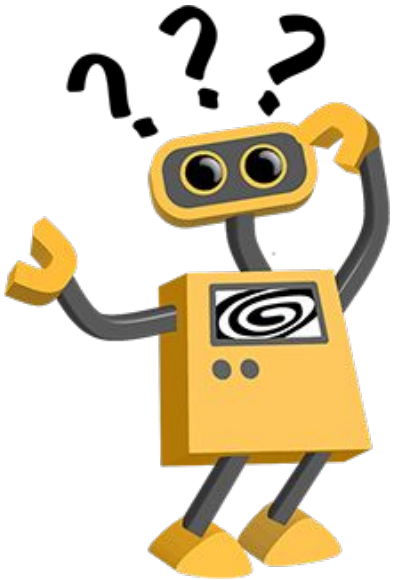
Based on: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Machine Translation

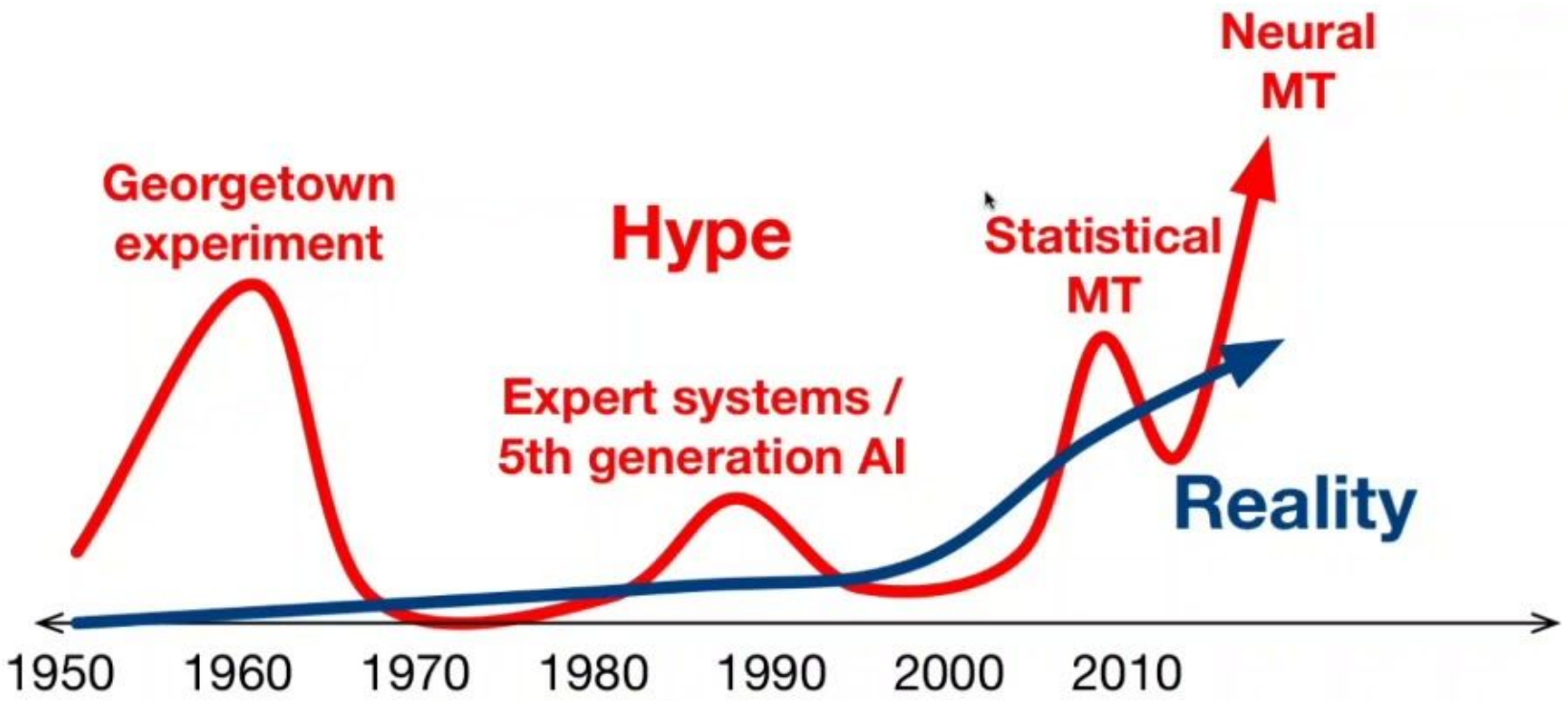
I'M GOING TO THE THEATER = ICH GEHE INS THEATER

I'M GOING TO THE CINEMA ??? = ICH GEHE INS KINO

KINO



Historical overview



Before Deep Learning

1950s: first Machine Translation

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

1990-2010: Statistical Machine Translation

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

$$\text{argmax}_y P(y|x)$$

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

$$= \text{argmax}_y \underbrace{P(x|y)}\underbrace{P(y)}$$

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.

1990-2010: Statistical Machine Translation

How to learn translation model from the parallel corpus?

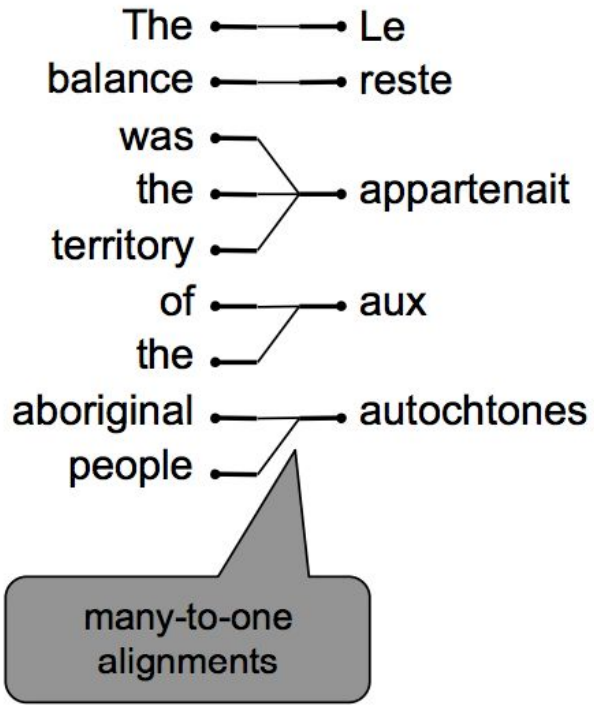
Let's calculate

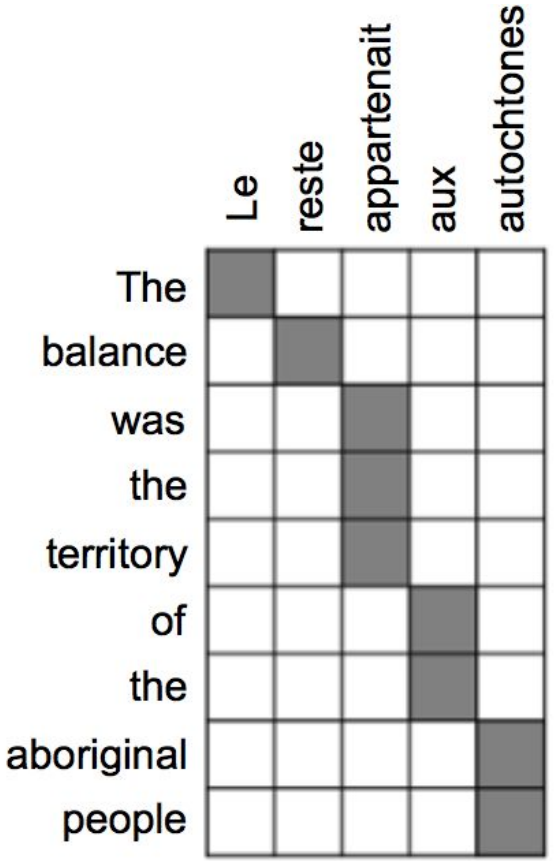
$$P(x, a|y)$$

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

1990-2010: Statistical Machine Translation

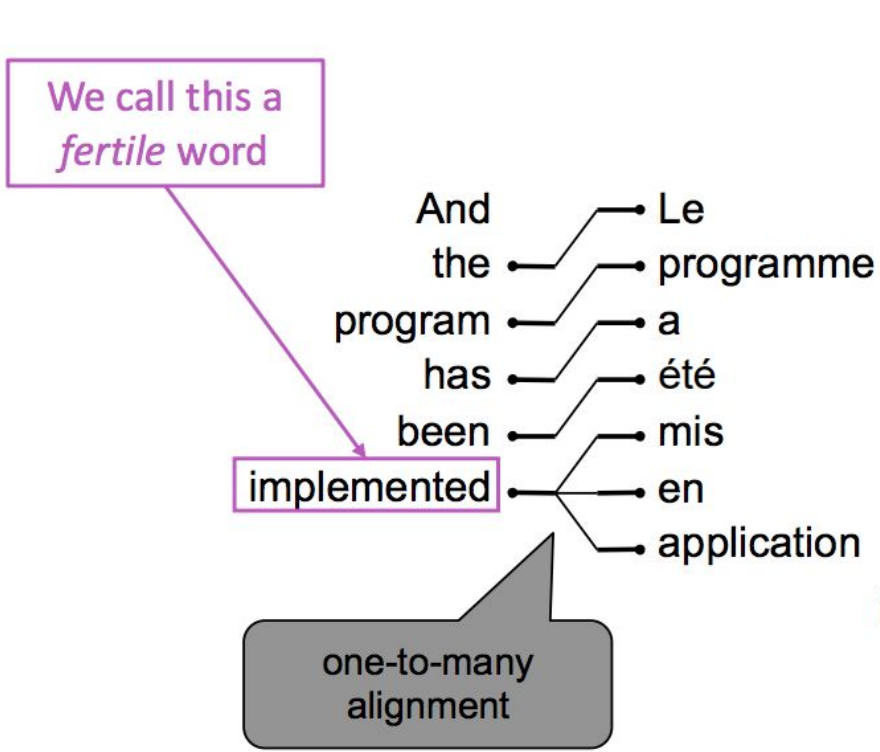
Alignment can be: **many-to-one**

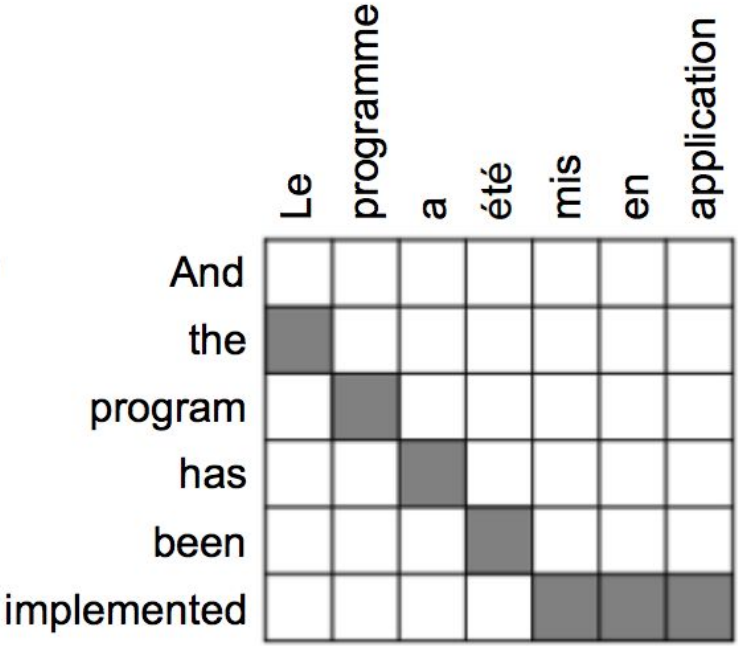




1990-2010: Statistical Machine Translation

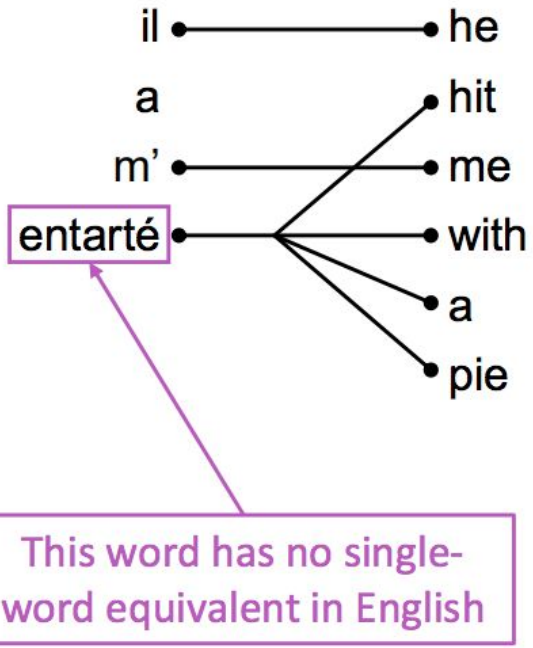
Alignment can be: **one-to-many**

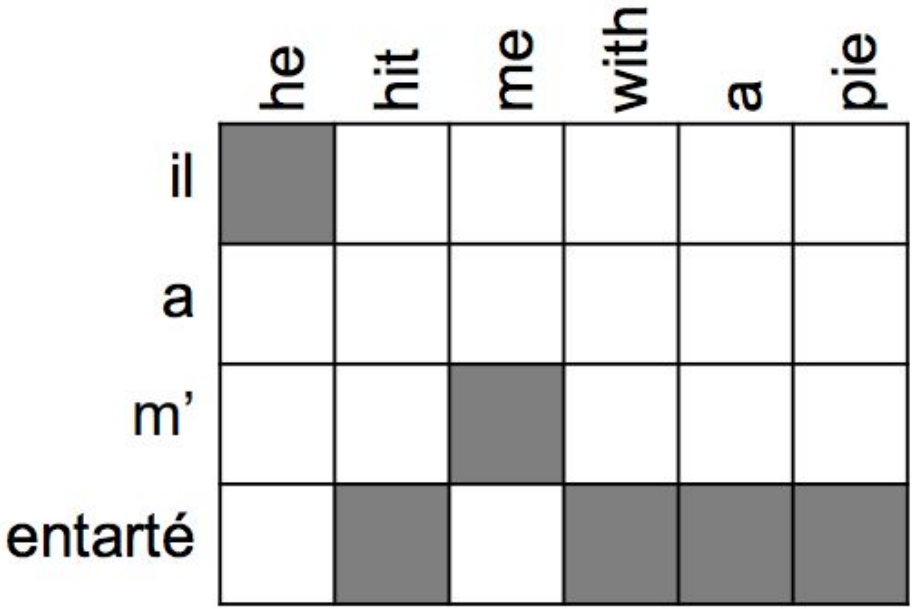




1990-2010: Statistical Machine Translation

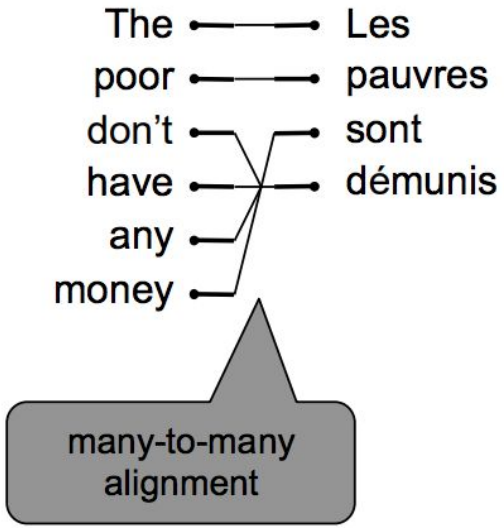
Some words are very fertile!

