

ANLP

19 - seq2seq models

David Schlangen

University of Potsdam, MSc Cognitive Systems
Winter 2019 / 2020

your projects

- we will need both slots next week for your presentations
- so be prepared already to be able to present on Wednesday, 4-6pm
- I will prepare reading list for Ethics in NLP (and hope to organise full seminar on this in later semesters)

your projects

- minimum viable product / baseline: start with simplest approach you can think of (find in literature), then make this fancier / use NNs
- embrace (and analyse) failure. Some things will not work. Turn that into a learning experience:
 - Why did it not work? What kind of knowledge seems to be missing? Can I add this knowledge somehow, at least for testing? Can I change task so that knowledge is not needed?

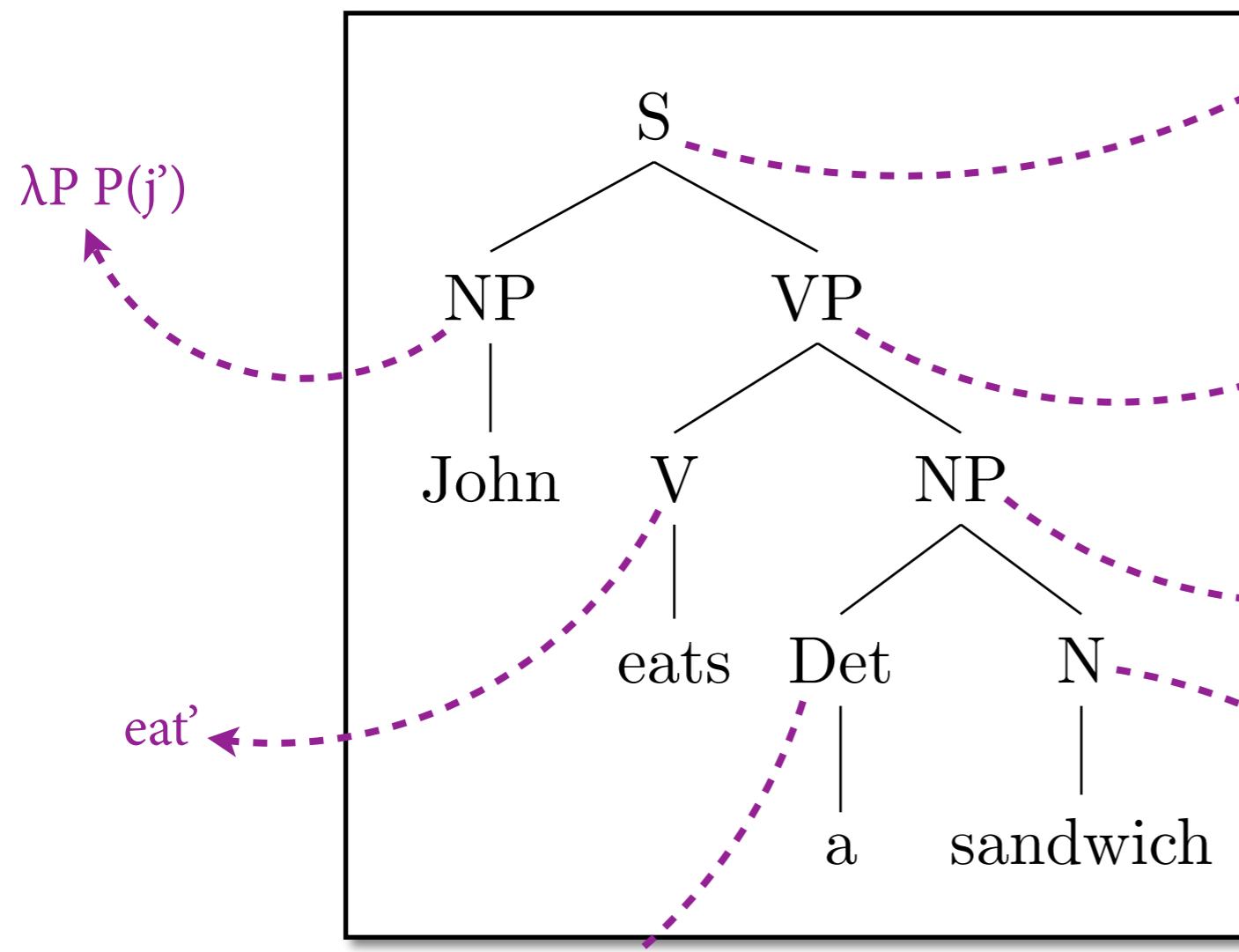
your projects

- think about the problem of evaluation. How do you know how well you've solved your problem?
- you may have to ask people what they think of the output of your model. This can be done in a principled way as well.
- document any external resources you've used (blogposts, tutorials, etc.)
- but do use external resources where they make sense (nltk, StanfordNLP, spacy, etc.)

where are we?

- representations, and how to compute them:
 - of words (nodes in graph; word vectors)
 - of texts (bag of words; features)
 - of sentences
 - LSTMs (vectors)
 - syntactic structure + semantic representation

Example



$\lambda P \lambda Q \exists x P(x) \wedge Q(x)$

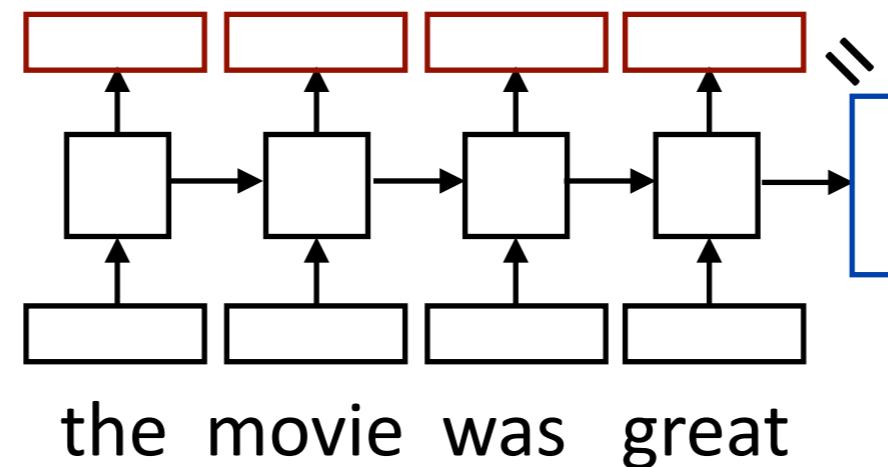
$$\begin{aligned}
 & (\lambda P P(j')) (\lambda y \exists x sw'(x) \wedge eat'(y)(x)) \\
 \rightarrow_{\beta} & (\lambda y \exists x sw'(x) \wedge eat'(y)(x))(j') \\
 \rightarrow_{\beta} & \exists x sw'(x) \wedge eat'(j')(x)
 \end{aligned}$$

$$\begin{aligned}
 & \lambda y (\lambda Q \exists x sw'(x) \wedge Q(x))(eat'(y)) \\
 \rightarrow_{\beta} & \lambda y \exists x sw'(x) \wedge eat'(y)(x)
 \end{aligned}$$

$$\begin{aligned}
 & (\lambda P \lambda Q \exists x P(x) \wedge Q(x))(sw') \\
 \rightarrow_{\beta} & \lambda Q \exists x sw'(x) \wedge Q(x)
 \end{aligned}$$

sw'

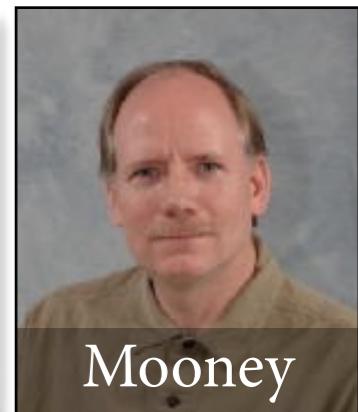
What do RNNs produce?



- ▶ **Encoding of the sentence** — can pass this to a decoder or make a classification decision about the sentence
- ▶ **Encoding of each word** — can pass this to another layer to make a prediction (can also pool these to get a different sentence encoding)
- ▶ RNN can be viewed as a transformation of a sequence of vectors into a sequence of context-dependent vectors

is that a meaning representation?

“You can’t cram the meaning of a whole sentence into a single vector!”



Mooney

Ray Mooney 2014, <https://yoavartzi.com/sp14/slides/mooney.sp14.pdf>

Overview

Today we will:

- Introduce a new task: Machine Translation



is a major use-case of

- Introduce a new neural architecture: sequence-to-sequence



is improved by

- Introduce a new neural technique: attention

Section 1: Pre-Neural Machine Translation

Machine Translation

Machine Translation (MT) is the task of translating a sentence x from one language (the **source language**) to a sentence y in another language (the **target language**).

$x:$ *L'homme est né libre, et partout il est dans les fers*



$y:$ *Man is born free, but everywhere he is in chains*

- Rousseau

1950s: Early Machine Translation

Machine Translation research began in the **early** 1950s.

- Russian → English
(motivated by the Cold War!)



1 minute video showing 1954 MT:
<https://youtu.be/K-HfpsHPmvw>

- Systems were mostly **rule-based**, using a bilingual dictionary to map Russian words to their English counterparts

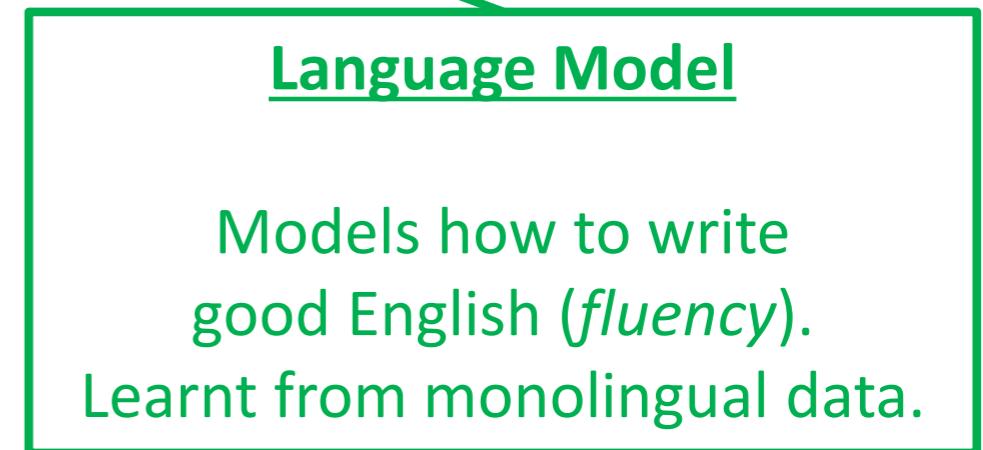
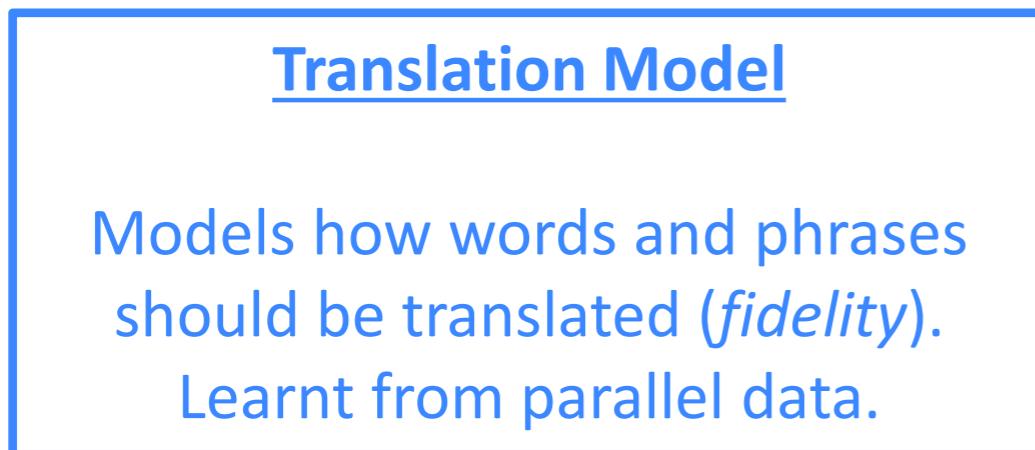
1990s-2010s: Statistical Machine Translation

- Core idea: Learn a **probabilistic model** from **data**
- Suppose we're translating French → English.
- We want to find **best English sentence y , given French sentence x**

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

- Use Bayes Rule to break this down into **two components** to be learnt separately:

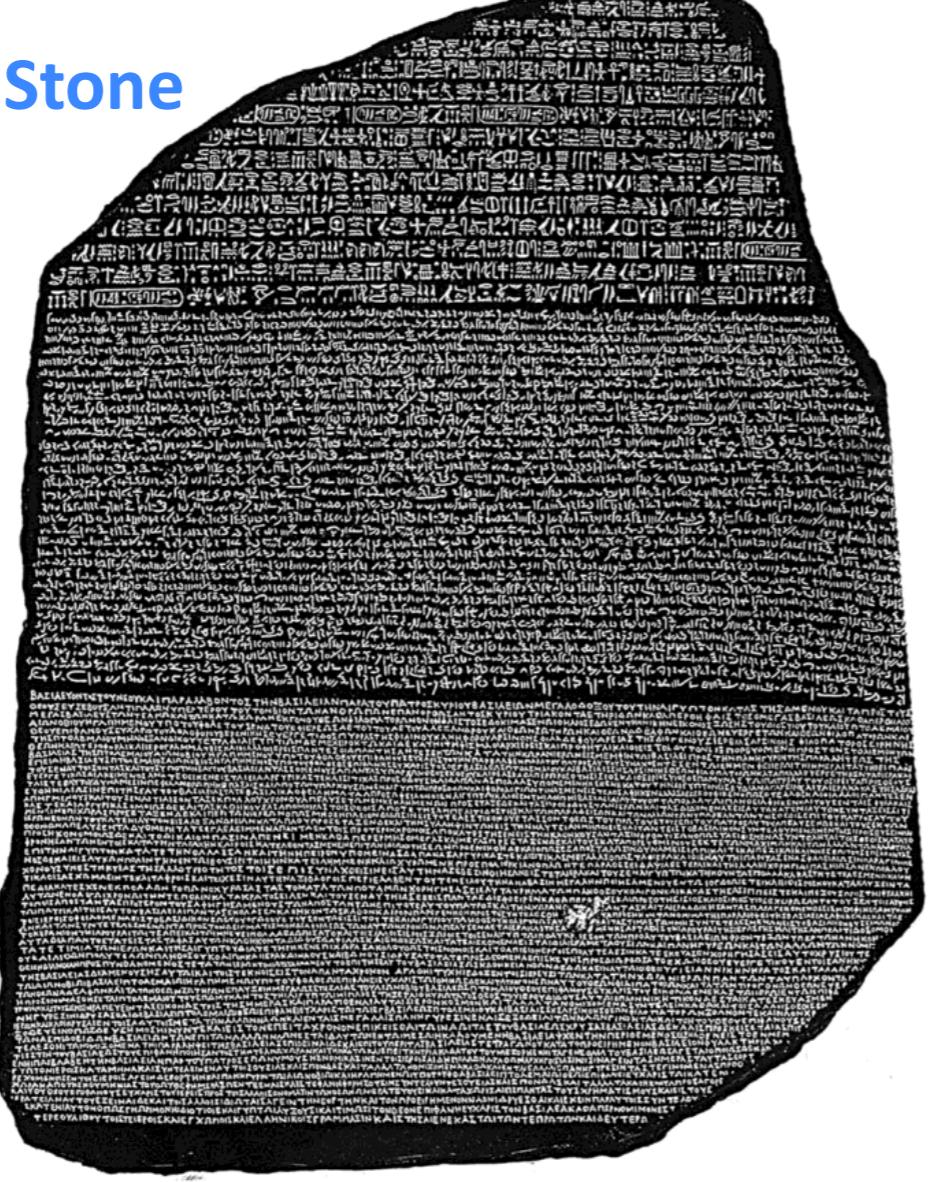
$$= \operatorname{argmax}_y P(x|y)P(y)$$



1990s-2010s: Statistical Machine Translation

- Question: How to learn translation model $P(x|y)$?
- First, need large amount of **parallel data**
(e.g. pairs of human-translated French/English sentences)

The Rosetta Stone



Ancient Egyptian

Demotic

Ancient Greek

Learning alignment for SMT

- Question: How to learn translation model $P(x|y)$ from the parallel corpus?
- Break it down further: we actually want to consider

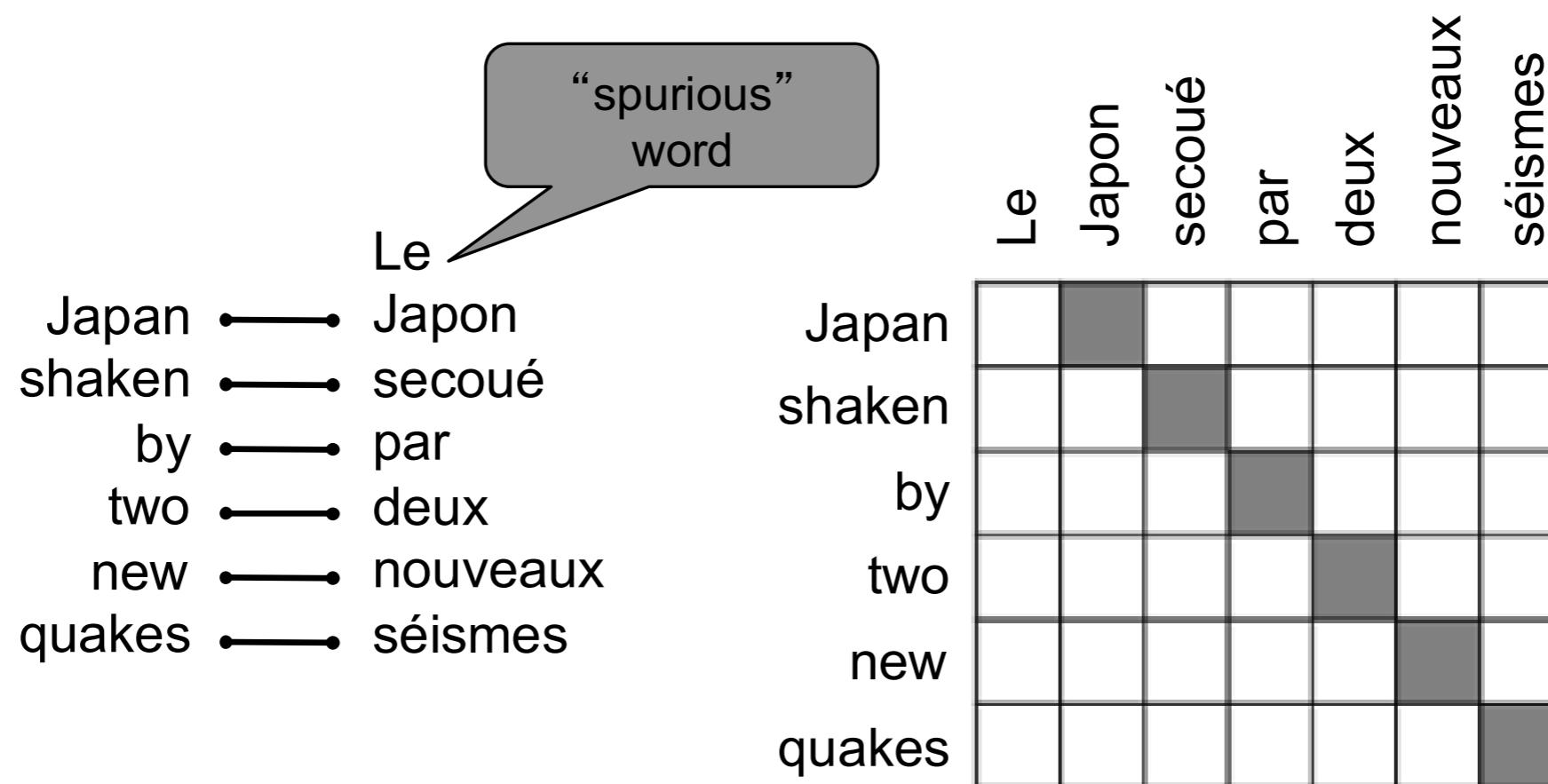
$$P(x, a|y)$$

where a is the **alignment**, i.e. word-level correspondence between French sentence x and English sentence y

What is alignment?

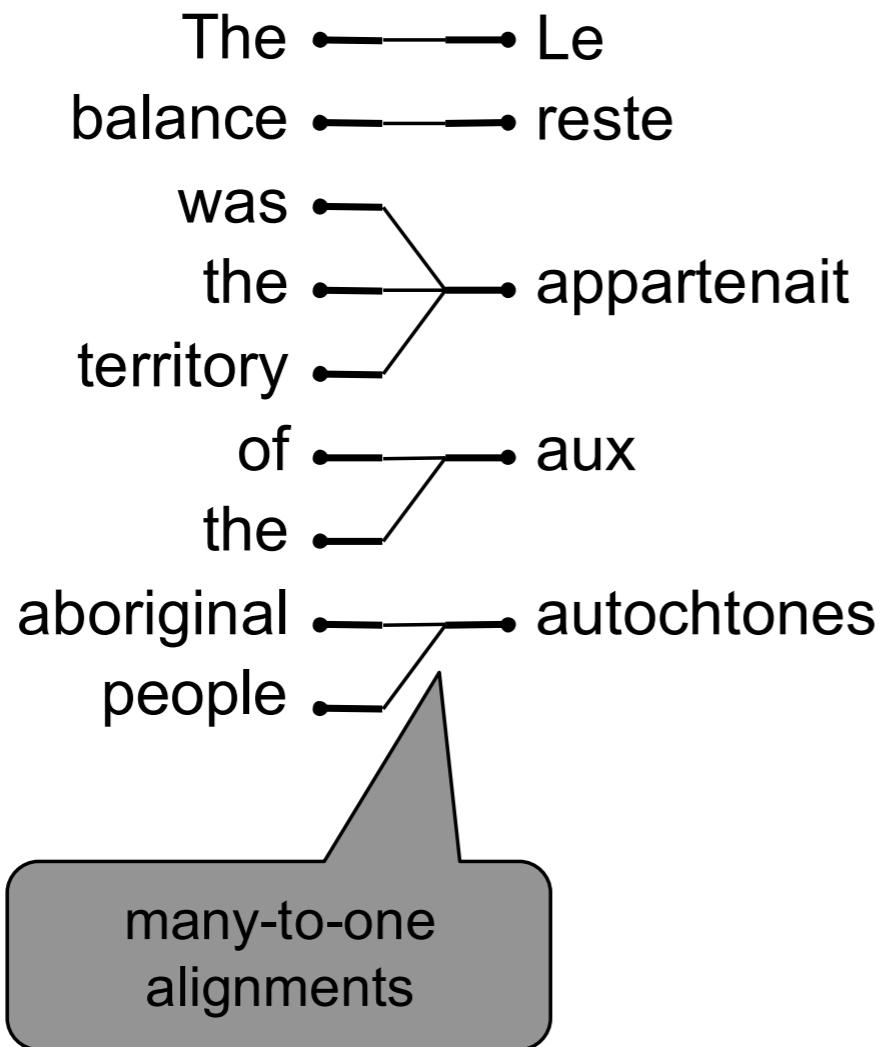
Alignment is the correspondence between particular words in the translated sentence pair.