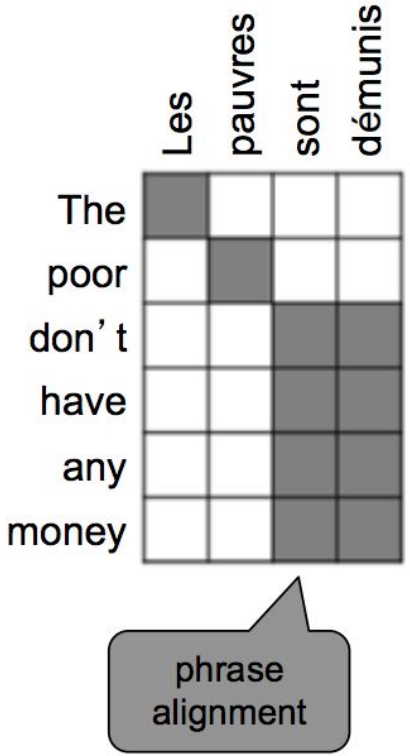




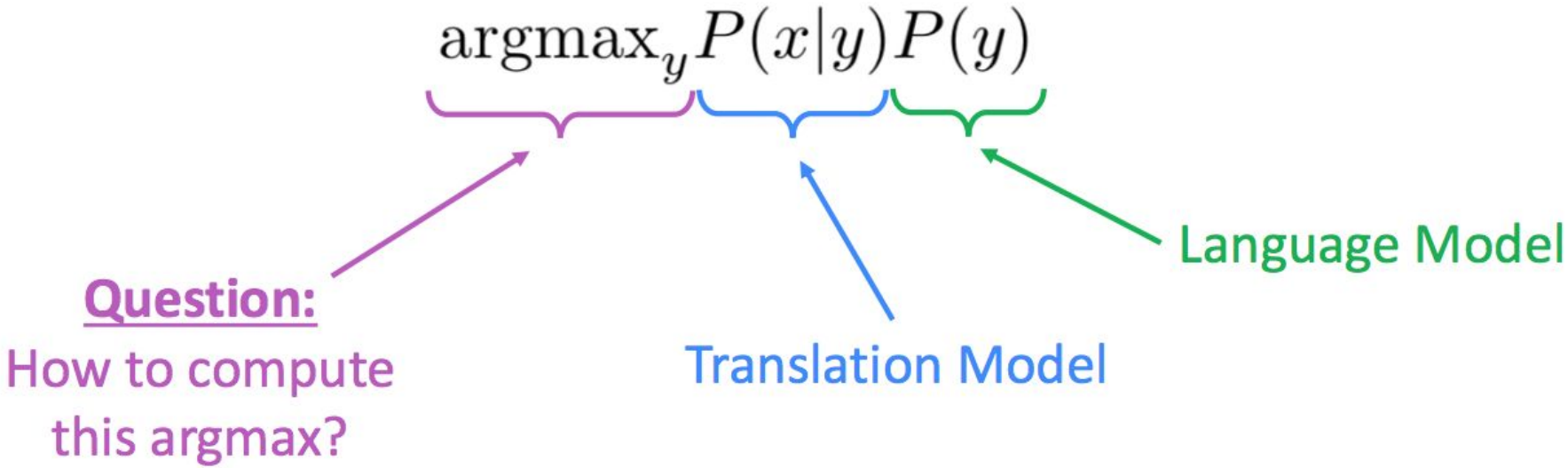
1990-2010: Statistical Machine Translation

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





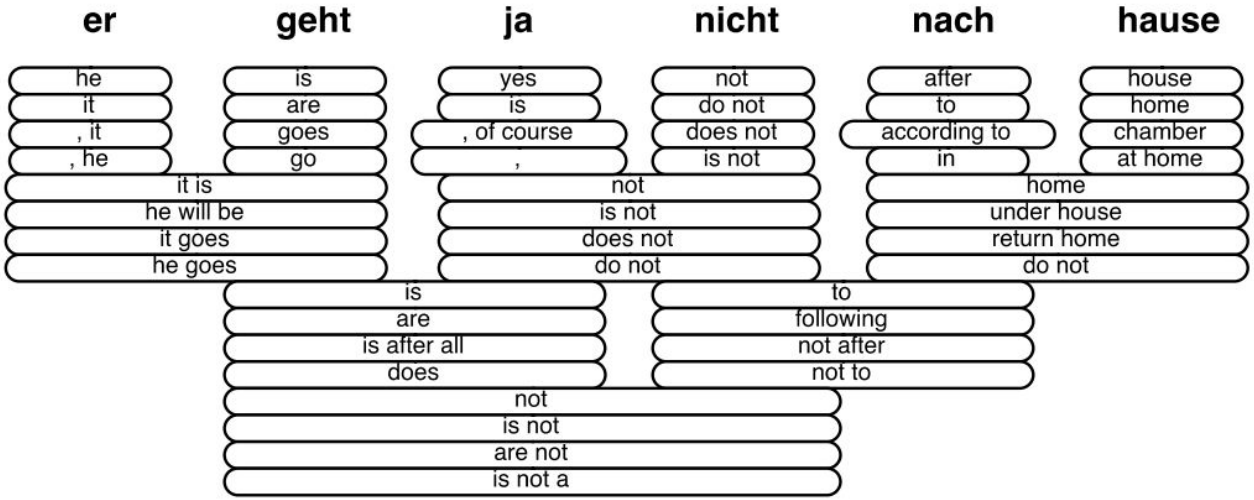
1990-2010: Statistical Machine Translation

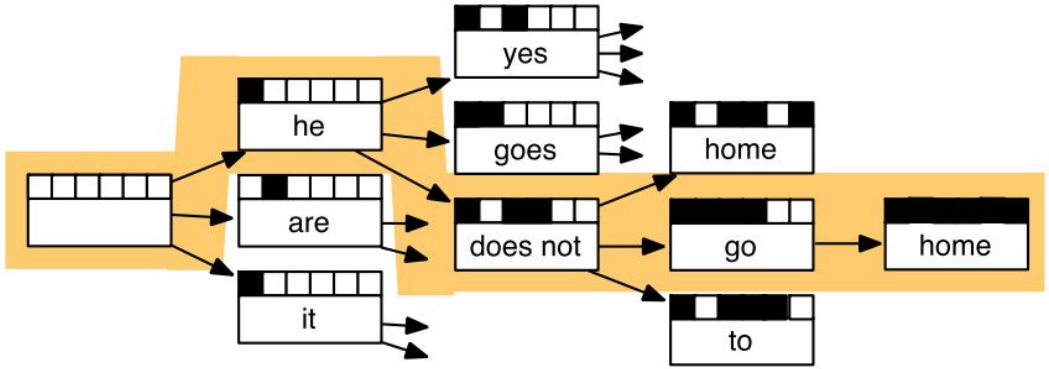


Enumerate every possible y and calculate the probability? No!

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

1990-2010: Statistical Machine Translation





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)

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!

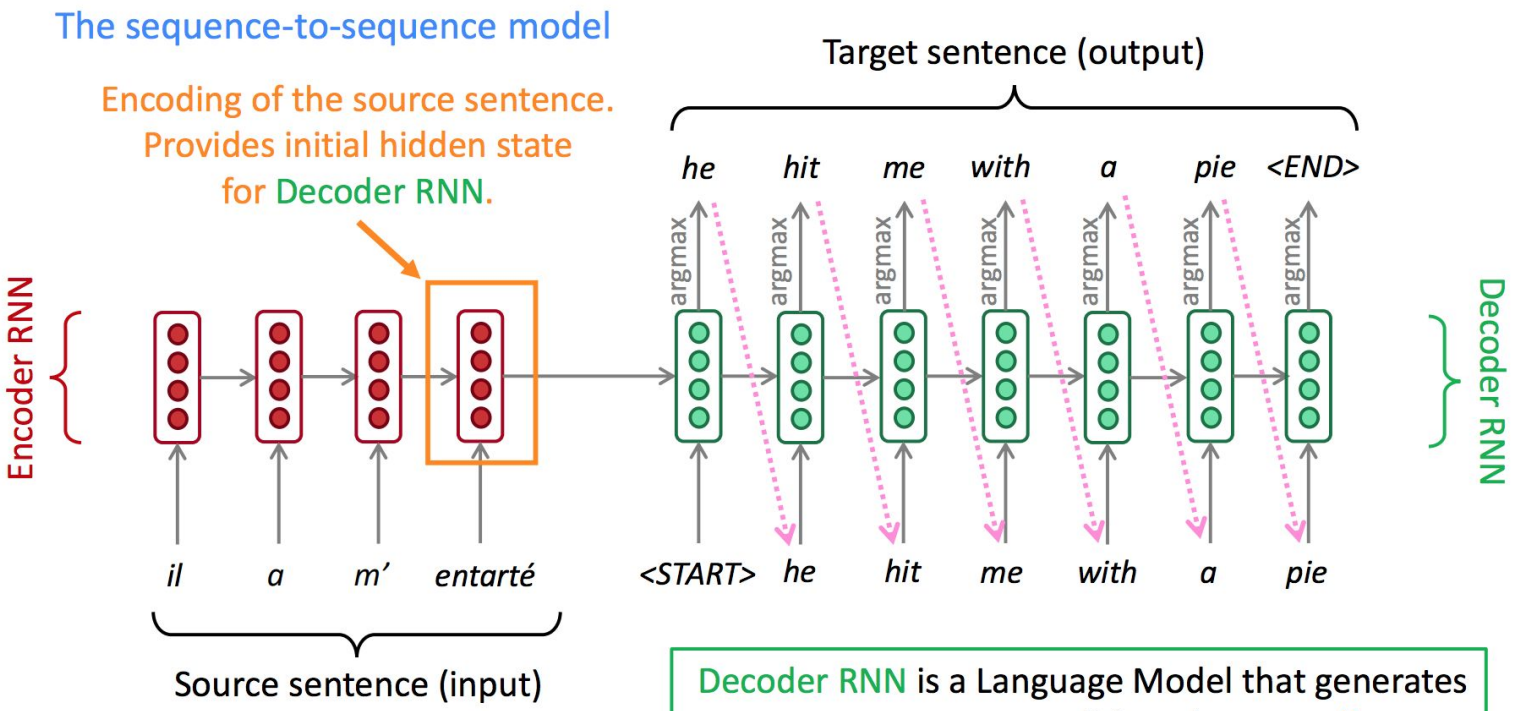
Neural Machine Translation

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**) and it involves two **RNNs**.

Neural Machine Translation



NMT: how does it work?

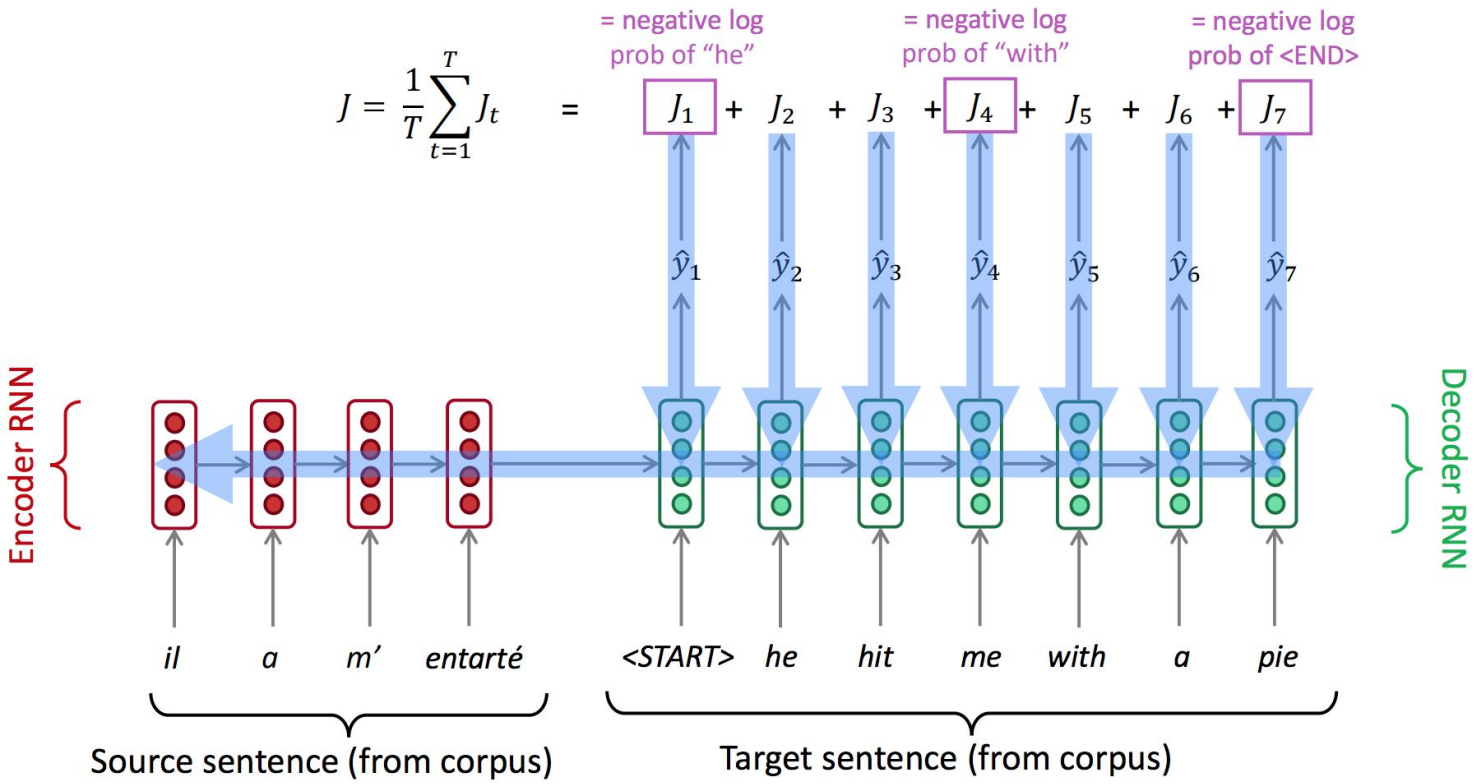
- NMT directly calculates P(y|x) (y - target sentence, x - source sentence)

$$P(y|x) = P(y_1|x)\, P(y_2|y_1, x)\, P(y_3|y_1, y_2, x) \ldots \underbrace{P(y_T|y_1, \ldots, y_{T-1}, x)}$$

Probability of next target word, given target words so far and source sentence *x*

- To train it we need a huge parallel corpus.

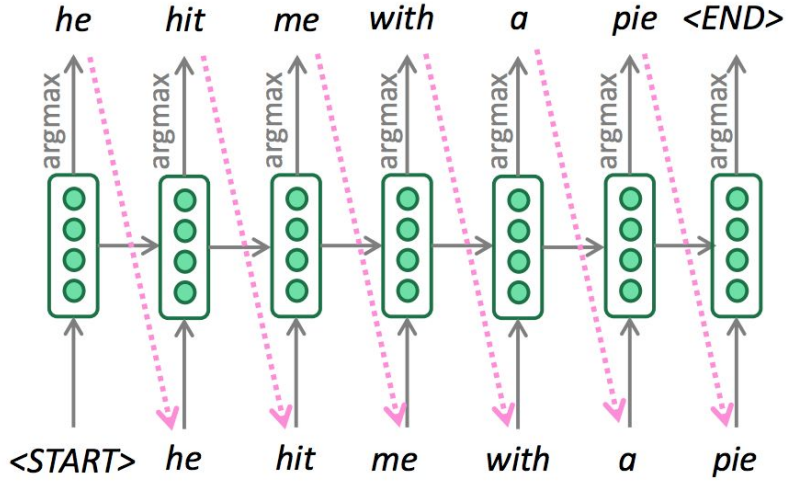
Neural Machine Translation



Seq2seq is optimized as a **single system.**
Backpropagation operates "*end-to-end*".

Greedy decoding

- We saw how to generate (or “decode”) the target sentence by taking argmax on each step of the decoder



- This is greedy decoding (take most probable word on each step)

Greedy decoding: problems

- Greedy decoding has no way to undo decisions!

Input: *il a m'entarté* *(he hit me with a pie)*

→ *he* ____

→ *he hit* ____

→ *he hit a* ____ (whoops! no going back now…)

Exhaustive search

- Ideally we want to find a (length T) translation y that maximizes

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x)$$
$$= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)$$

- We could try computing all possible sequences y. That's too slow!

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 is not guaranteed to find optimal solution

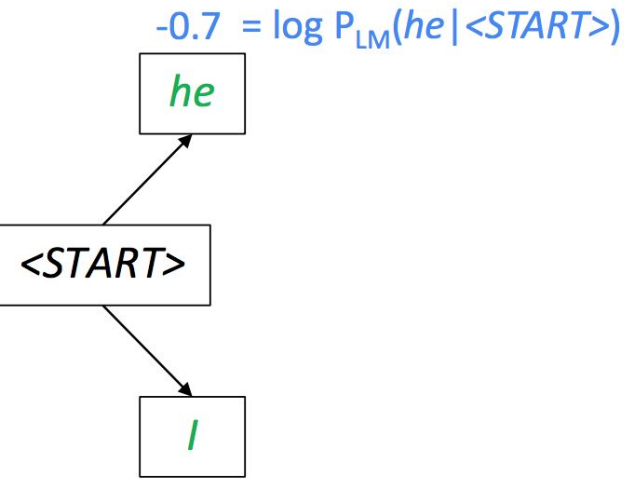
Beam search decoding: example

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

<START>

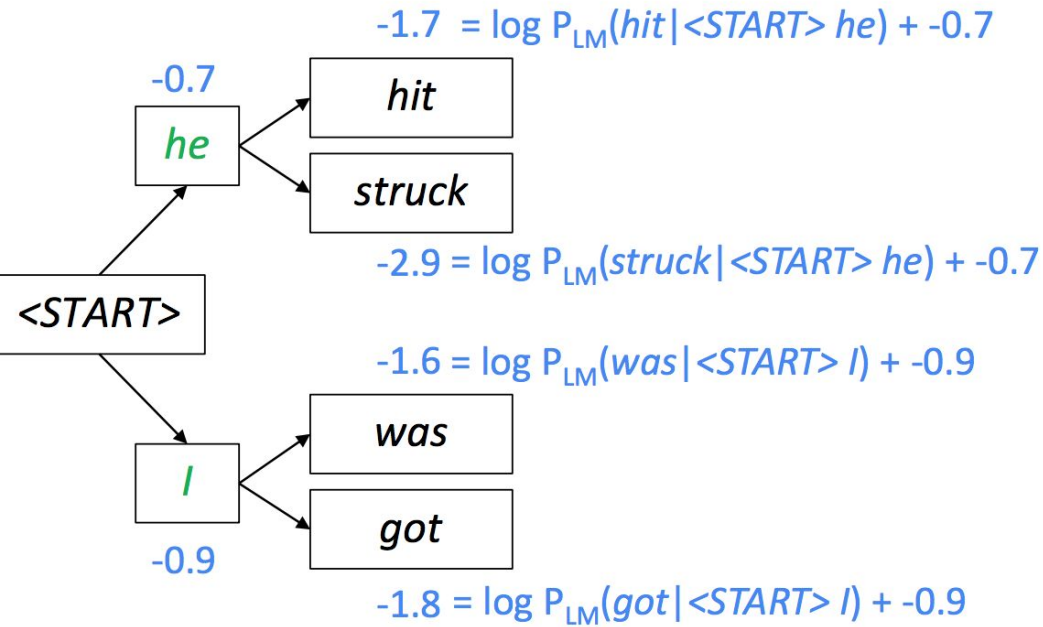
Beam search decoding: example

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



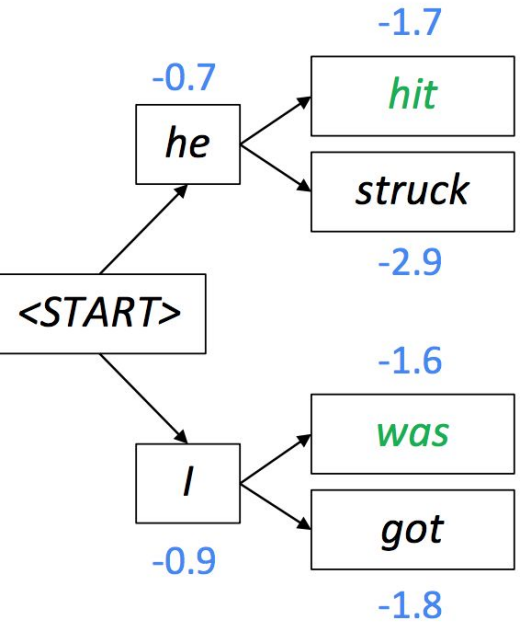
Beam search decoding: example

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



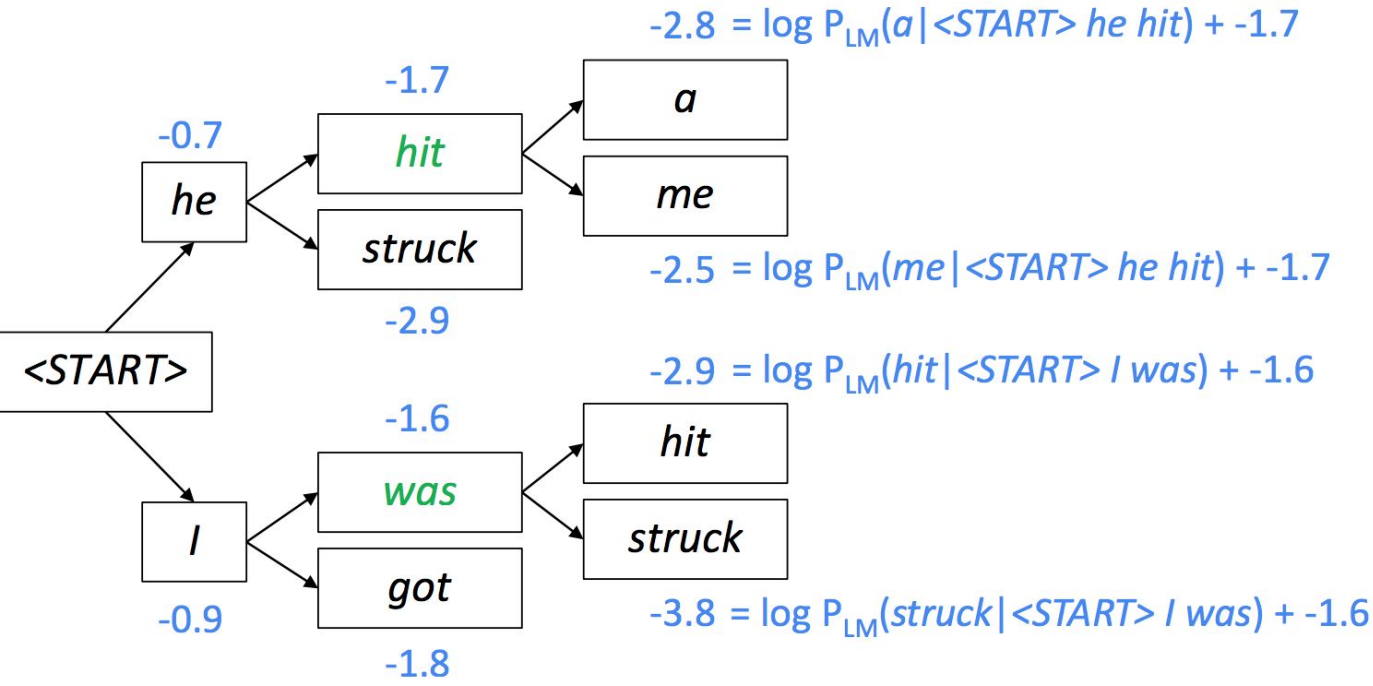
Beam search decoding: example

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



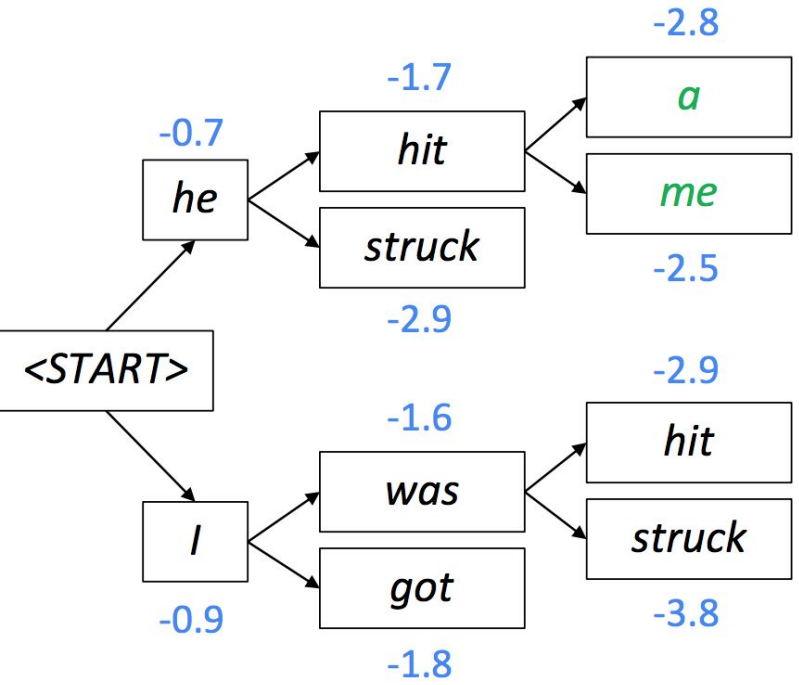
Beam search decoding: example

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



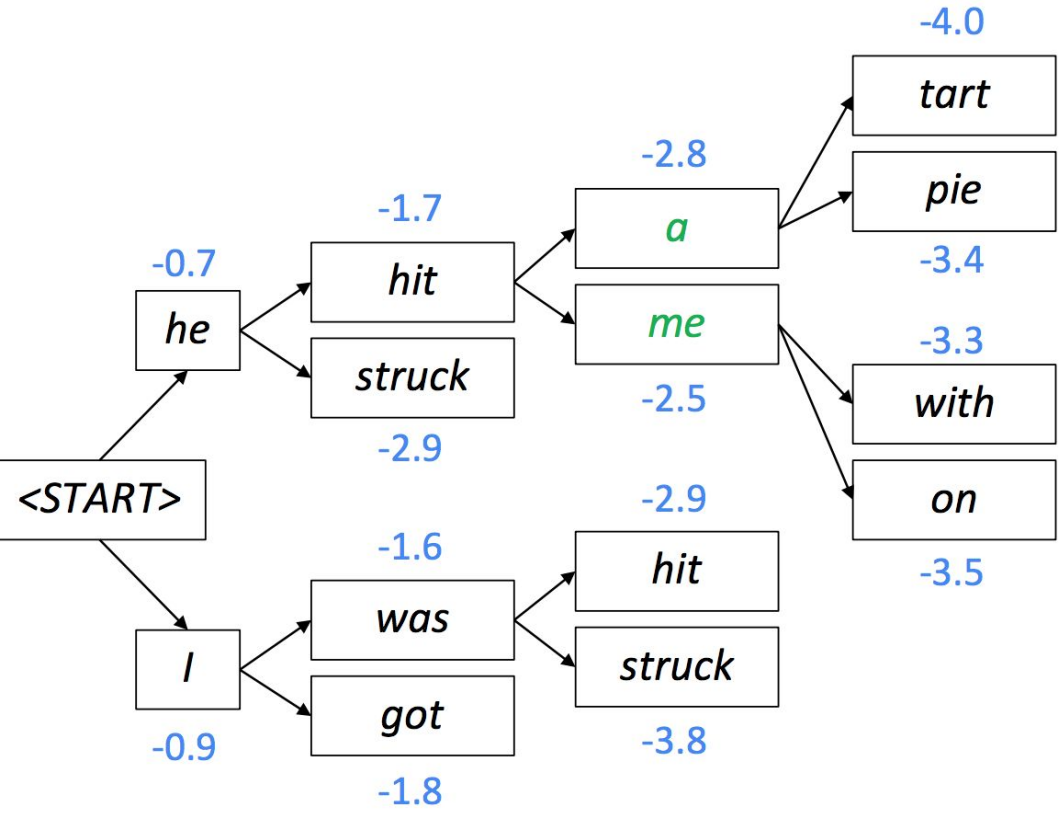
Beam search decoding: example

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



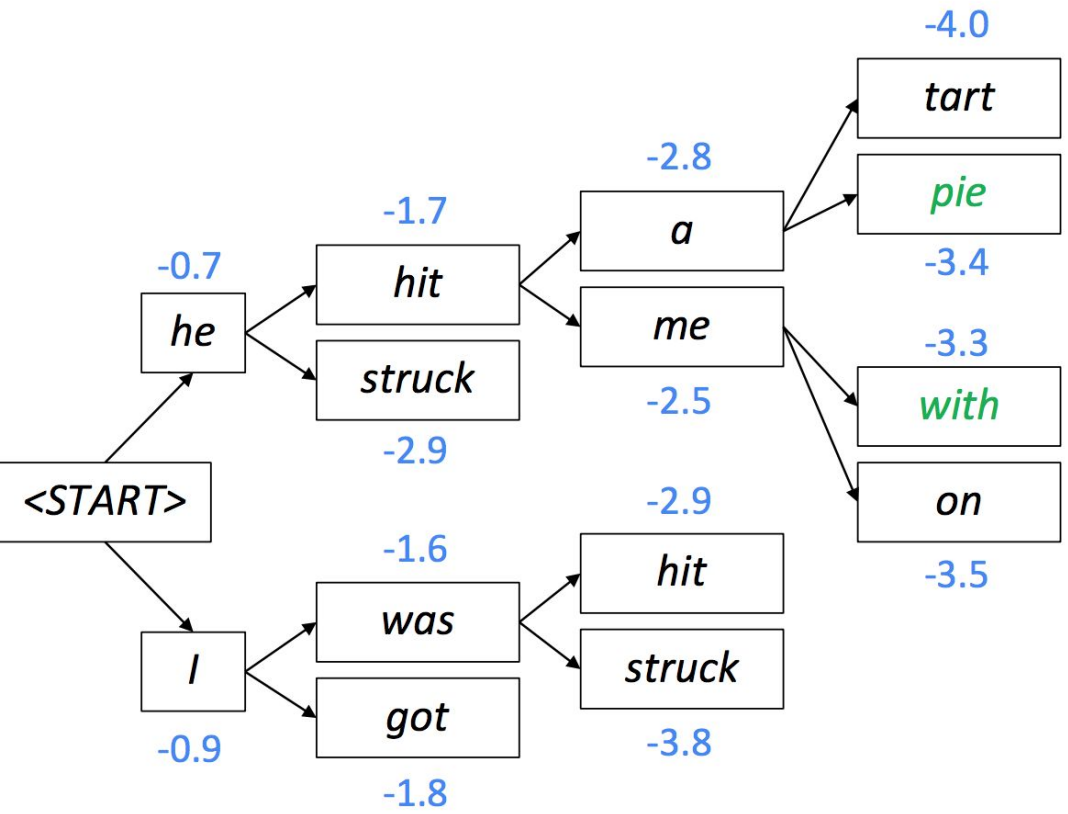
Beam search decoding: example

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



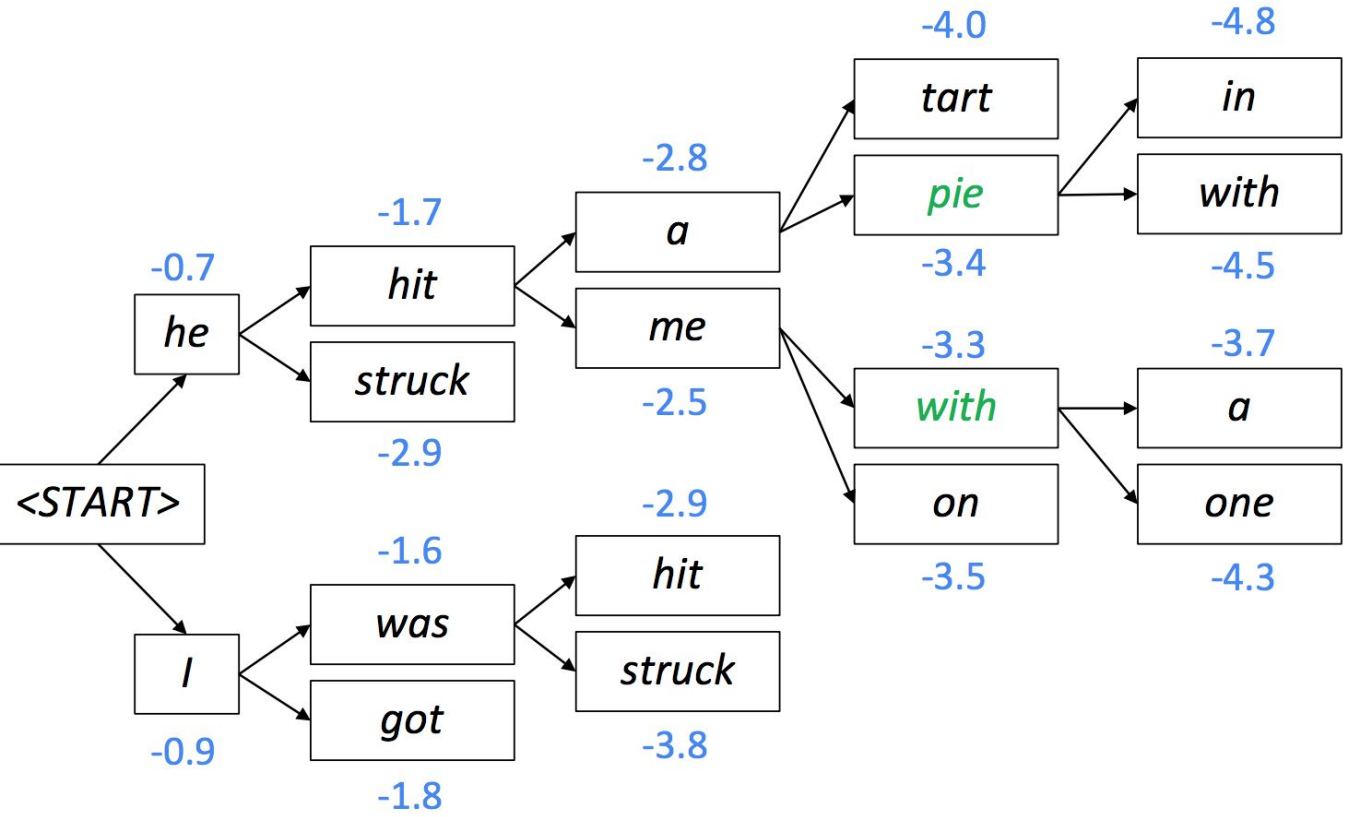