- Note: Some words have no counterpart



Alignment is complex

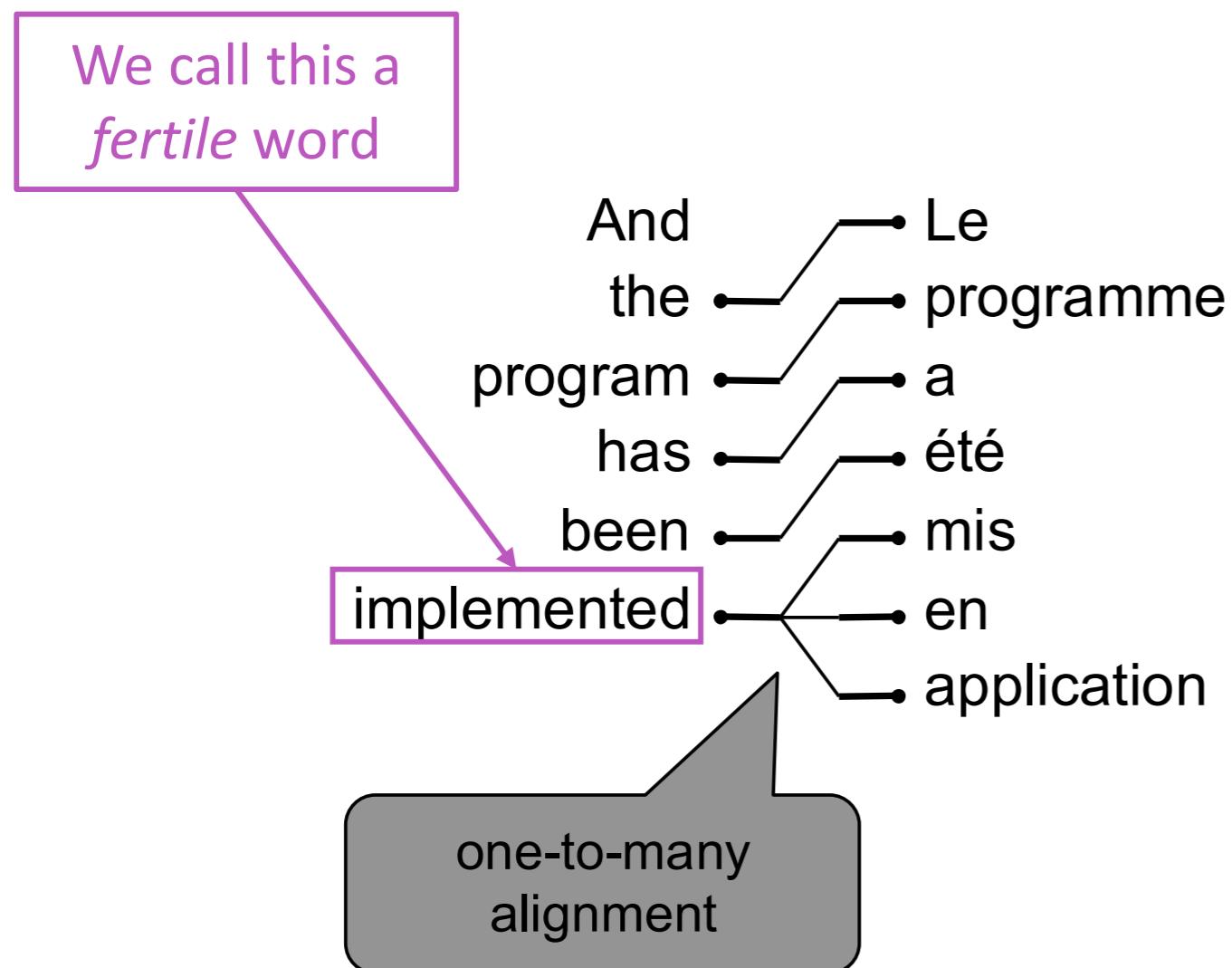
Alignment can be many-to-one



	Le	reste	appartenait	aux	autochtones
The	■				
balance		■			
was			■		
the				■	
territory					■
of				■	
the					■
aboriginal					■
people					■

Alignment is complex

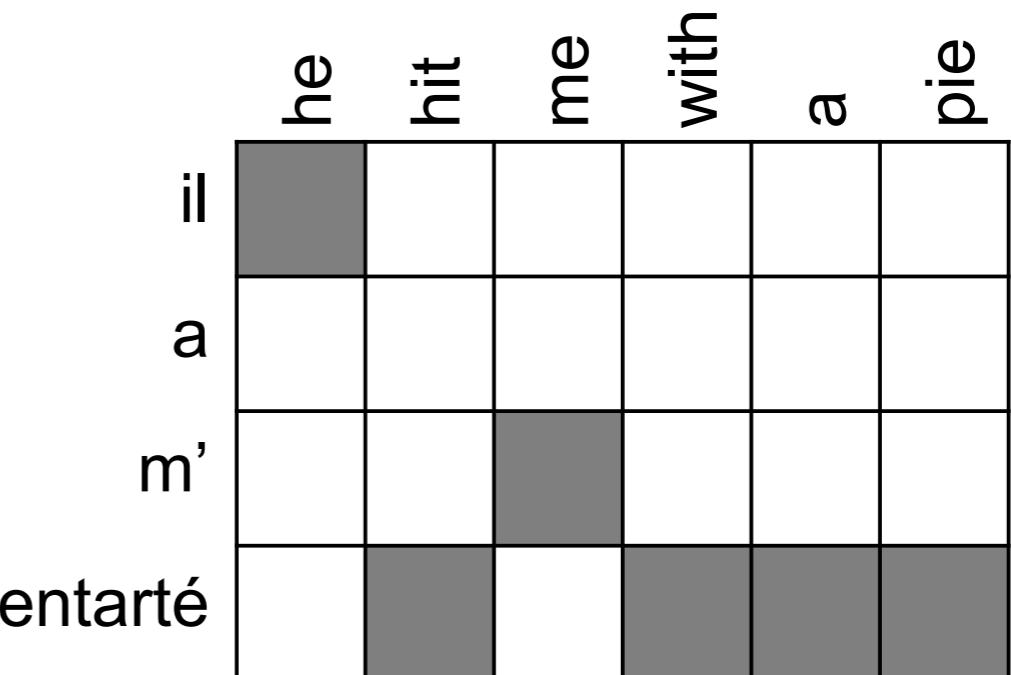
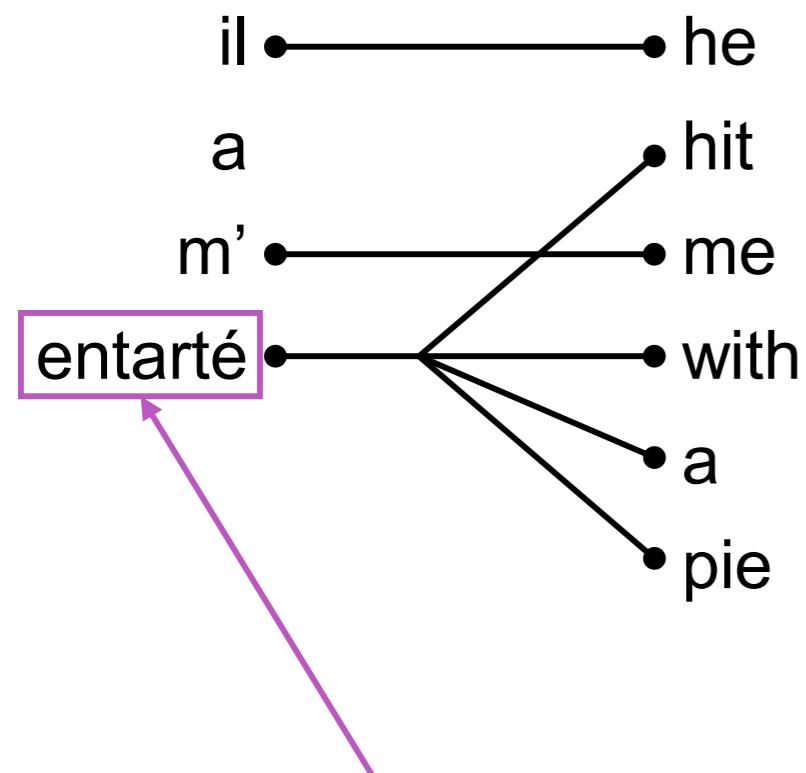
Alignment can be one-to-many



Le	programme					
a		été				
program			mis			
has			en			
been			application			
implemented						

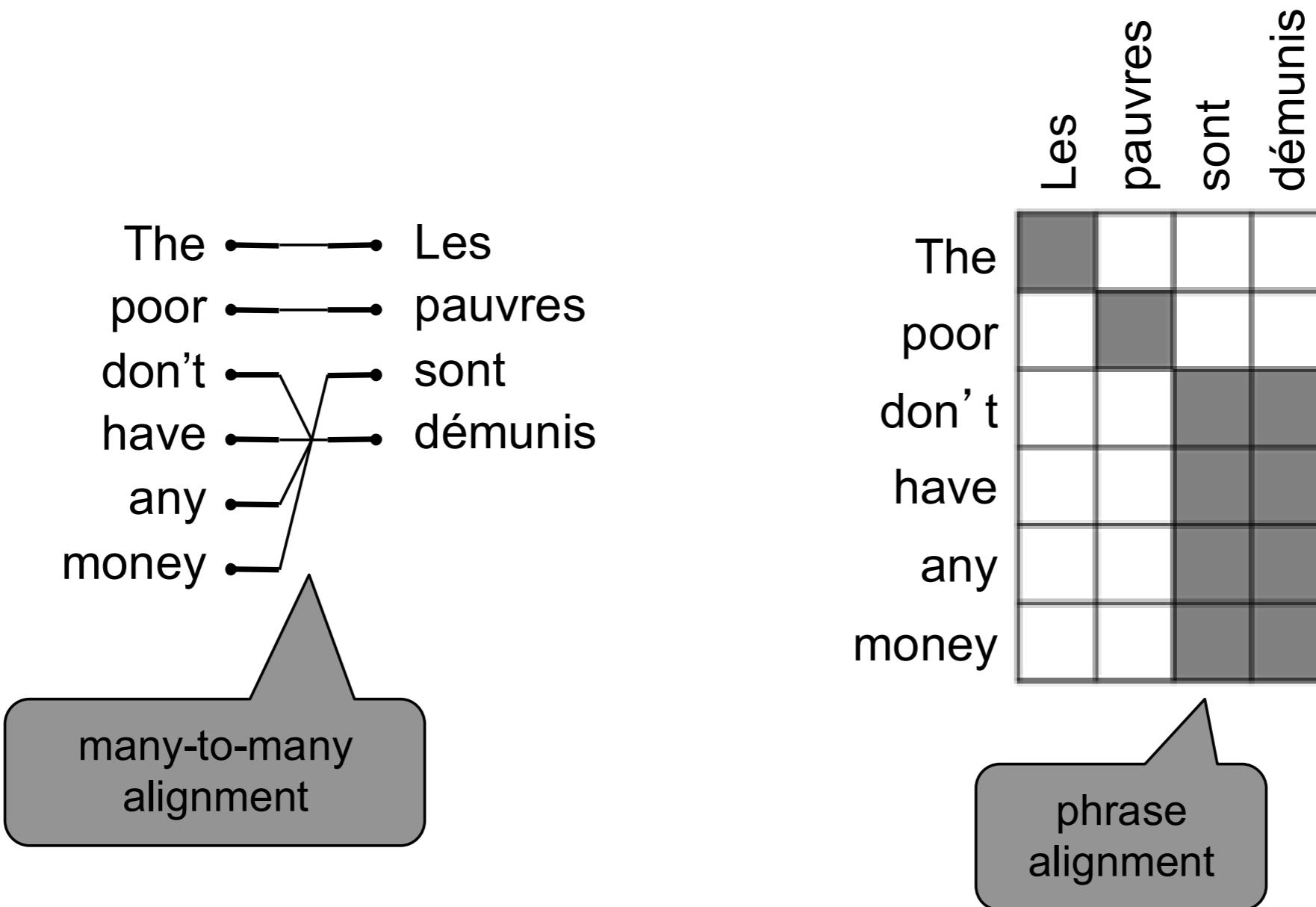
Alignment is complex

Some words are very fertile!



Alignment is complex

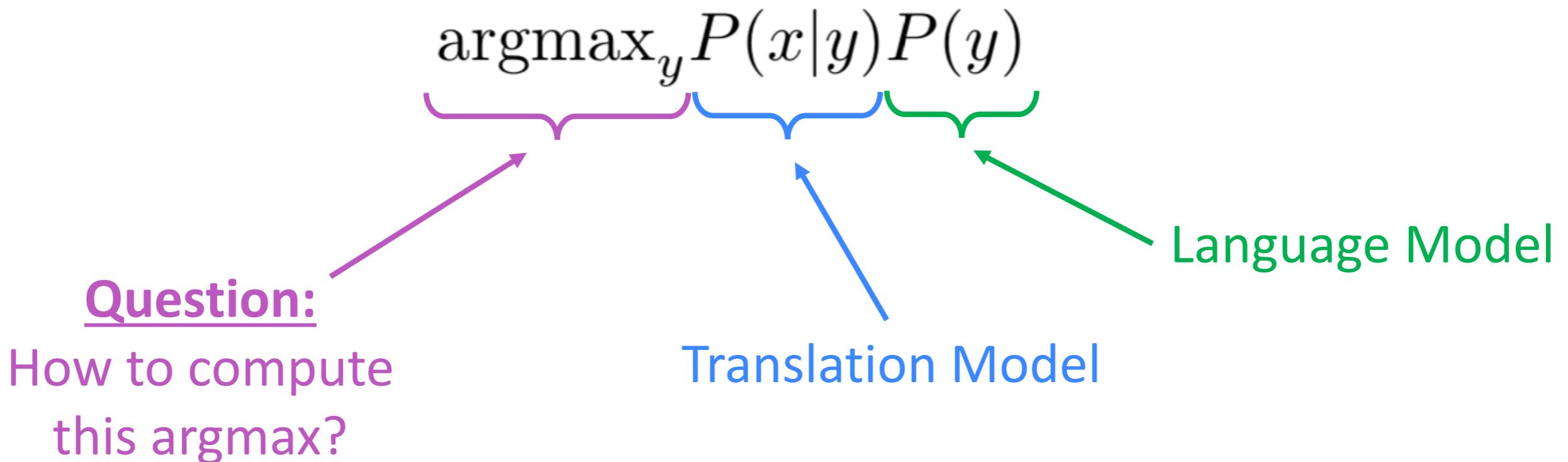
Alignment can be many-to-many (phrase-level)



Learning alignment for SMT

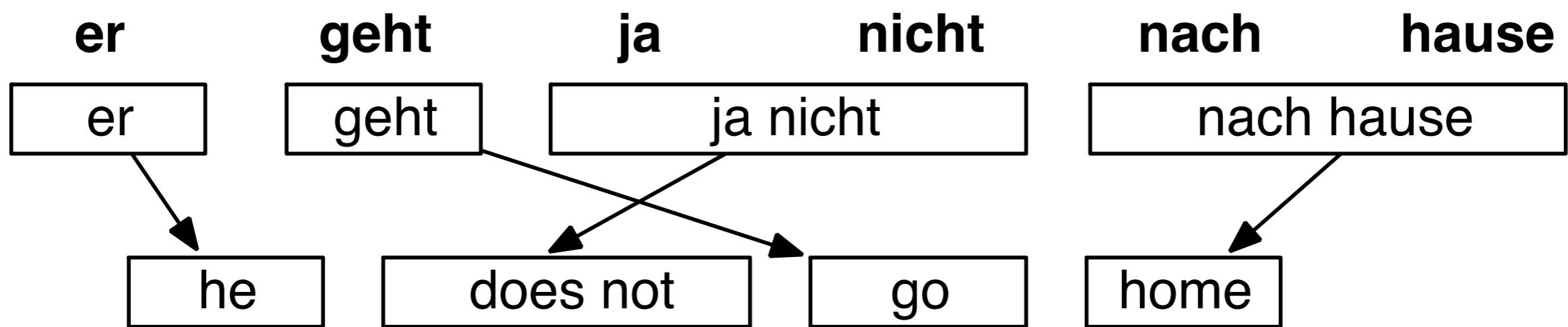
- We learn $P(x, a|y)$ as a combination of many factors, including:
 - Probability of particular words aligning (also depends on position in sent)
 - Probability of particular words having particular fertility (number of corresponding words)
 - etc.

Decoding for SMT



- We could enumerate every possible y and calculate the probability? → Too expensive!
- **Answer:** Use a **heuristic search algorithm** to **search for the best translation**, discarding hypotheses that are too low-probability
- This process is called *decoding*

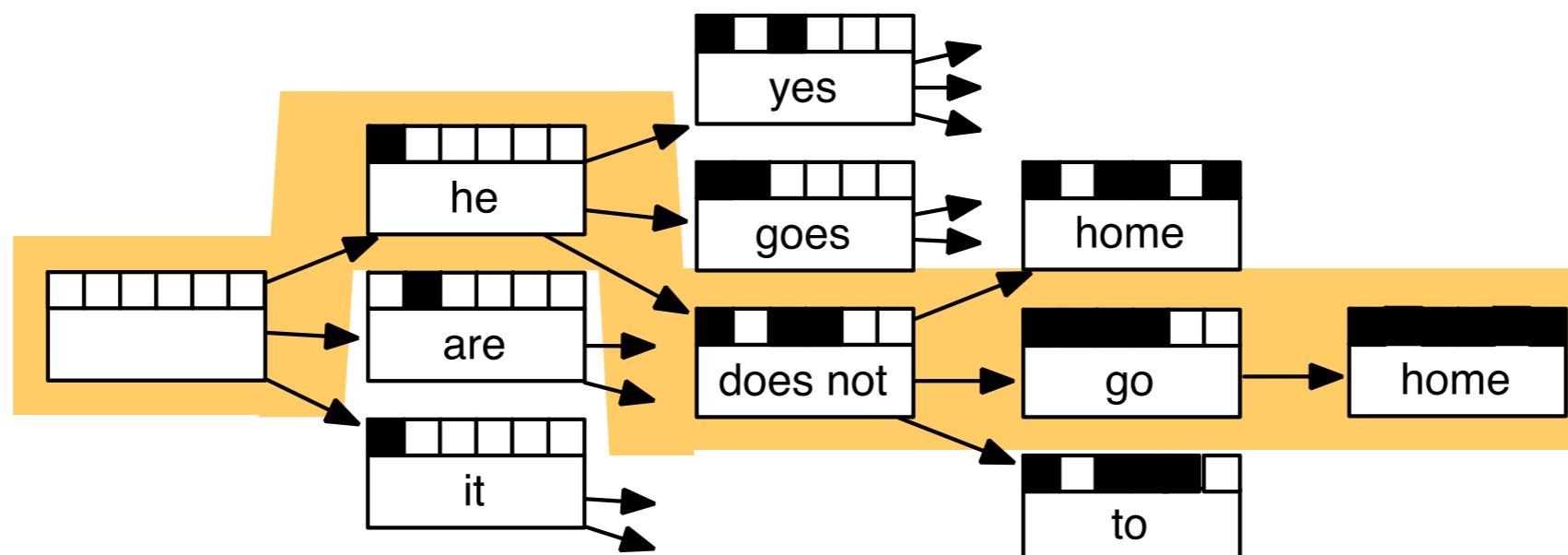
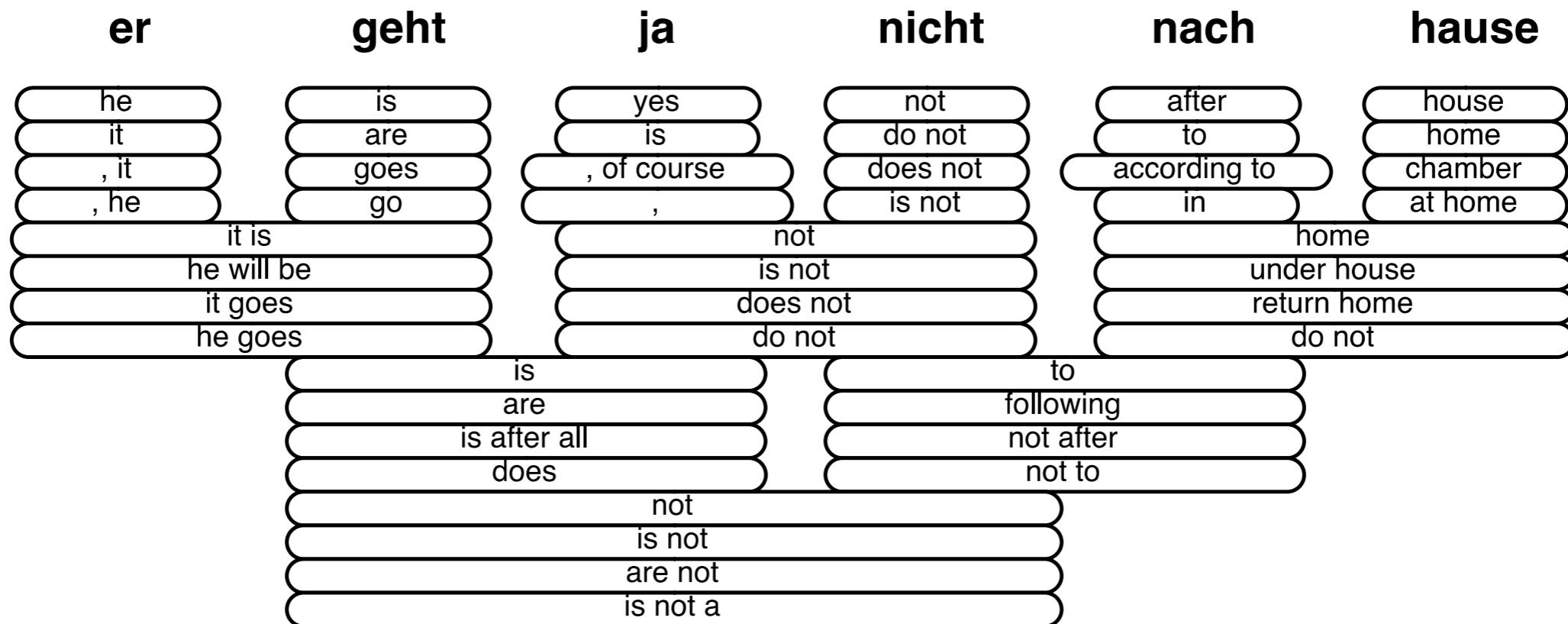
Decoding for SMT



Source: "Statistical Machine Translation", Chapter 6, Koehn, 2009.

<https://www.cambridge.org/core/books/statistical-machine-translation/94EADF9F680558E13BE759997553CDE5>

Decoding for SMT



Source: "Statistical Machine Translation", Chapter 6, Koehn, 2009.

<https://www.cambridge.org/core/books/statistical-machine-translation/94EADF9F680558E13BE759997553CDE5>

1990s-2010s: Statistical Machine Translation

- SMT was a huge research field
- The best systems were extremely complex
 - Hundreds of important details we haven't mentioned here
 - 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
 - Like tables of equivalent phrases
 - Lots of human effort to maintain
 - Repeated effort for each language pair!

Section 2: Neural Machine Translation

2014

(dramatic reenactment)

2014

MT research

*Neural
Machine
Translation*

(dramatic reenactment)

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*.

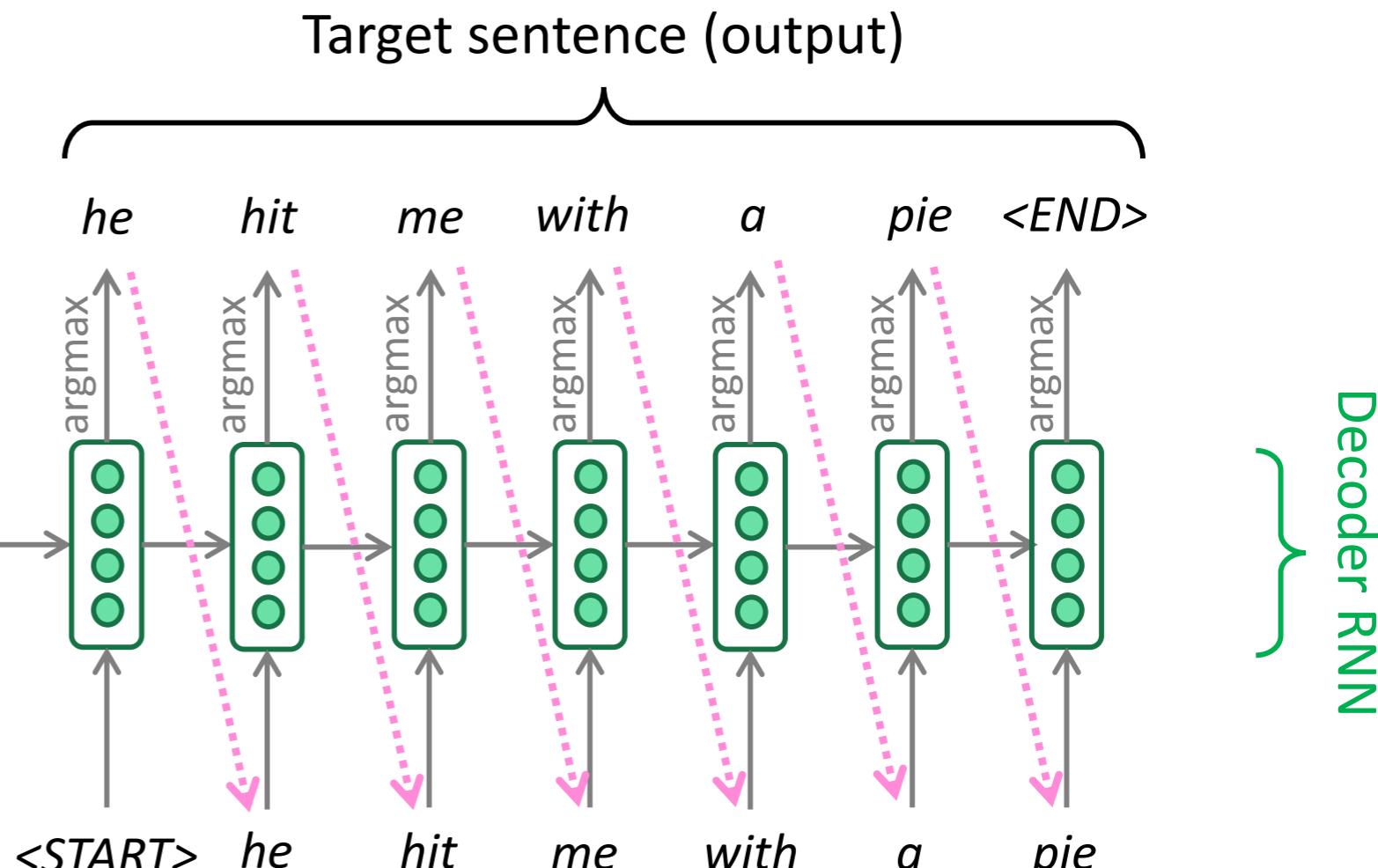
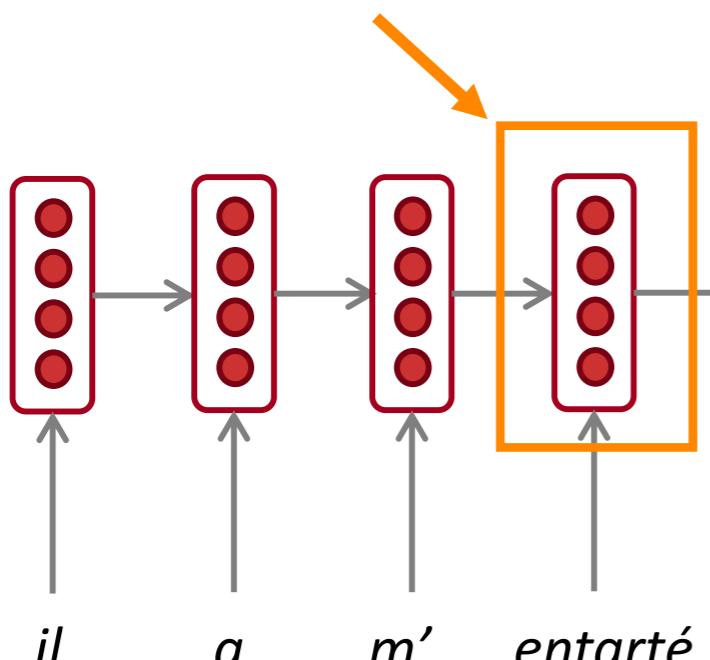
Neural Machine Translation (NMT)