Beam search decoding: example

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



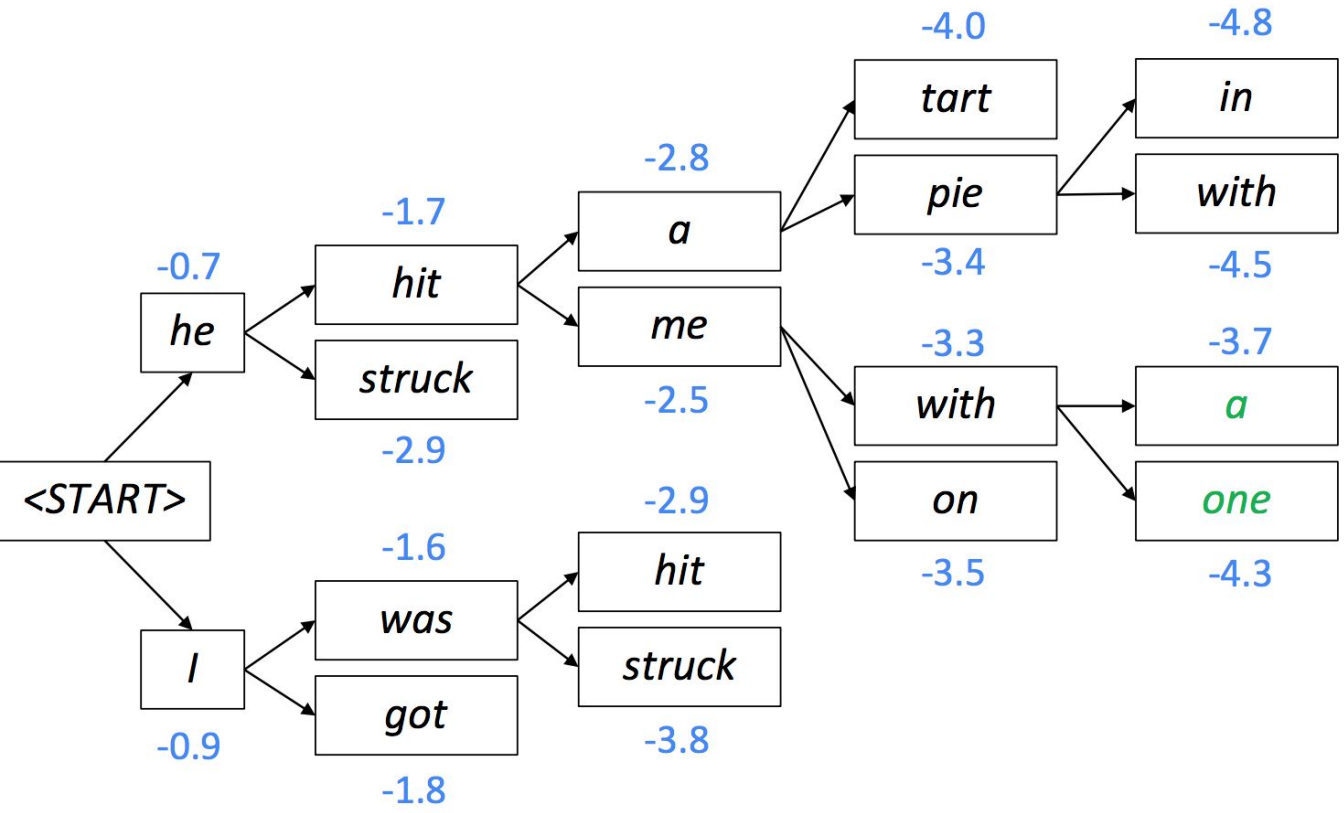
Beam search decoding: example

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



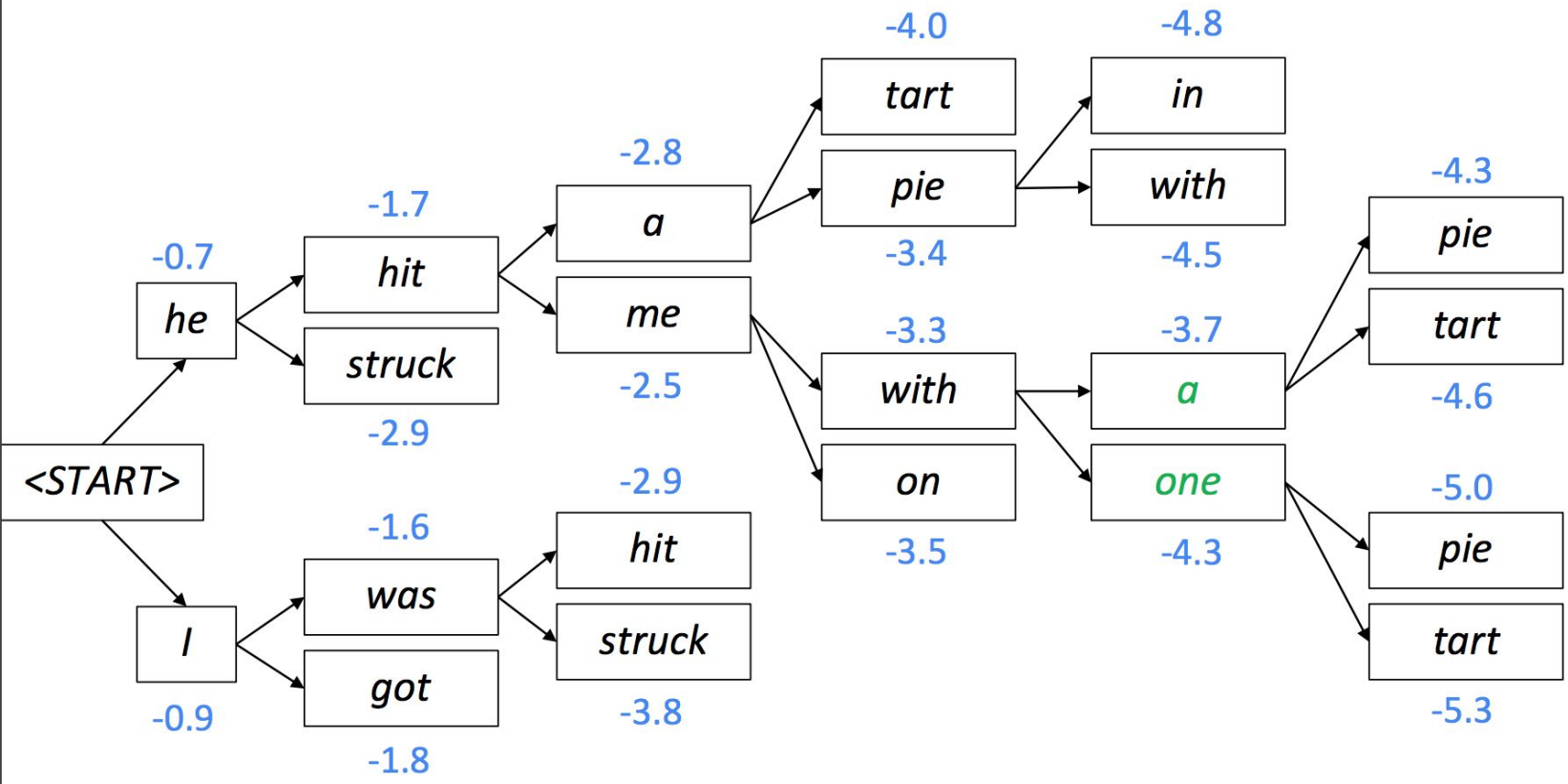
Beam search decoding: example

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



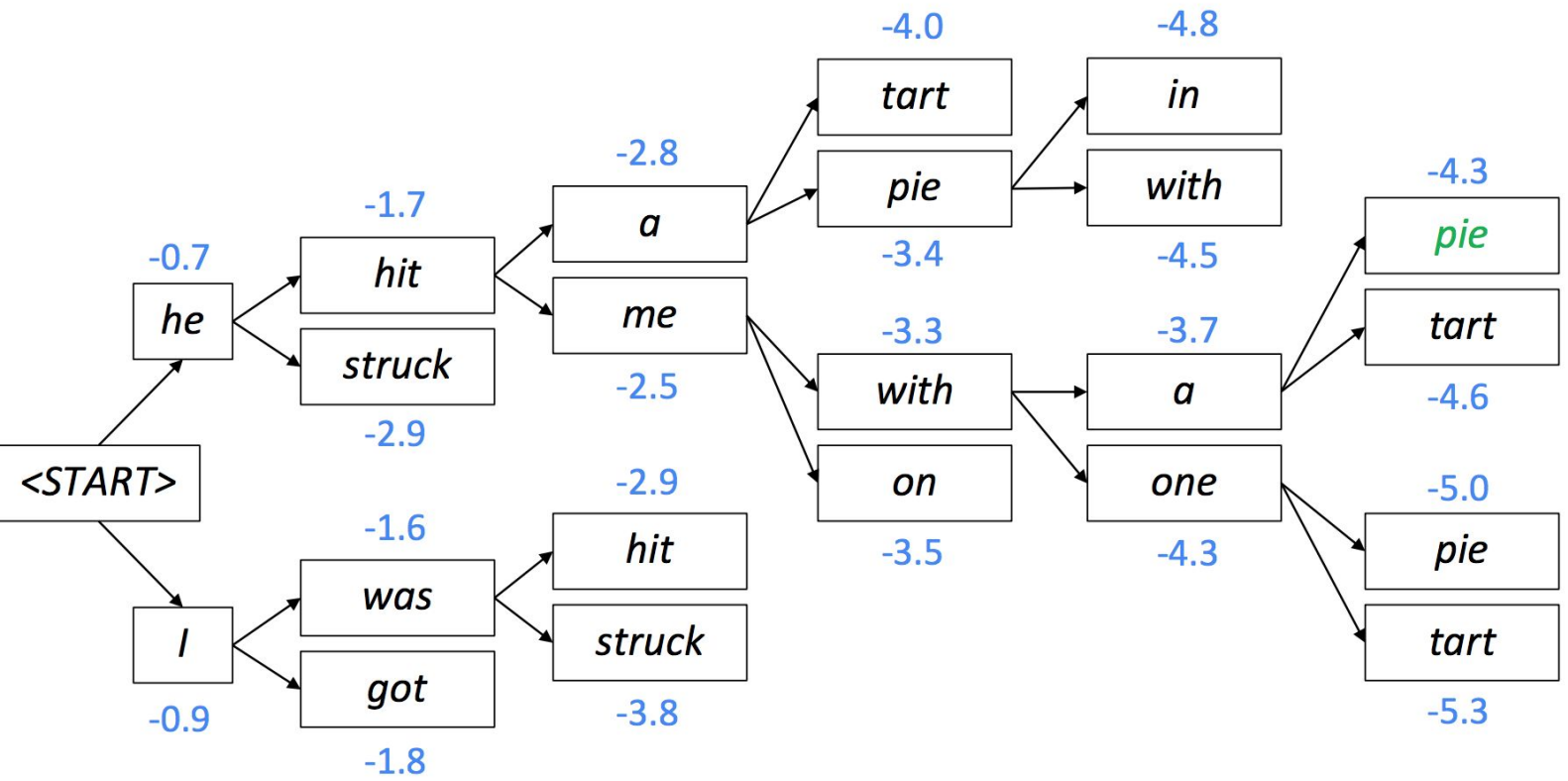
Beam search decoding: example

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



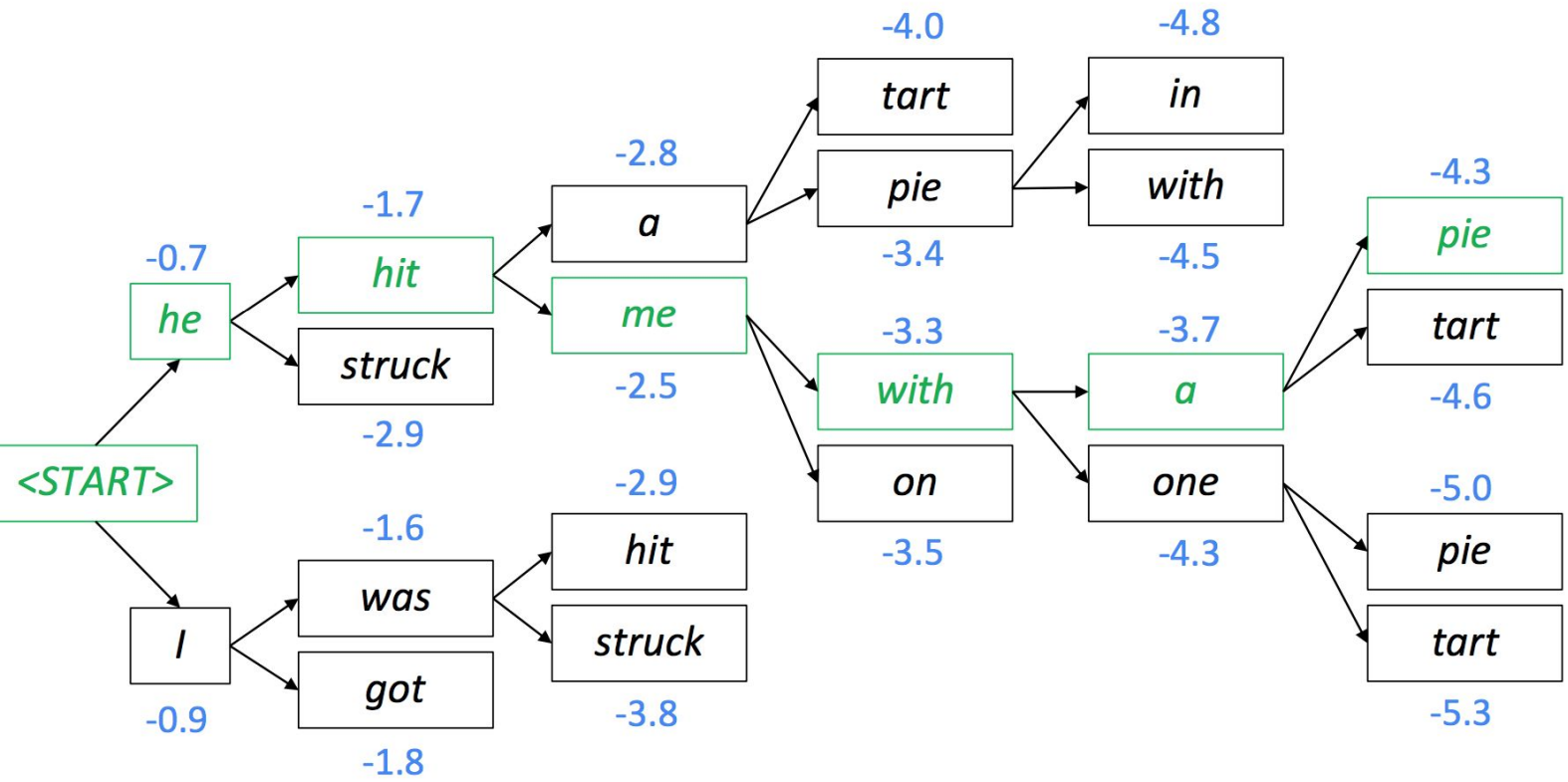
Beam search decoding: example

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



Beam search decoding: example

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



Beam search decoding: stopping criterion

- In **greedy decoding**, usually we decode until the model produces <END> token
- In **beam search decoding**, different hypotheses may produce <END> tokens on different timesteps
  - When a hypothesis produces <END>, 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?**