The sequence-to-sequence model

Encoding of the source sentence.

Provides initial hidden state

for Decoder RNN.



Encoder RNN produces an encoding of the source sentence.

Decoder RNN is a Language Model that generates target sentence, *conditioned on encoding*.

Note: This diagram shows test time behavior: decoder output is fed in as next step's input

Sequence-to-sequence is versatile!

- Sequence-to-sequence is useful for *more than just MT*
- Many NLP tasks can be phrased as sequence-to-sequence:
 - **Summarization** (long text → short text)
 - **Dialogue** (previous utterances → next utterance)
 - **Parsing** (input text → output parse as sequence)
 - **Code generation** (natural language → Python code)

Neural Machine Translation (NMT)

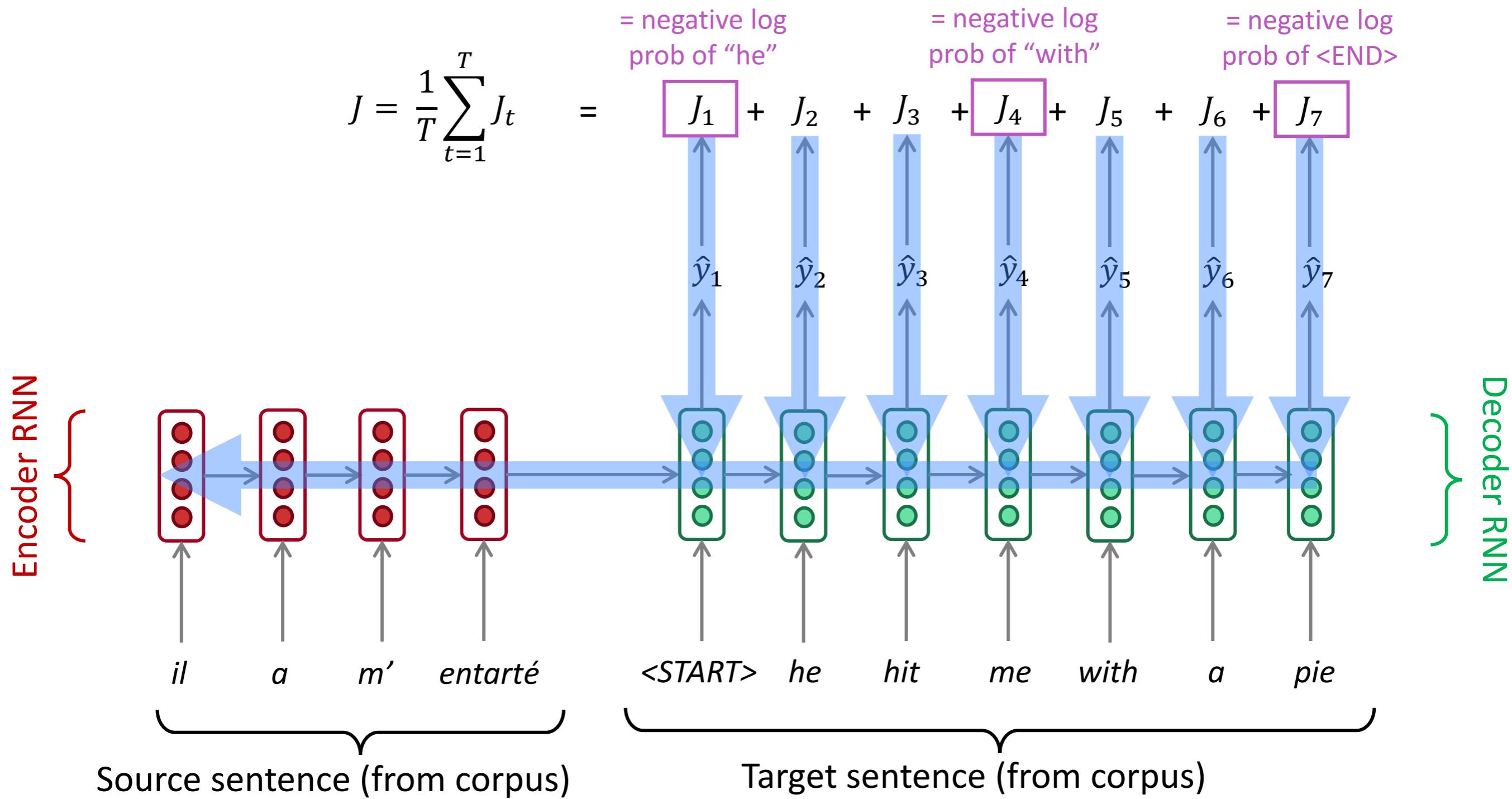
- The **sequence-to-sequence** model is an example of a **Conditional Language Model**.
 - **Language Model** because the decoder is predicting the next word of the target sentence y
 - **Conditional** because its predictions are *also* conditioned on the source sentence x
- NMT directly calculates $P(y|x)$:

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

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

- **Question:** How to **train** a NMT system?
- **Answer:** Get a big parallel corpus...

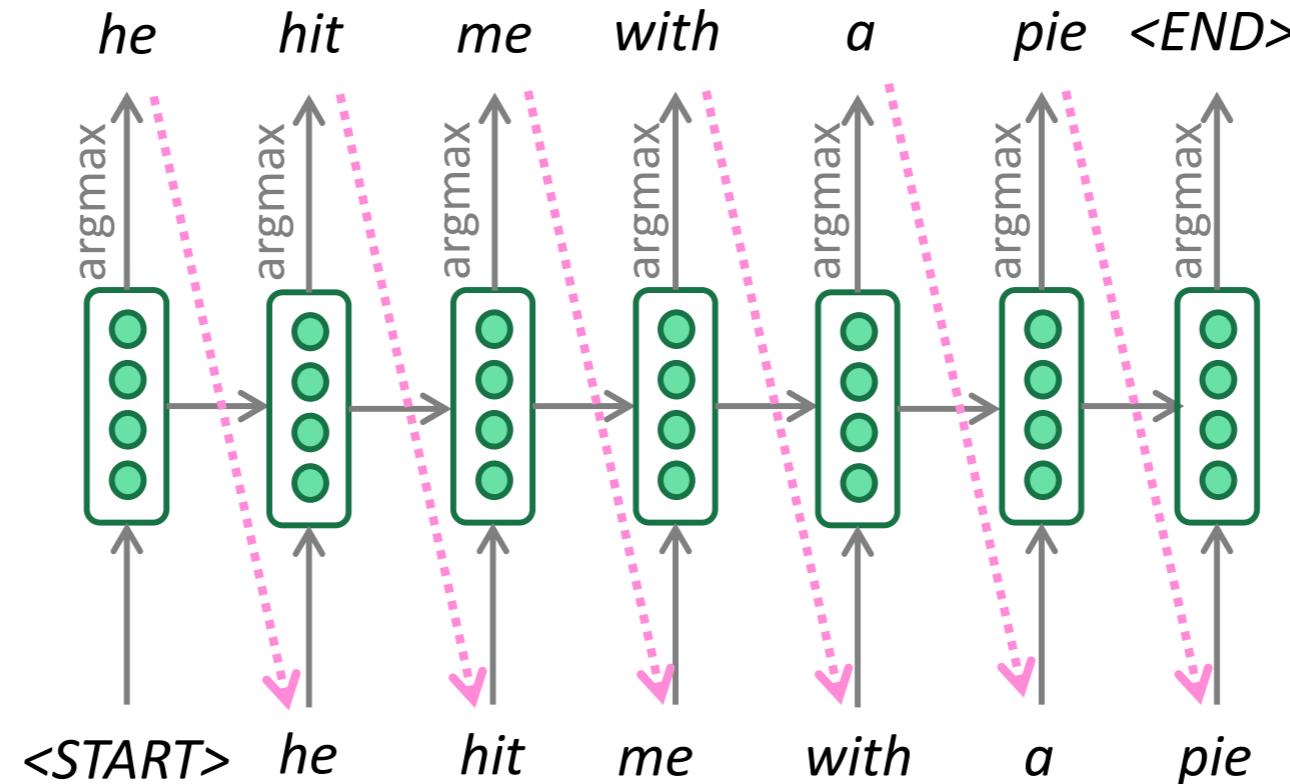
Training a Neural Machine Translation system



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)
- Problems with this method?**

Problems with greedy decoding

- 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...)
- How to fix this?

Exhaustive search decoding

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

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

- We could try computing all possible sequences y
 - This means that on each step t of the decoder, we're tracking V^t possible partial translations, where V is vocab size
 - This $O(V^T)$ complexity is far too expensive!

Beam search decoding

- Core idea: 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 y_1, \dots, y_t has a score which is its log probability:
$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$
 - Scores are all negative, and higher score is better
 - We search for high-scoring hypotheses, tracking top k on each step
- Beam search is not guaranteed to find optimal solution
- But much more efficient than exhaustive search!