Beam search decoding: finishing up

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

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

- **Problem:** longer hypotheses have lower scores

Beam search decoding: finishing up

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

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

- **Problem:** 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

Quality evaluation: Perplexity

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1})}}$$

Quality evaluation: BLEU

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

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

REFERENCE: Israeli officials are responsible for airport security

SYSTEM B: airport security (2-GRAM MATCH) Israeli officials are responsible (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%

Quality evaluation: BLEU

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

Quality evaluation: ROUGE

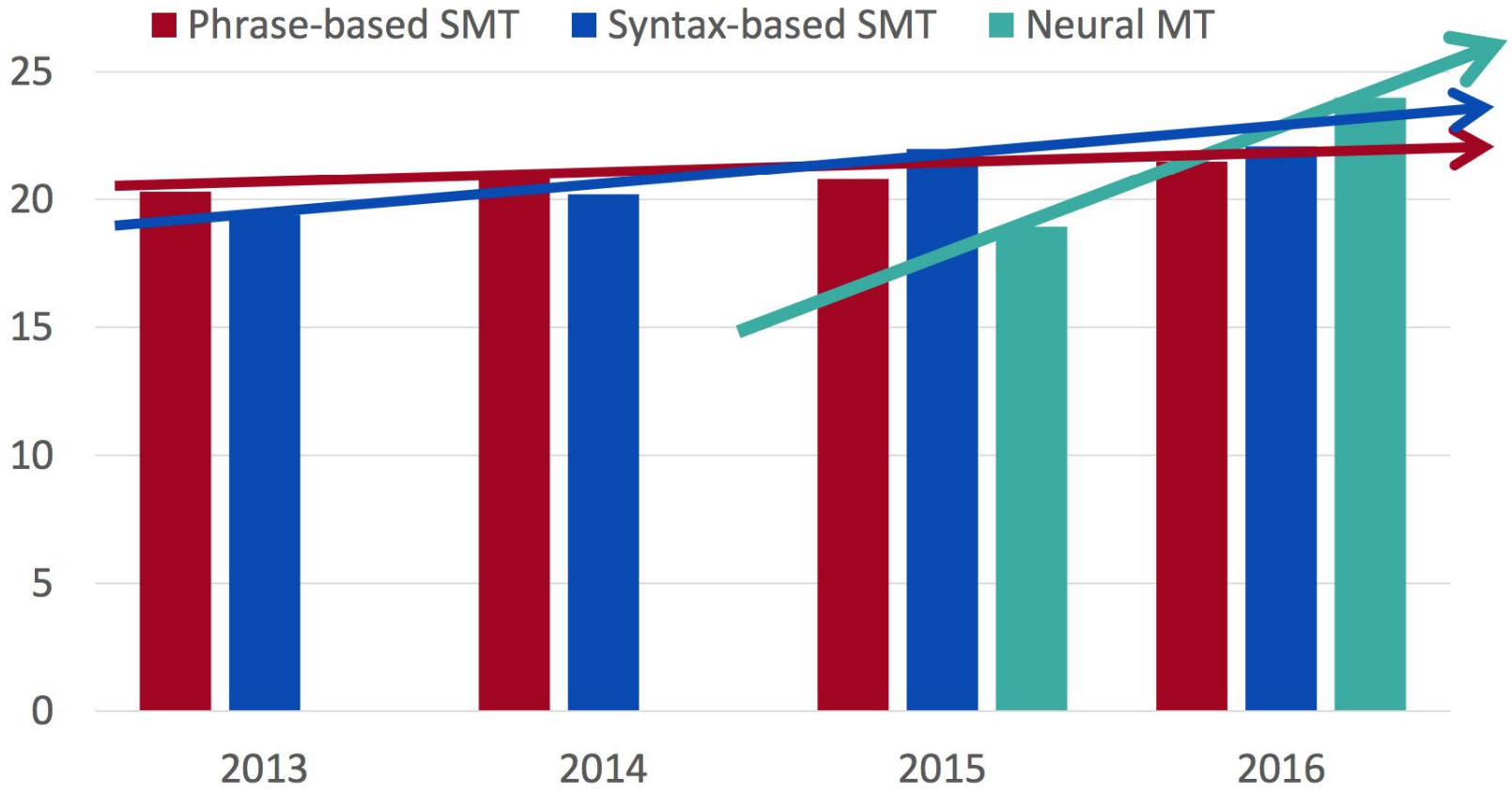
ROUGE - Recall-Oriented Understudy for Gisting Evaluation

- **ROUGE-N:** Overlap of N-grams between the system and reference summaries.
- **ROUGE-L:** Longest Common Subsequence (LCS)[3] 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 that favors consecutive LCSes
- etc.

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 progress over time

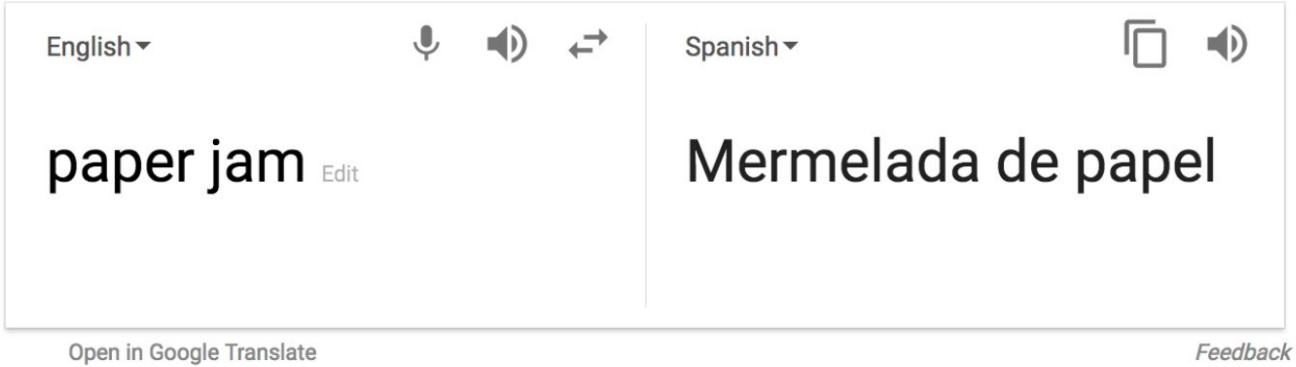


NMT: disadvantages

- NMT is less interpretable
  - Hard to debug

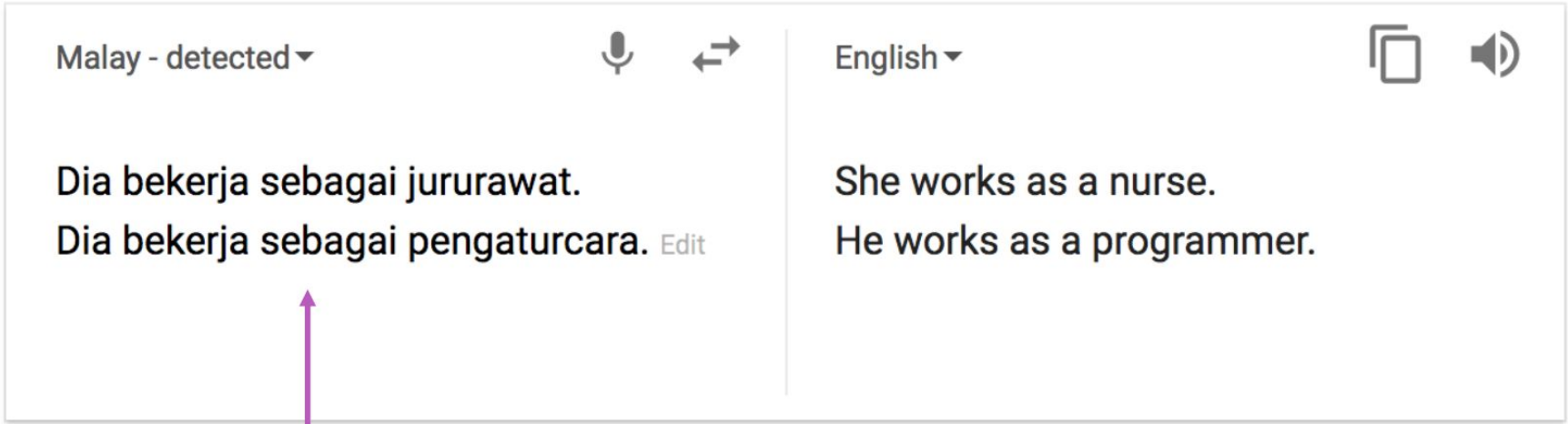
- NMT is difficult to control
  - For example, can’t easily specify rules or guidelines for translation
  - Safety concerns!

NMT: disadvantages



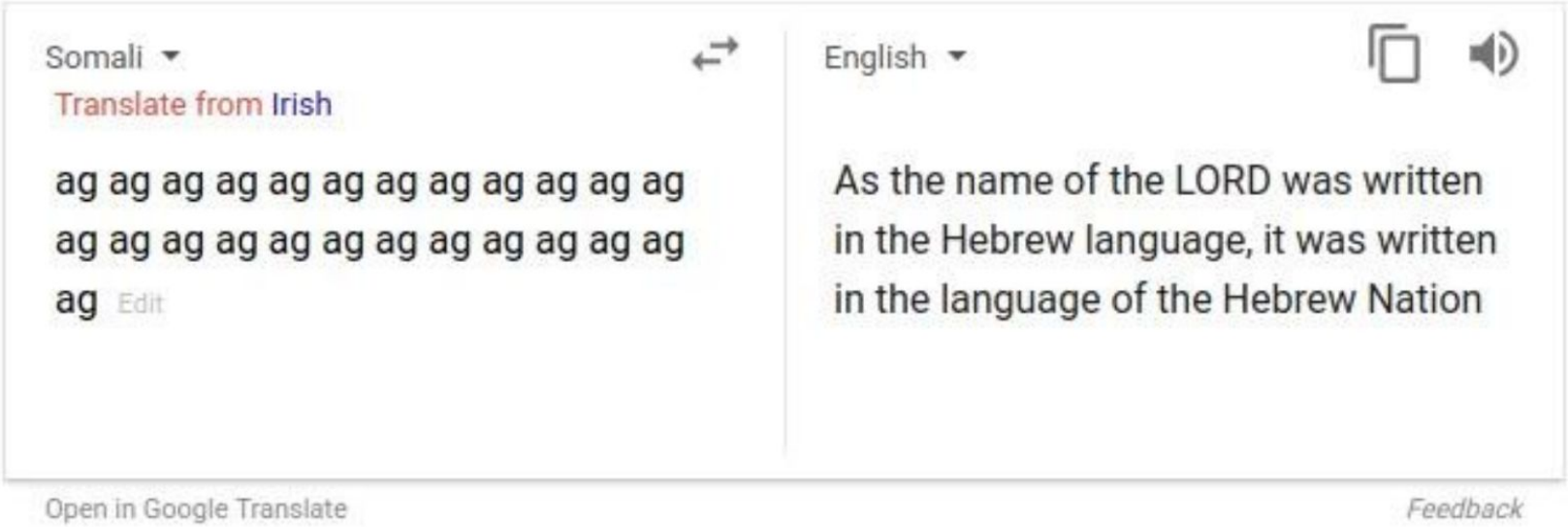
?

NMT: disadvantages



Didn’t specify gender

NMT: disadvantages

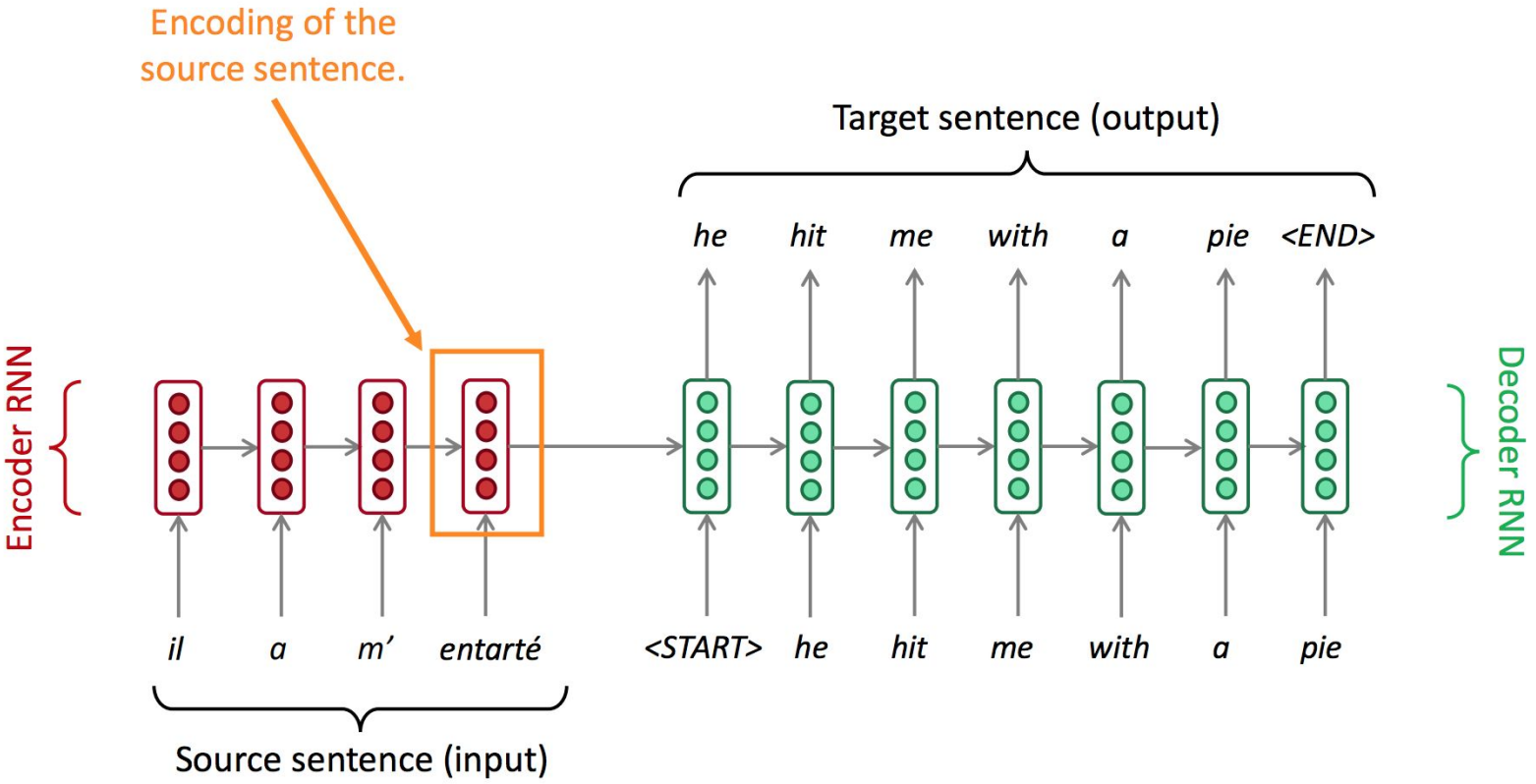


Is Machine Translation solved?

- 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)

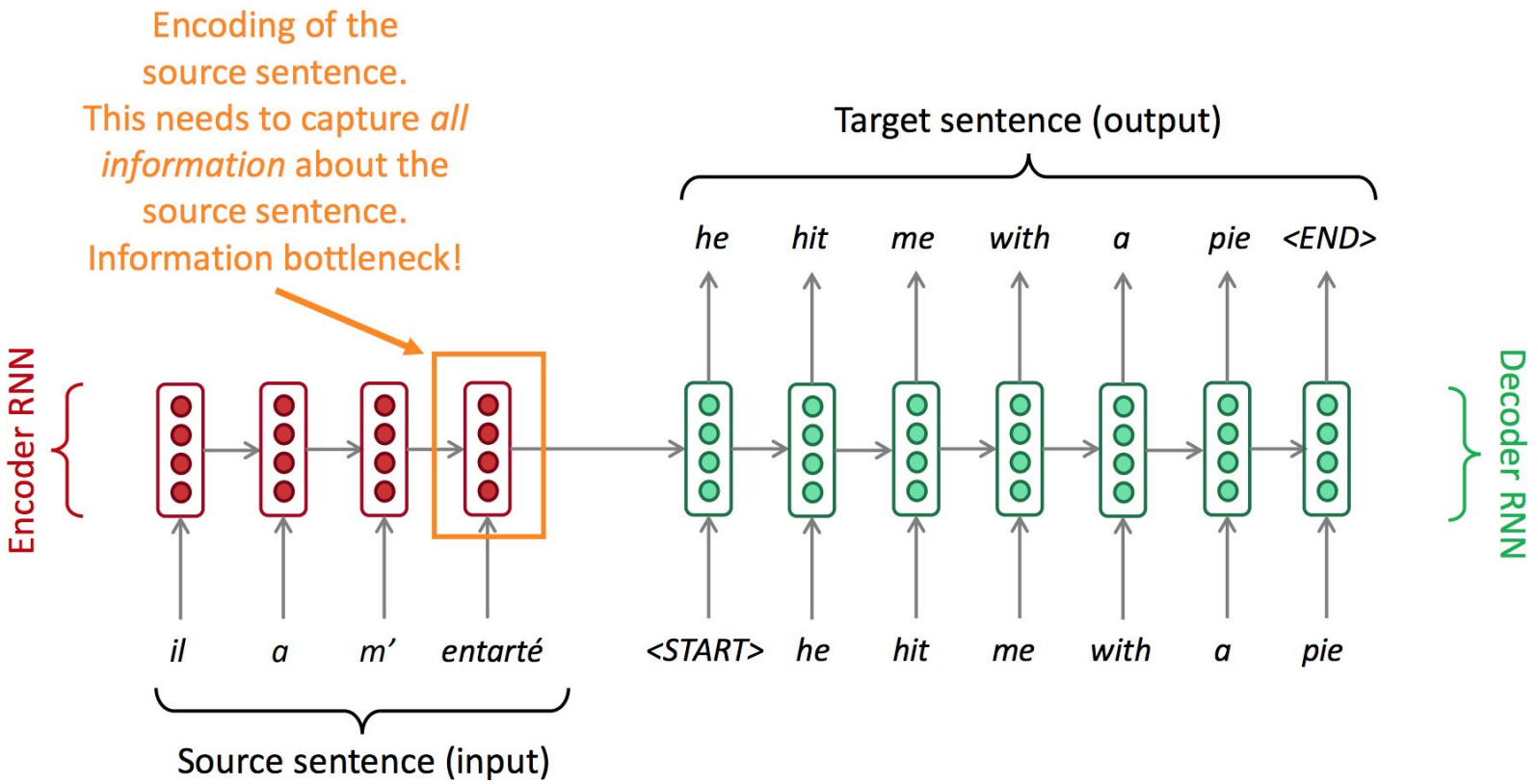
Attention

Seq2seq: the bottleneck problem



Problems with this architecture?

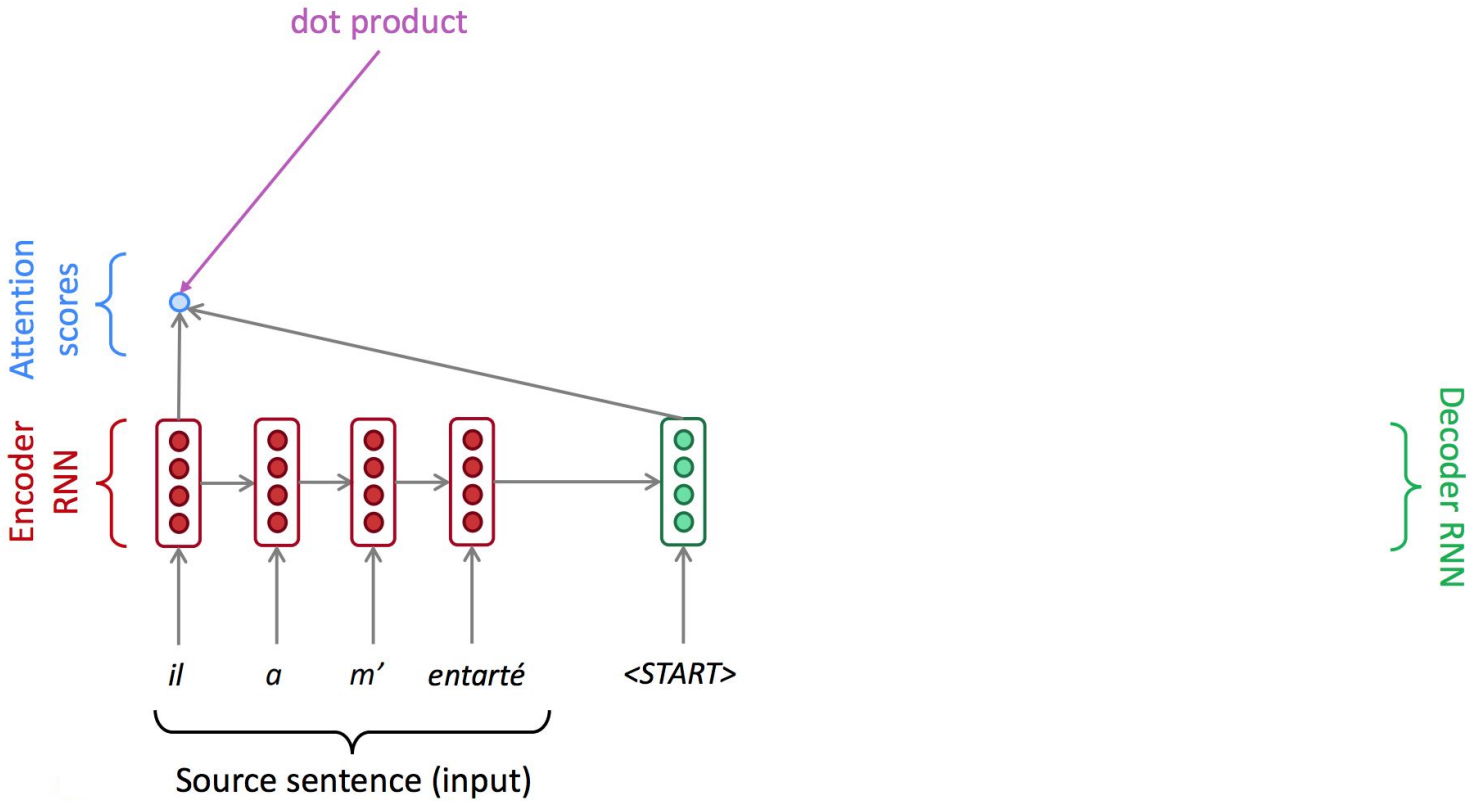
Seq2seq: the bottleneck problem



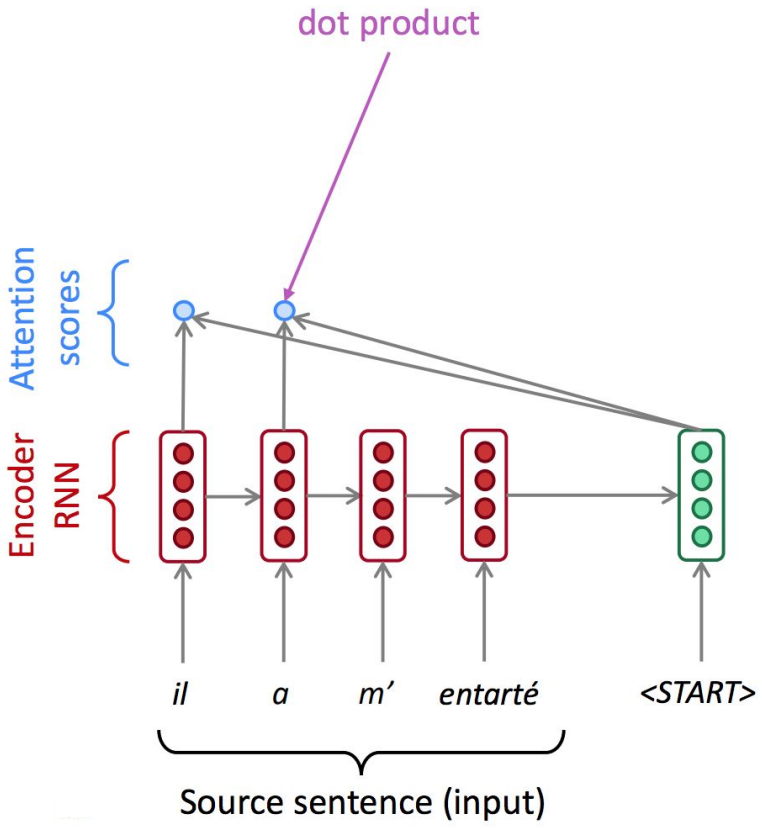
Attention

<u>Main idea:</u> on each step of the **decoder**, use **direct connection to the encoder** to focus on a particular part of the source sequence