Beam search decoding: example

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

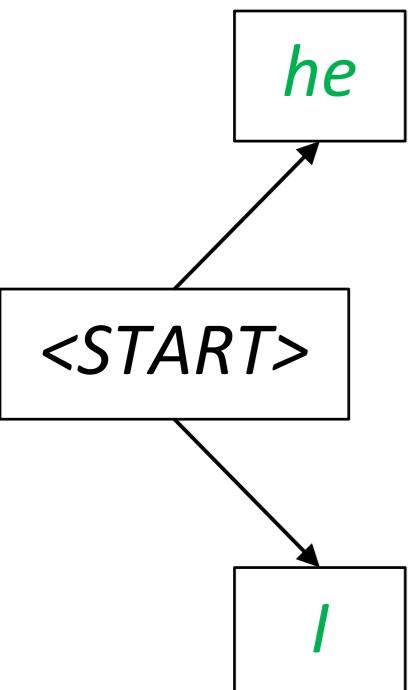
<START>

Calculate prob
dist of next word

Beam search decoding: example

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

$$-0.7 = \log P_{\text{LM}}(he | <\text{START}>)$$

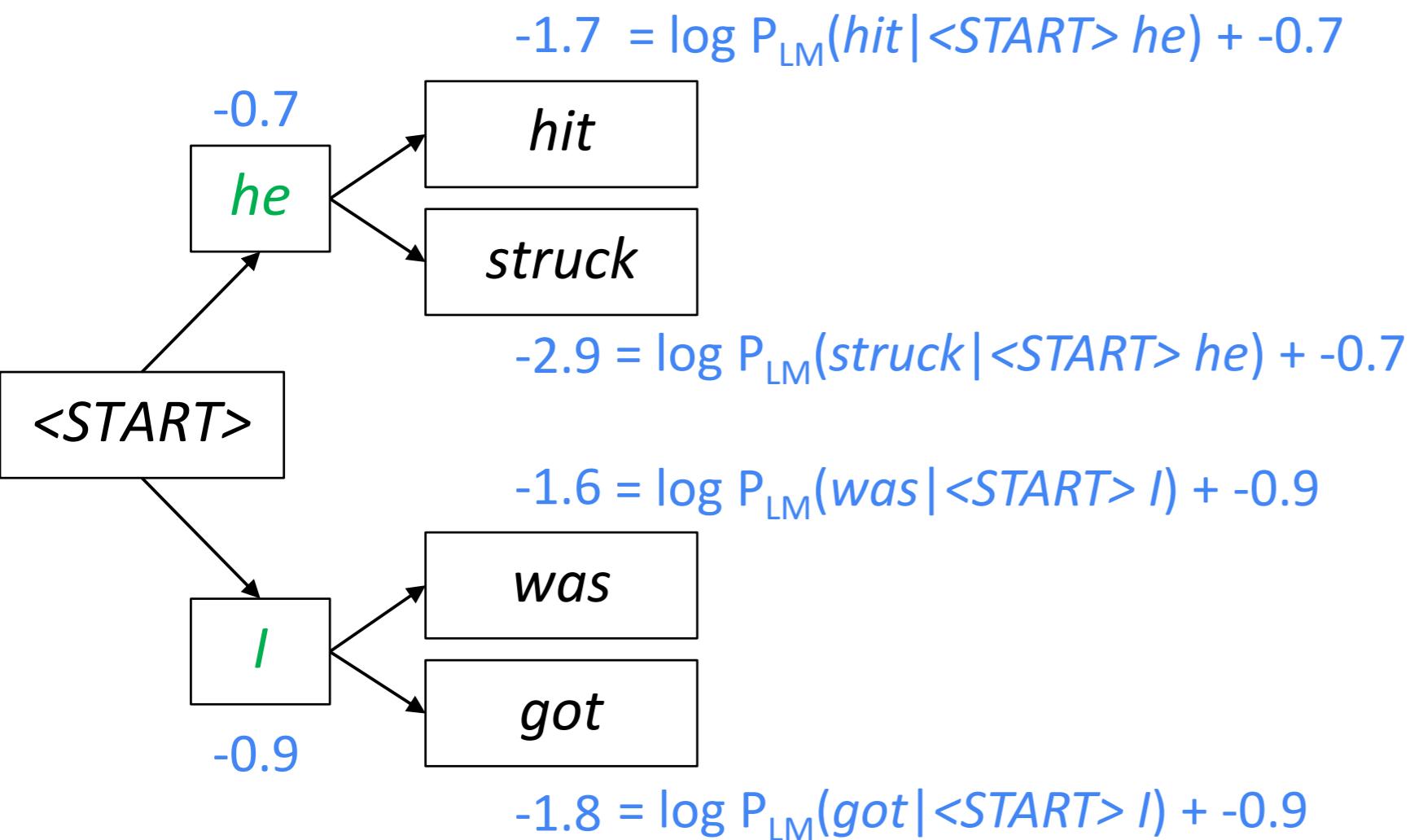


$$-0.9 = \log P_{\text{LM}}(I | <\text{START}>)$$

Take top k words
and compute scores

Beam search decoding: example

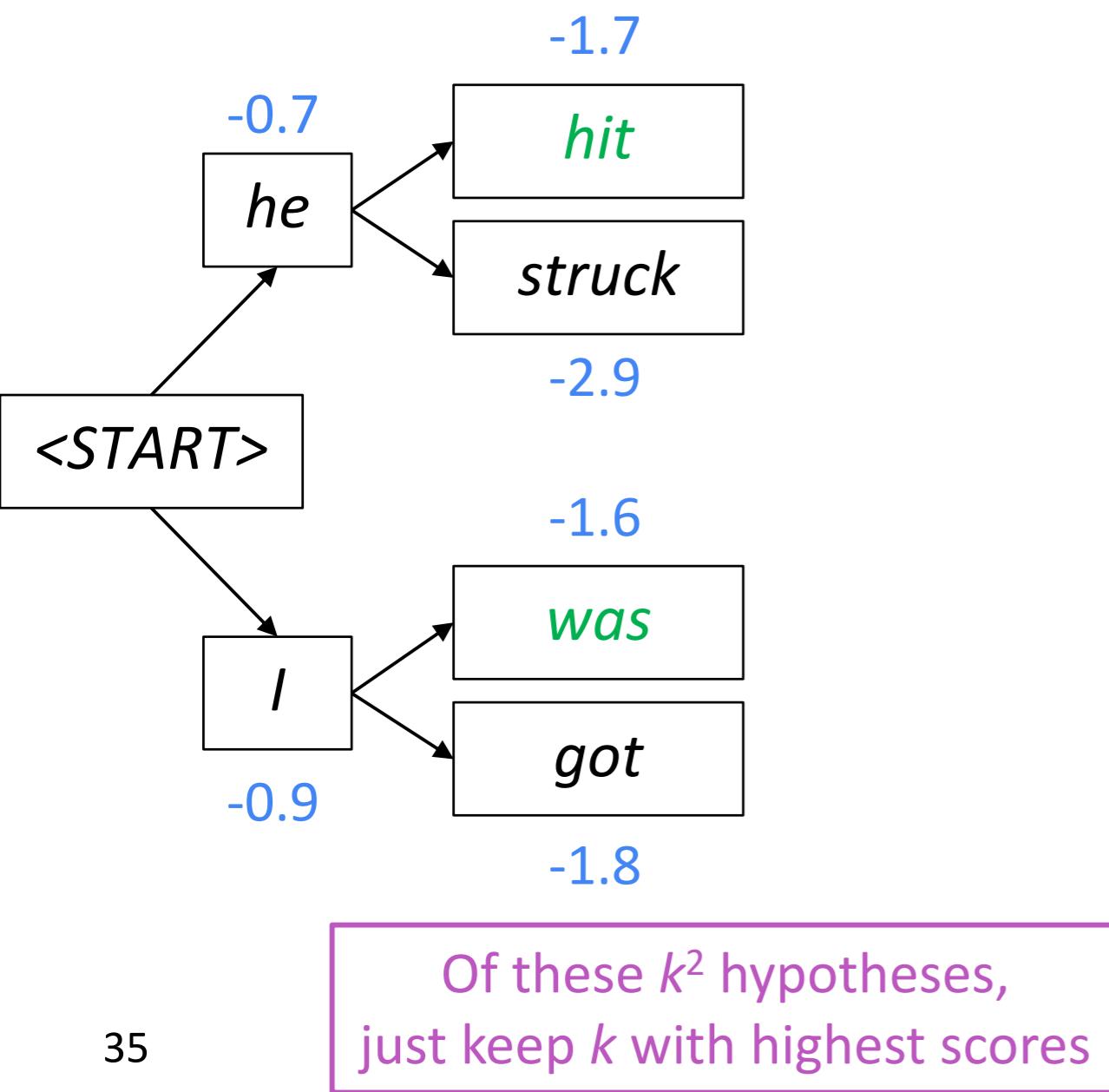
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find
top k next words and calculate scores