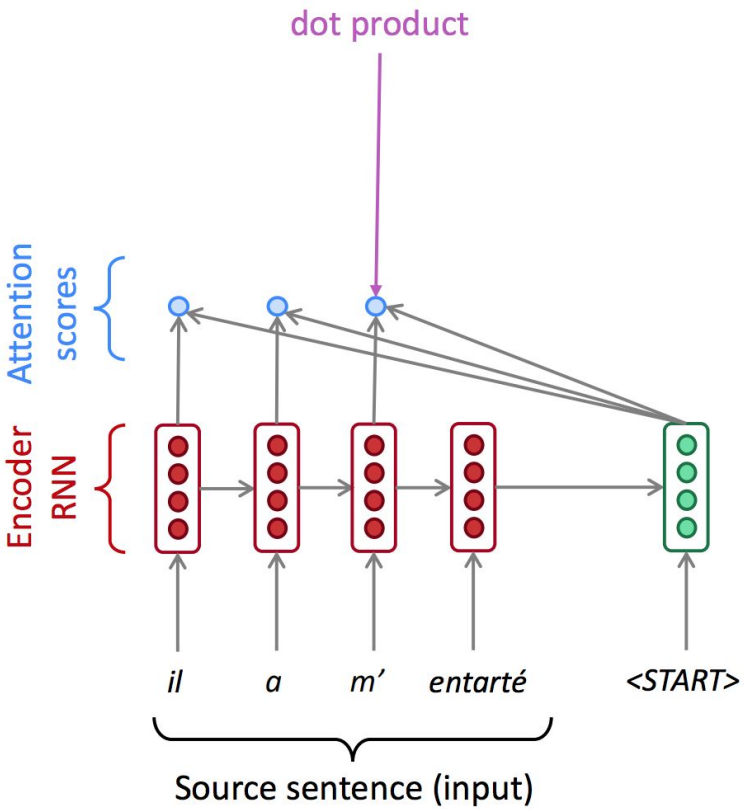
Seq2seq with attention



Seq2seq with attention

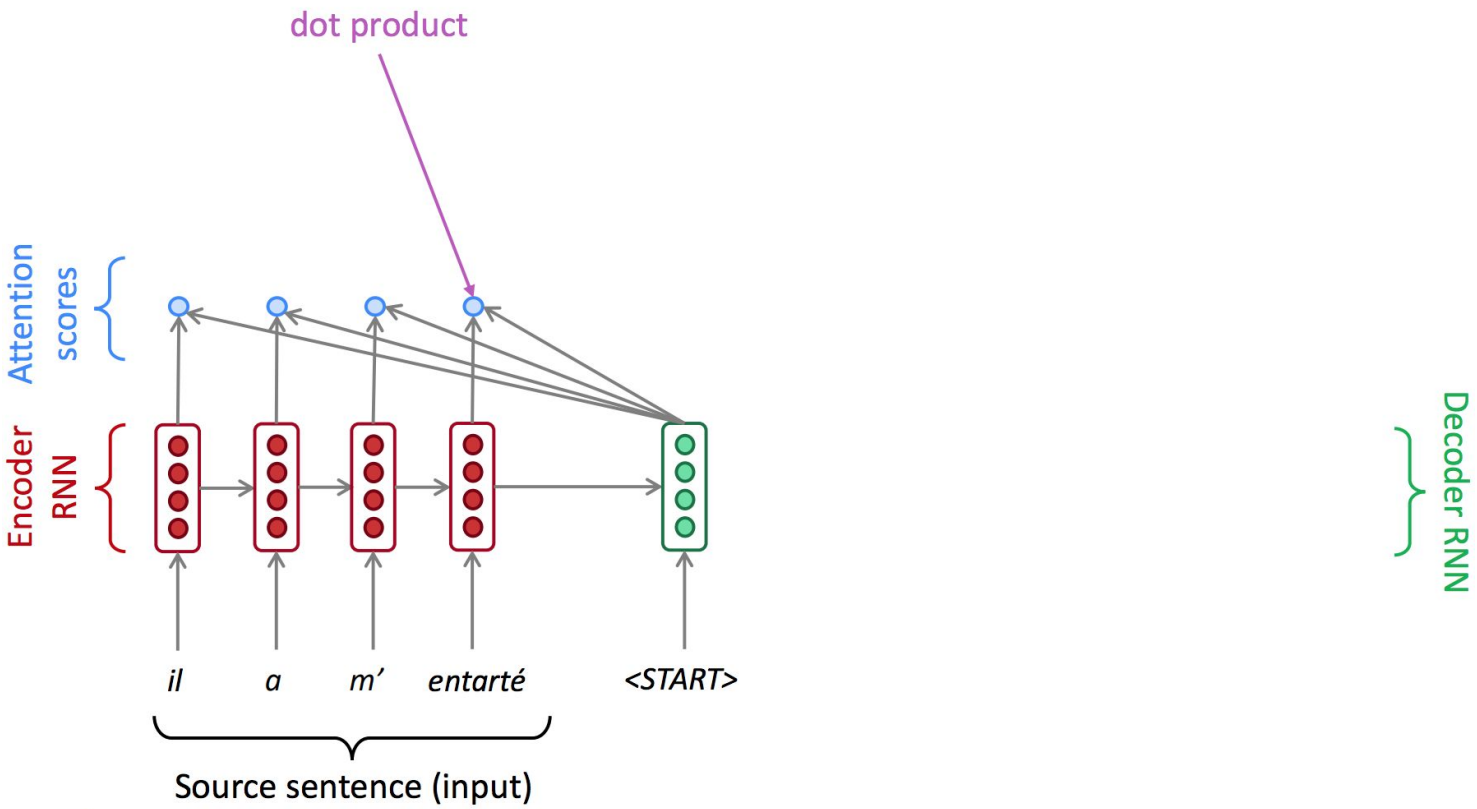


Seq2seq with attention

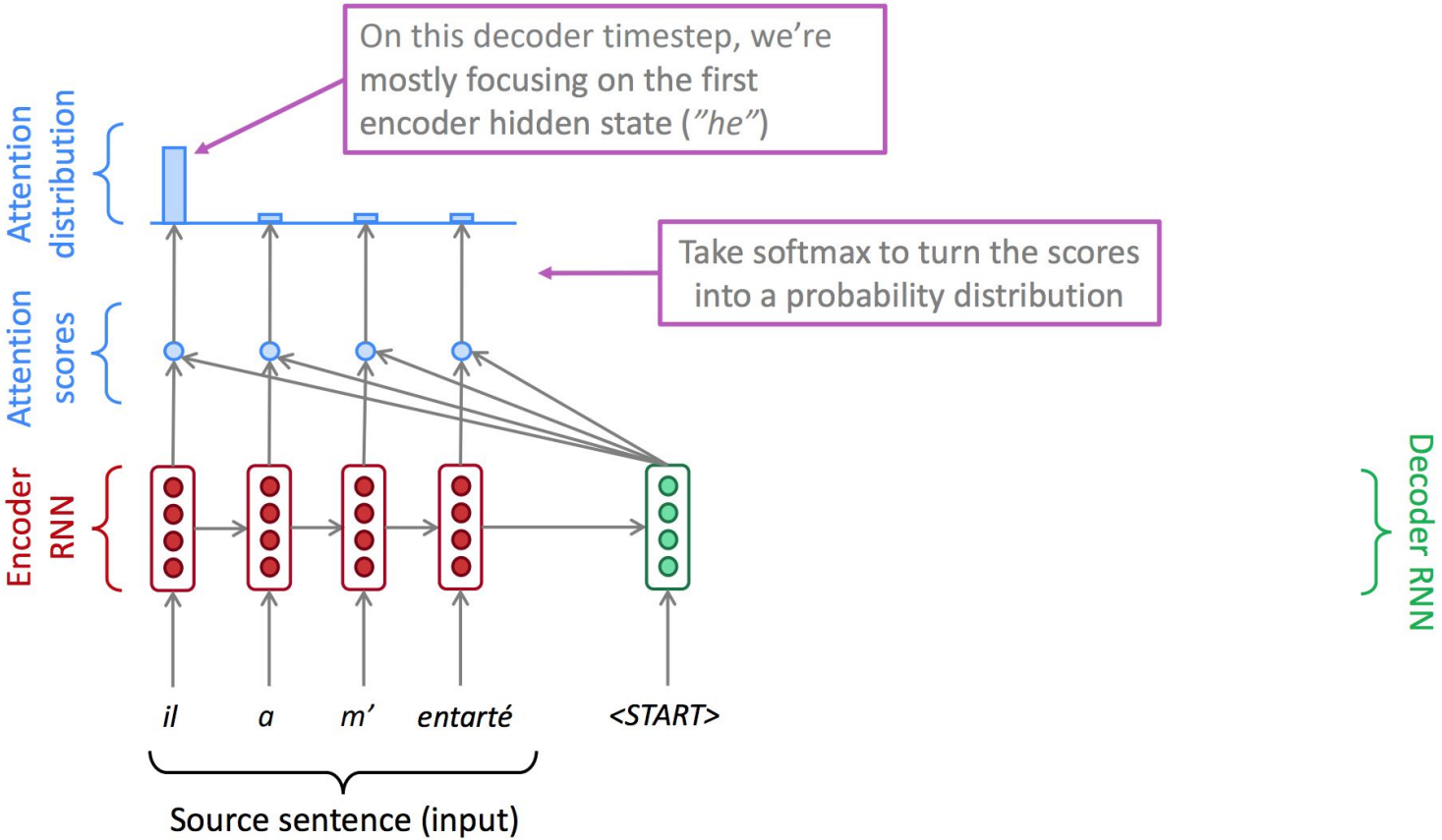


Decoder RNN

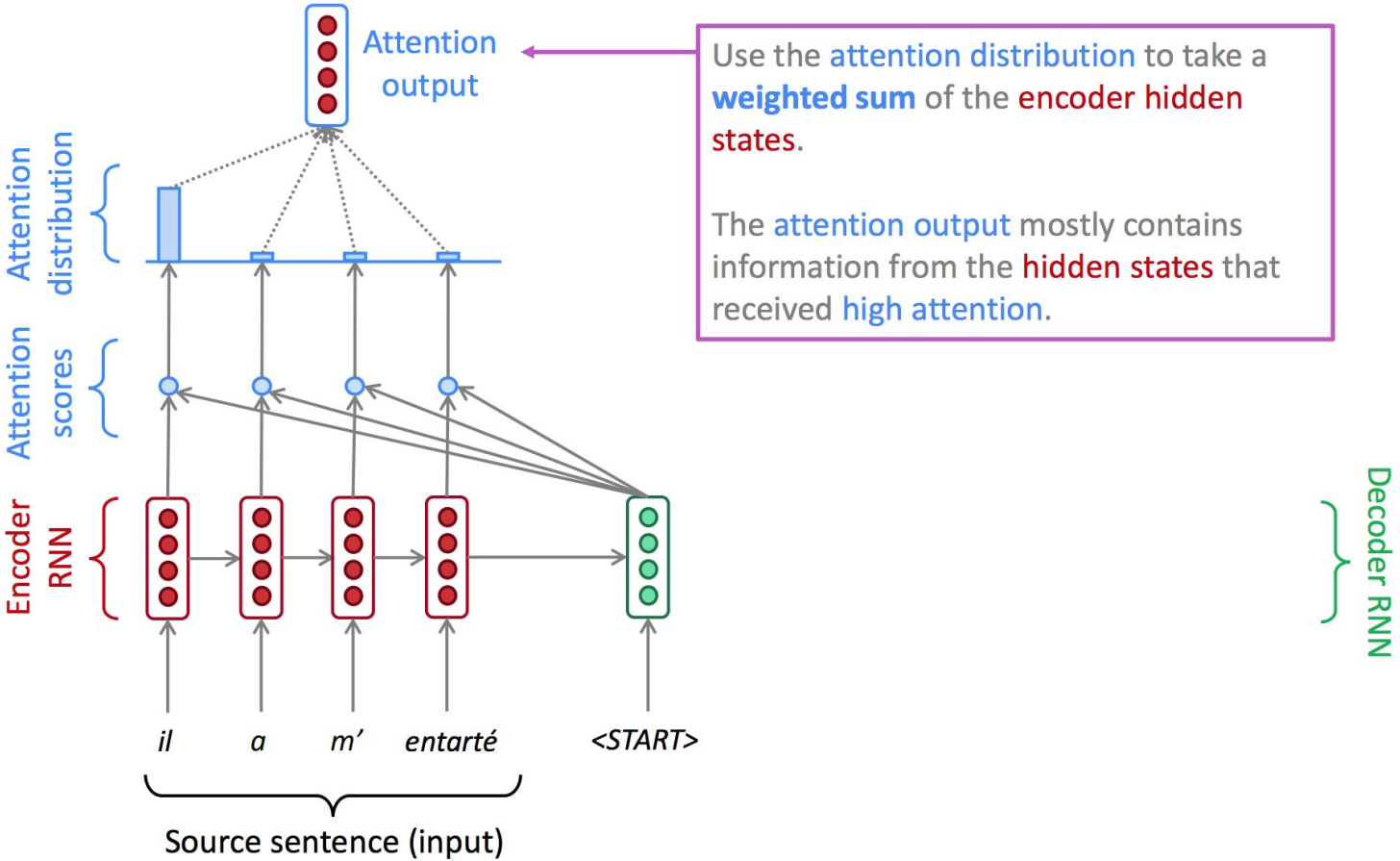
Seq2seq with attention



Seq2seq with attention



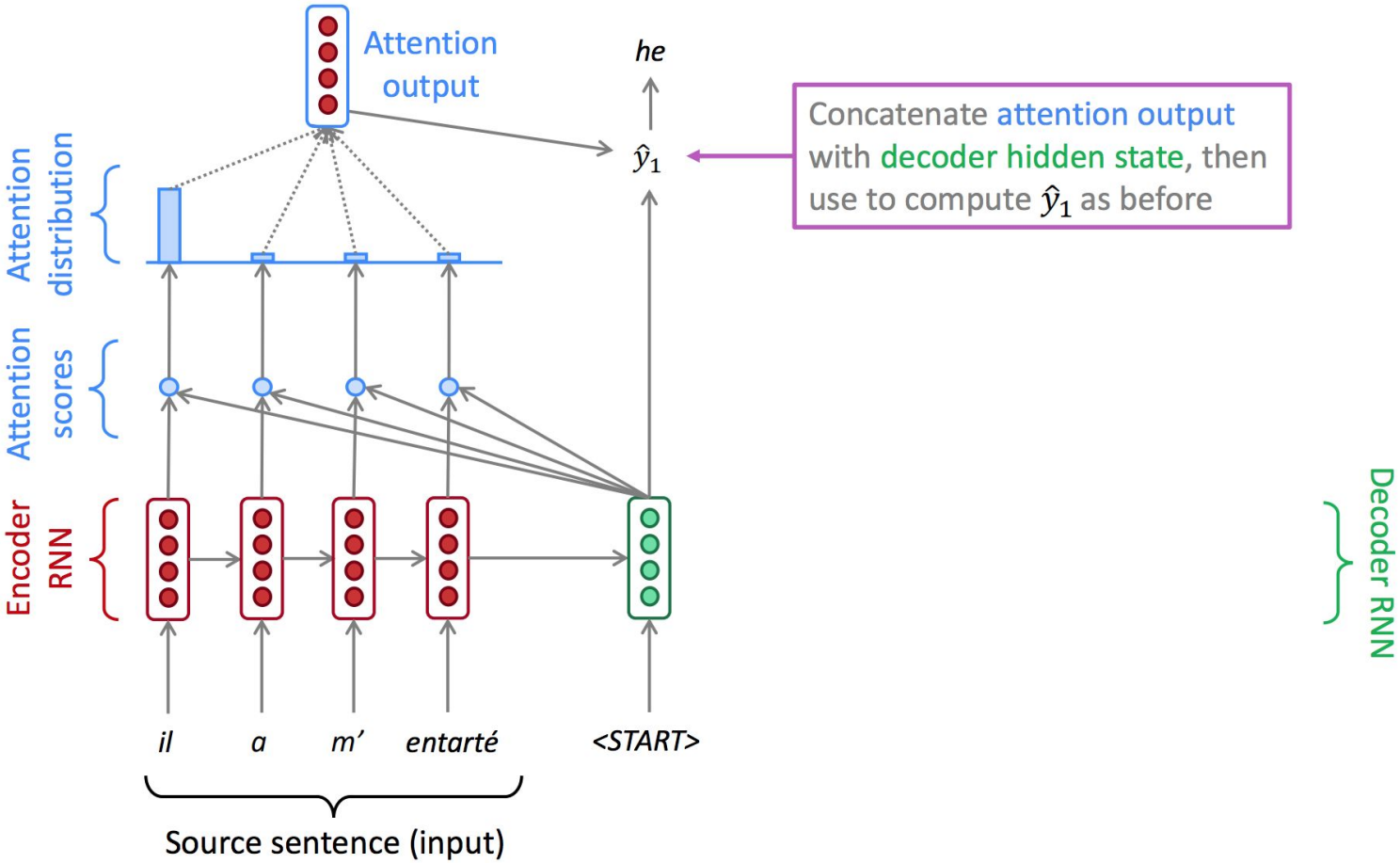
Seq2seq with attention



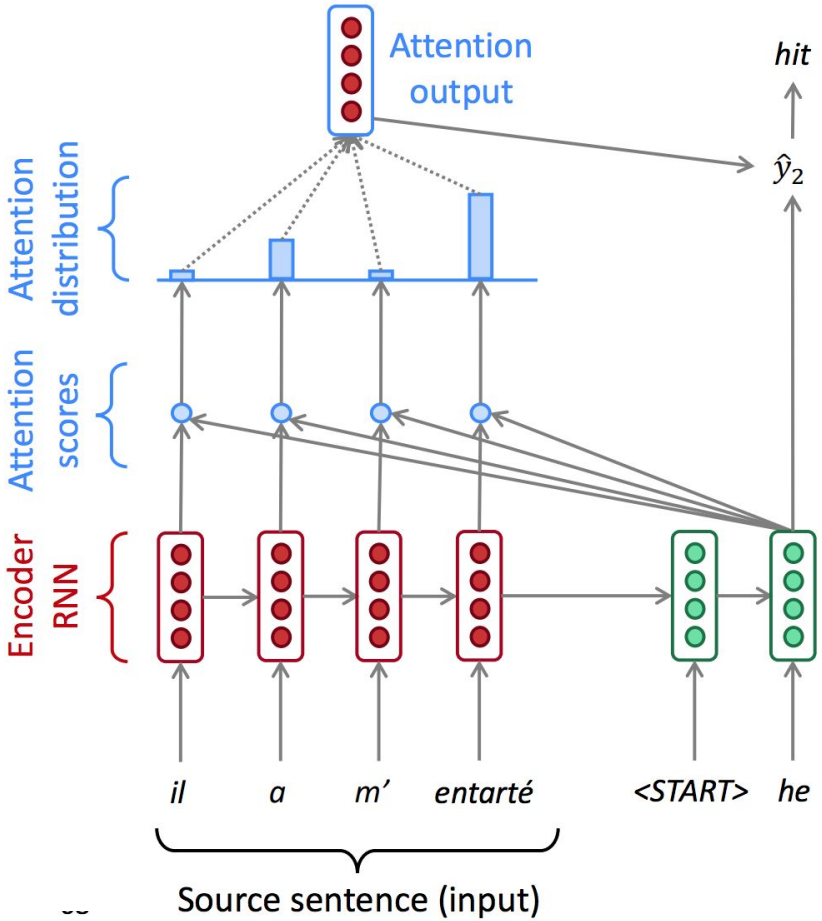
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

Seq2seq with attention



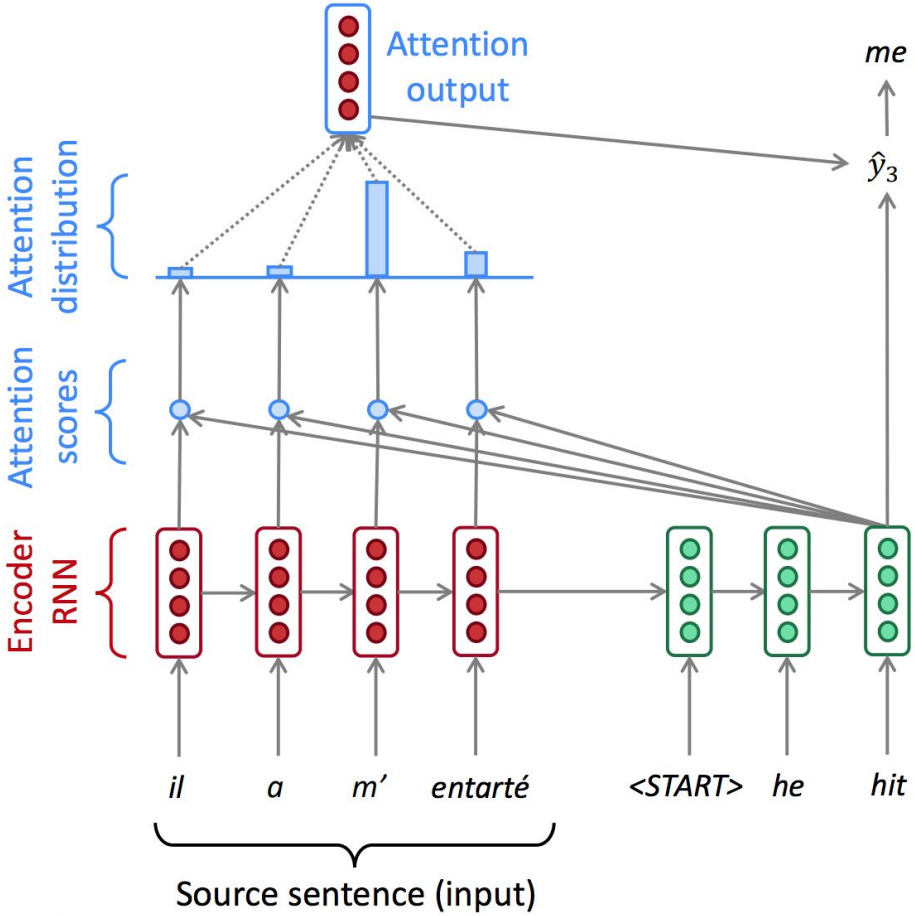
Seq2seq with attention



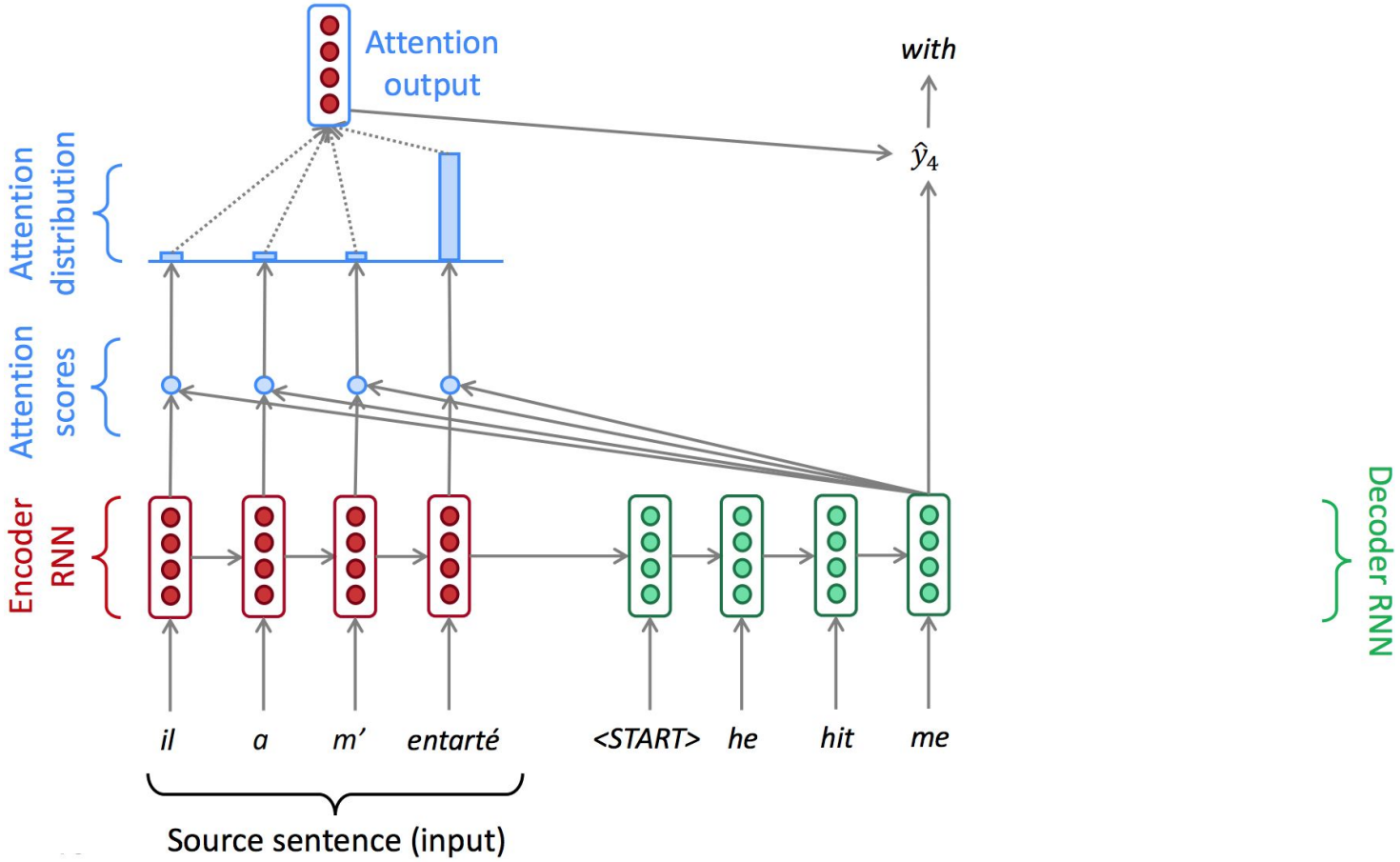
Sometimes we take the attention output from the previous step, and also feed it into the decoder (along with the usual decoder input). We do this in Assignment 4.

Decoder RNN

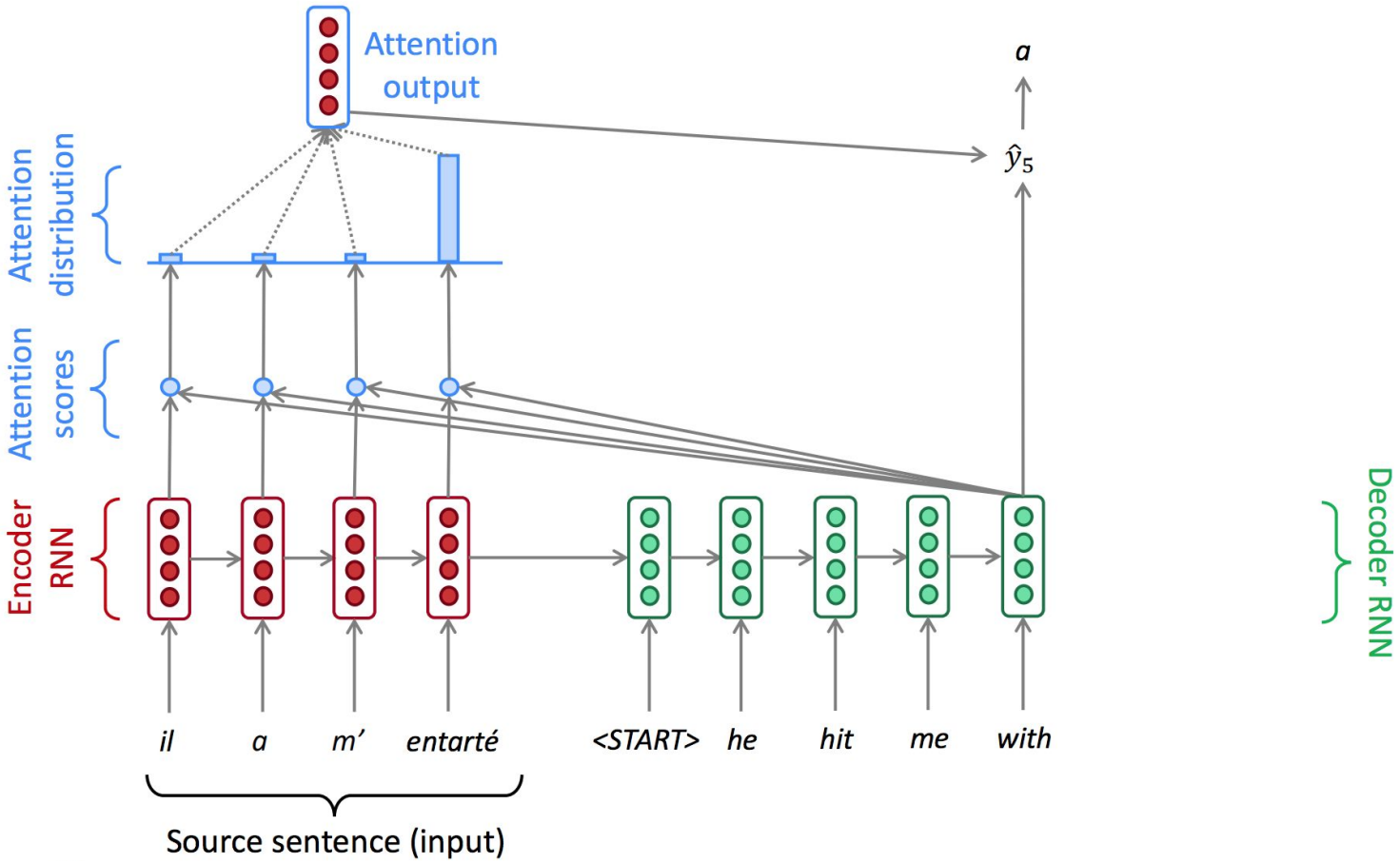
Seq2seq with attention



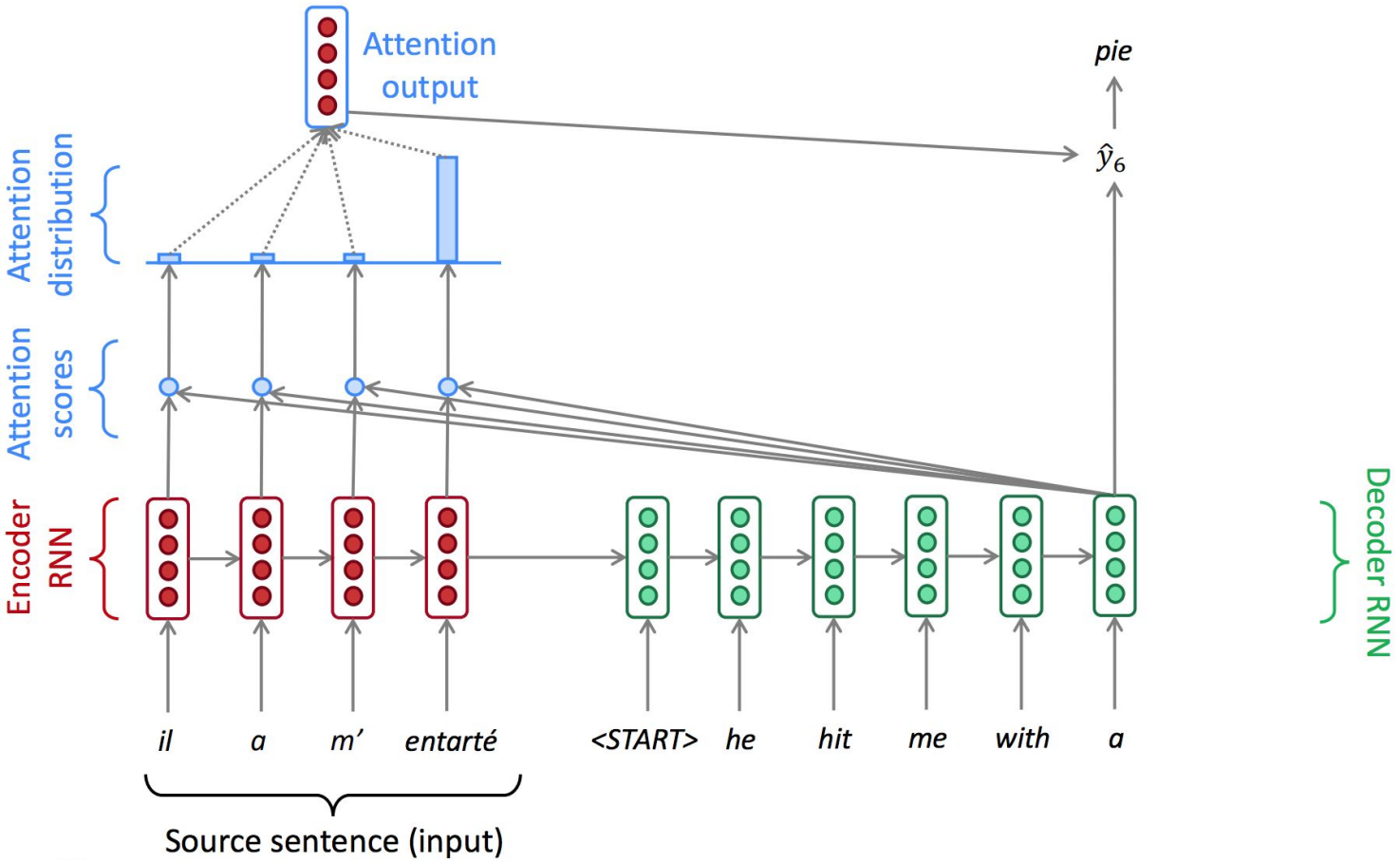
Seq2seq with attention



Seq2seq with attention



Seq2seq with attention



Attention in equations

We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$

On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$

We get the attention scores $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

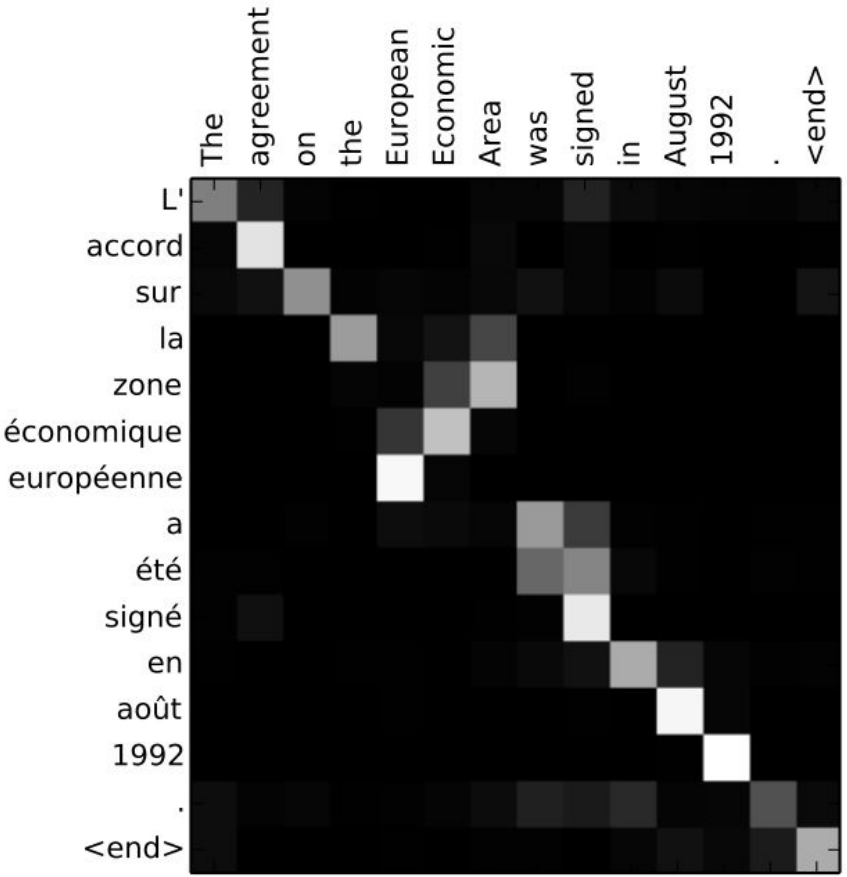
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Attention provides interpretability

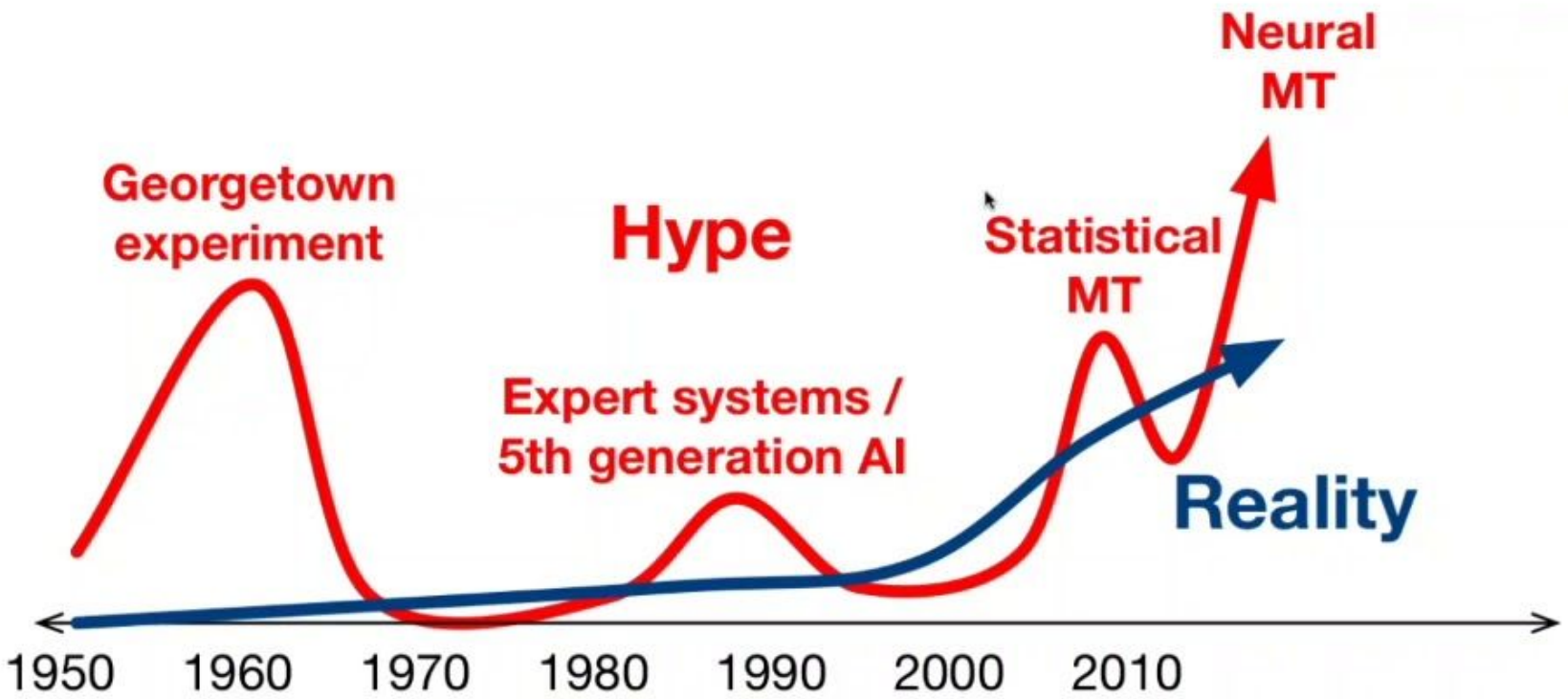
- We may see what the decoder was focusing on
- We get word alignment for free!



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