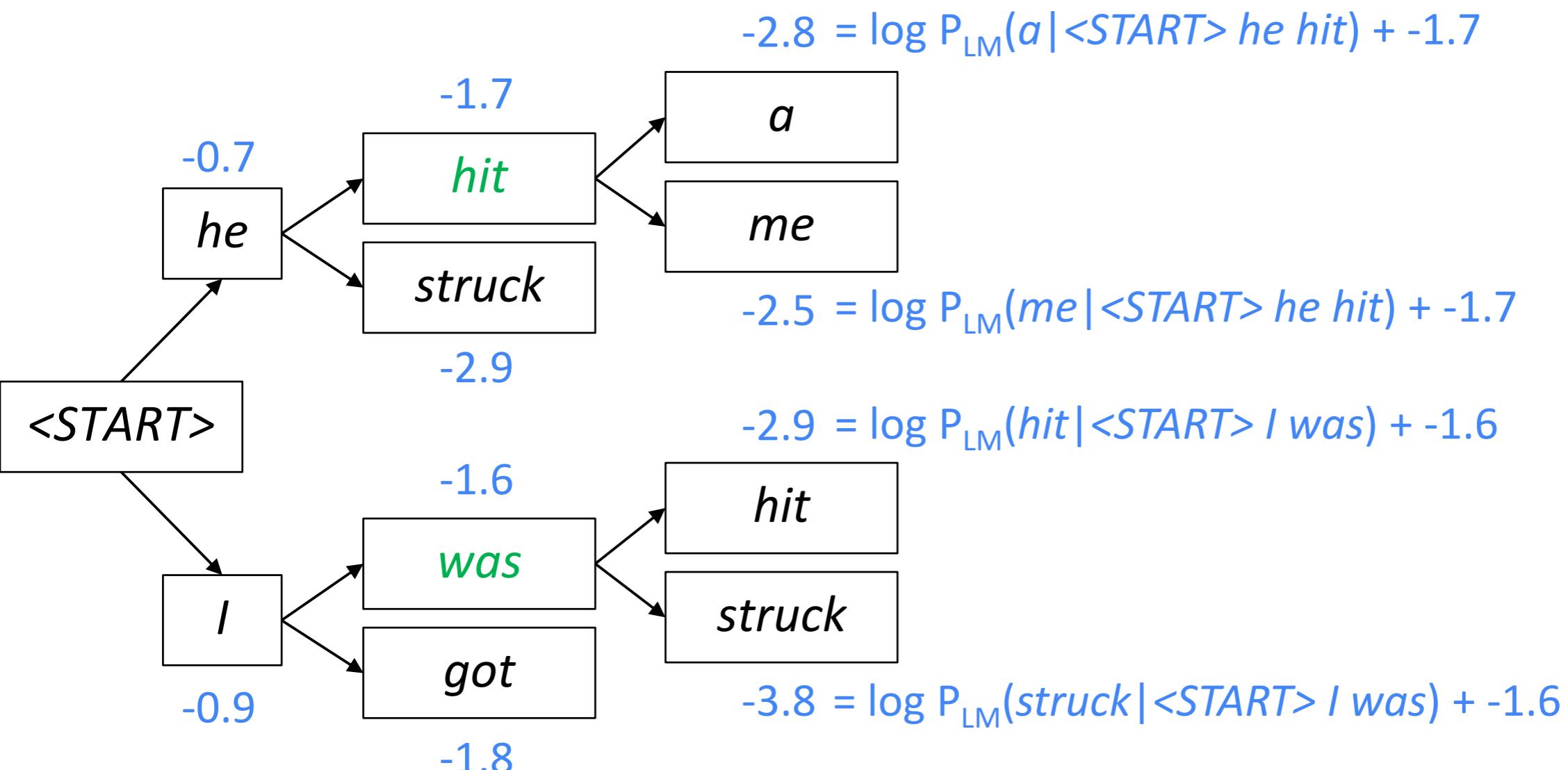
Beam search decoding: example

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



Beam search decoding: example

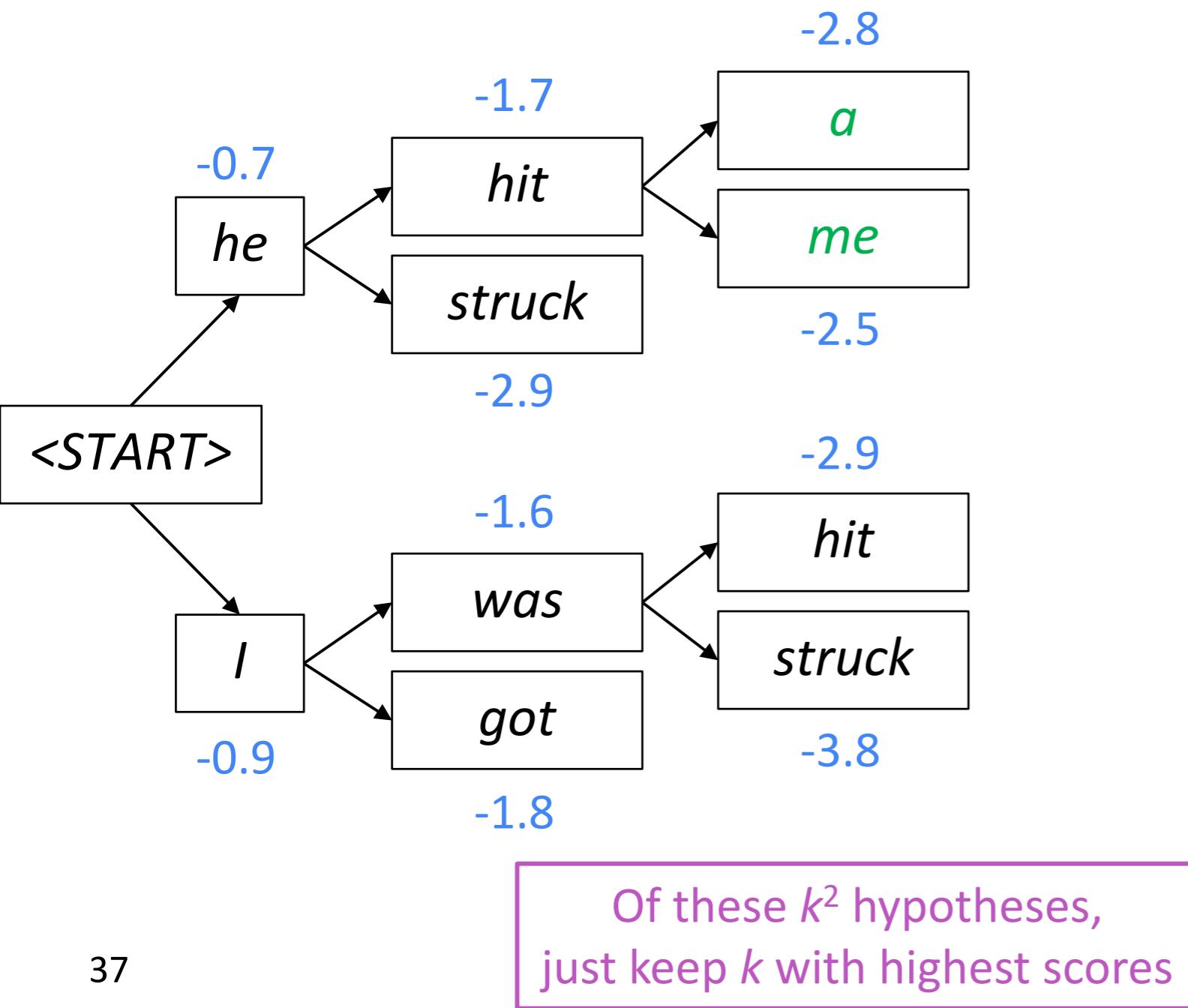
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find top k next words and calculate scores

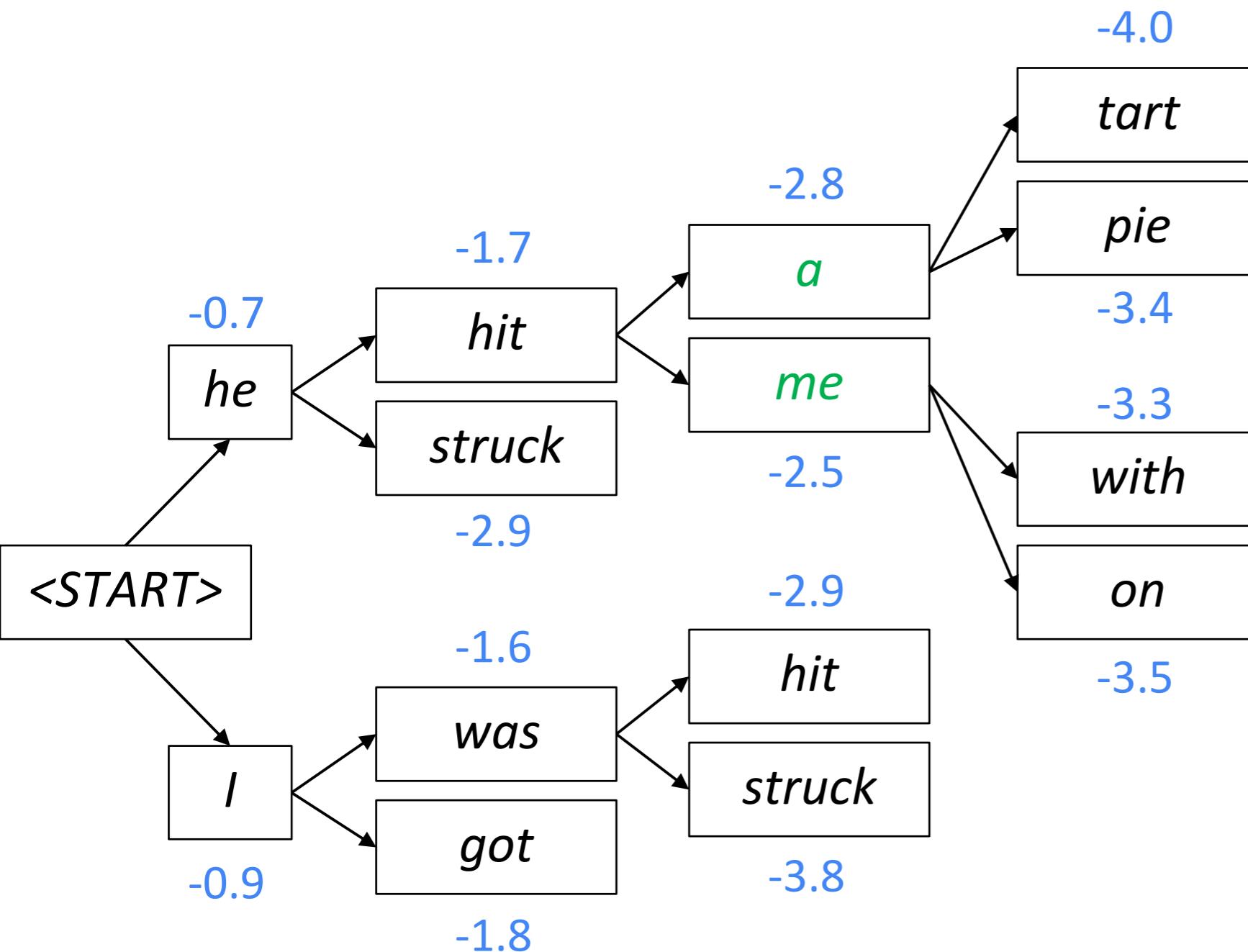
Beam search decoding: example

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



Beam search decoding: example

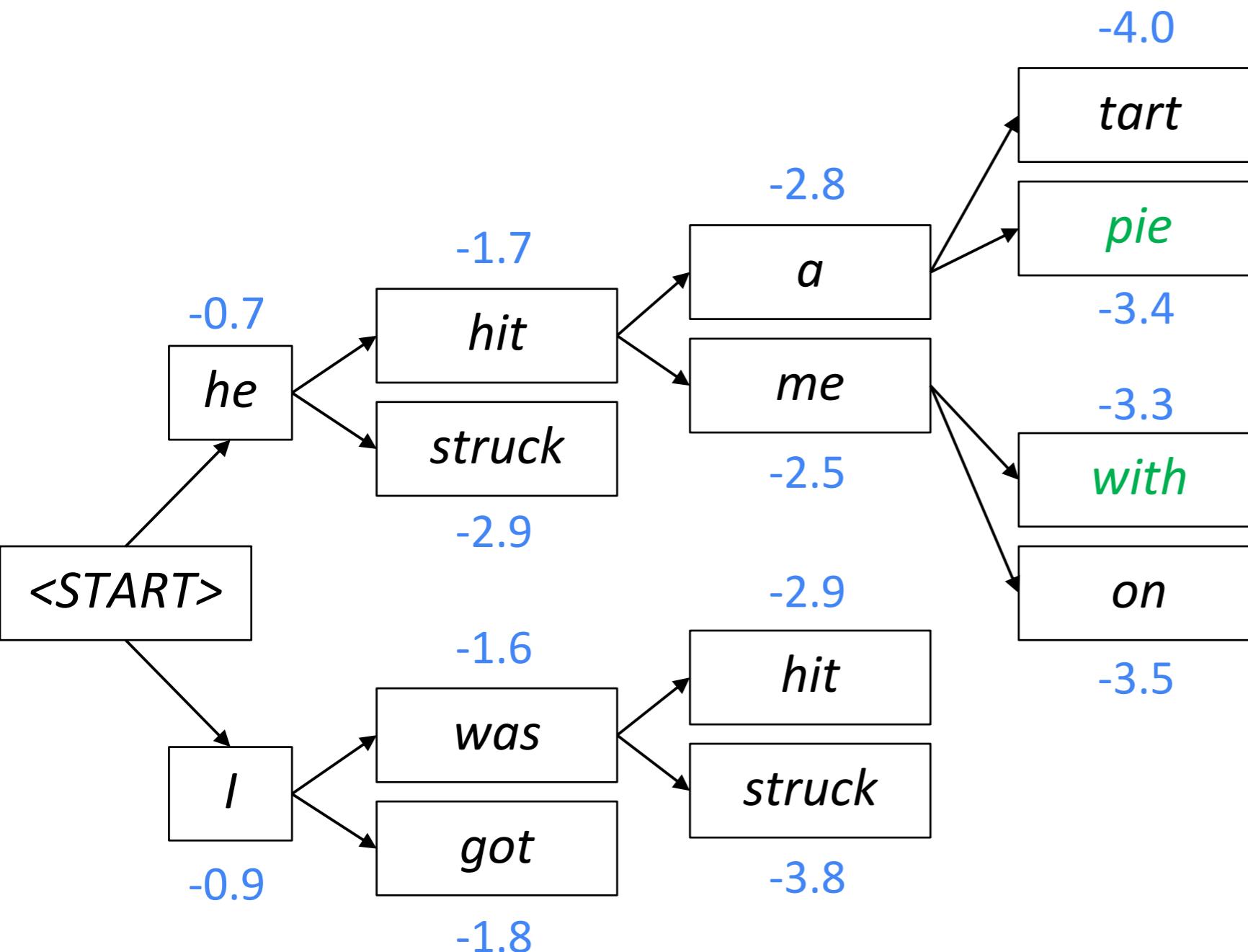
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding: example

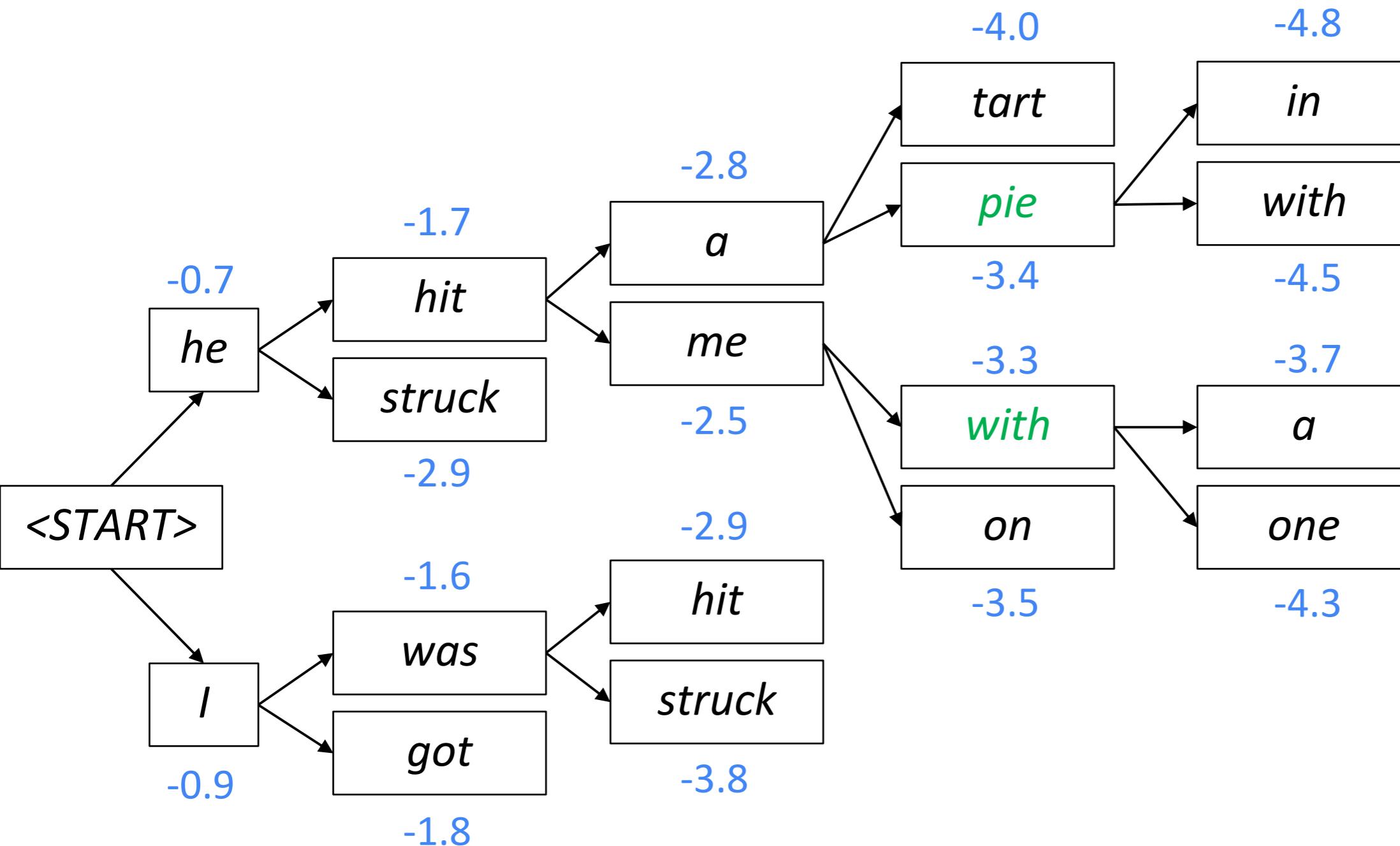
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Of these k^2 hypotheses,
just keep k with highest scores

Beam search decoding: example

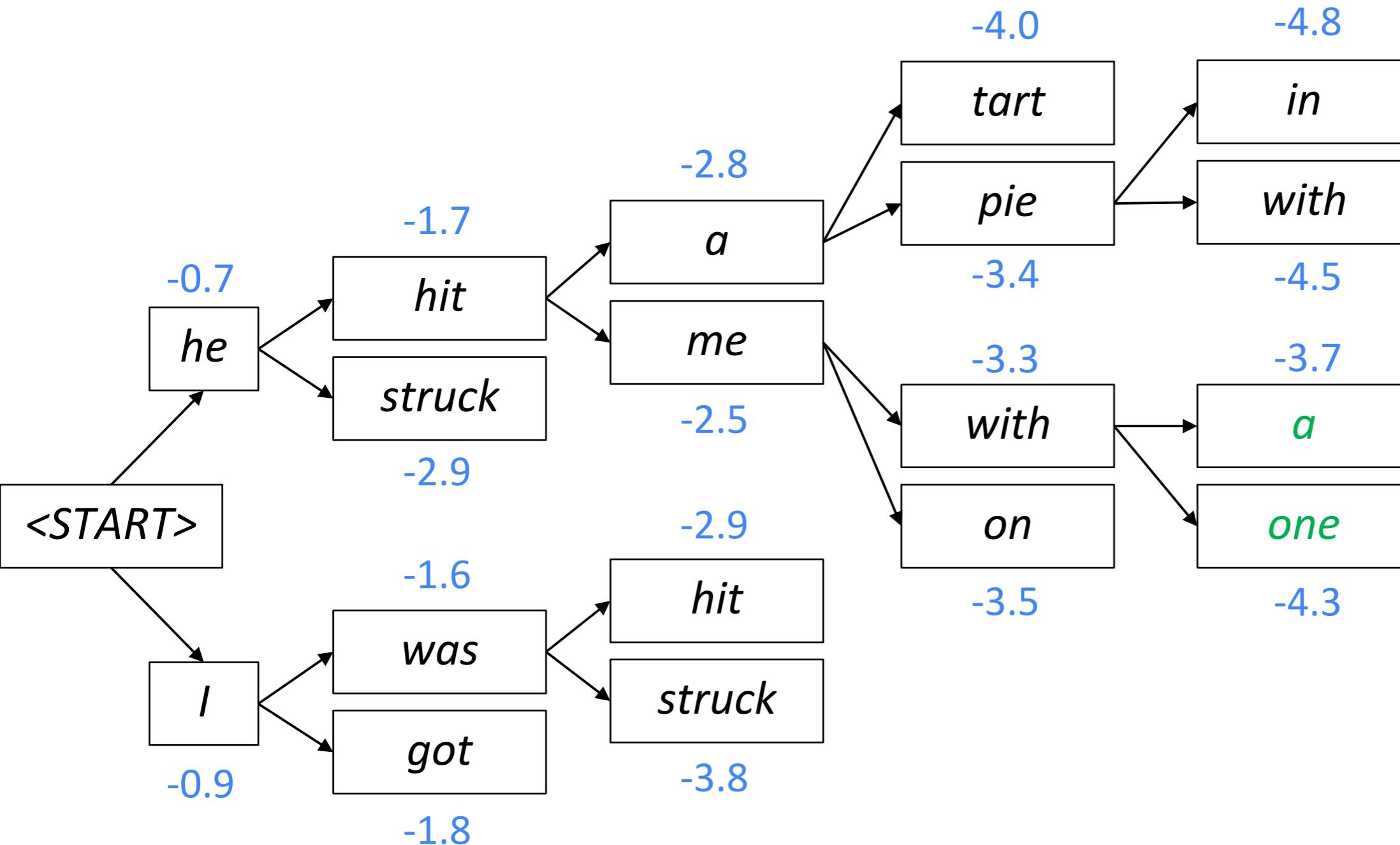
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding: example

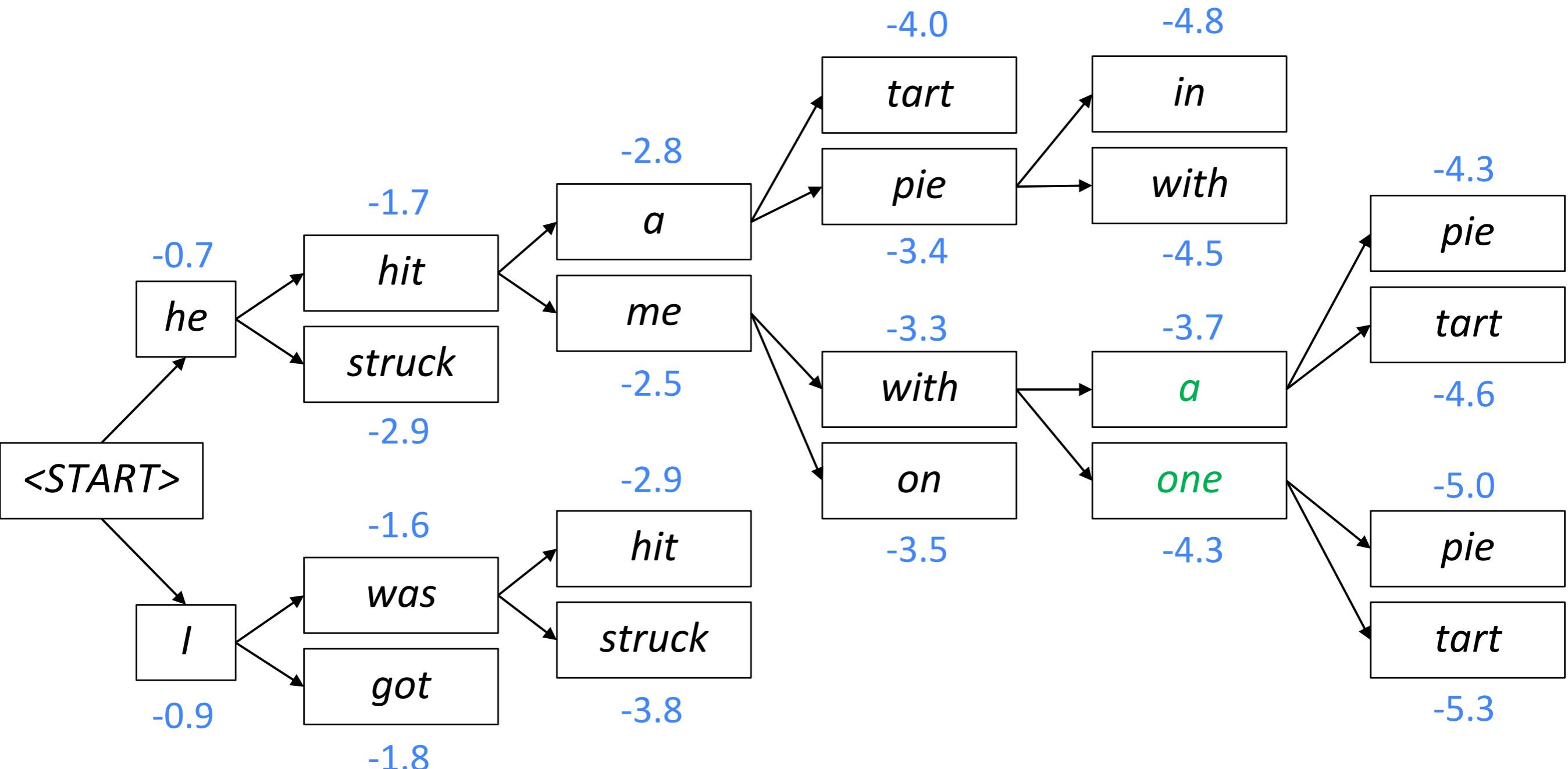
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Of these k^2 hypotheses,
just keep k with highest scores

Beam search decoding: example

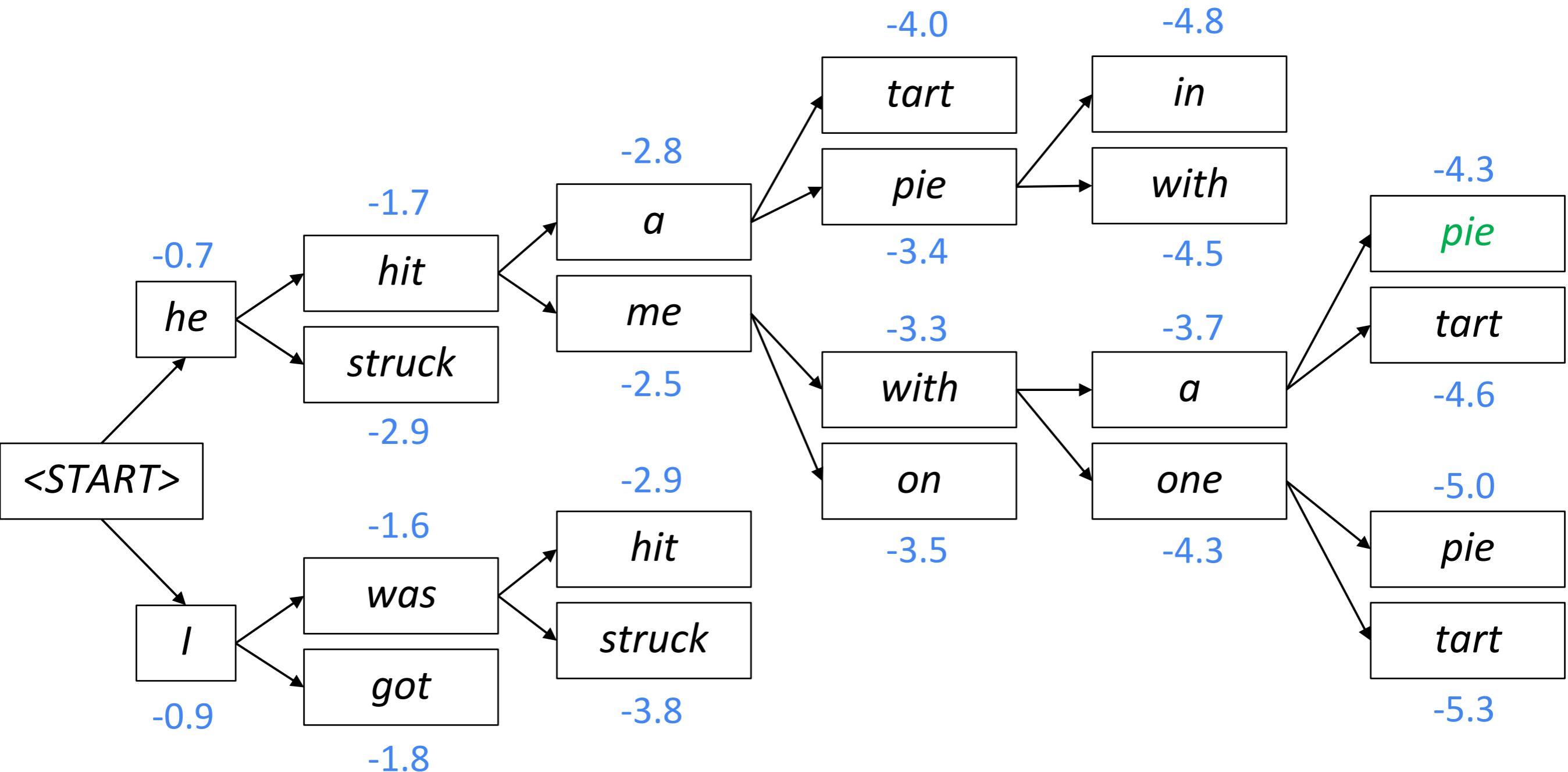
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find top k next words and calculate scores

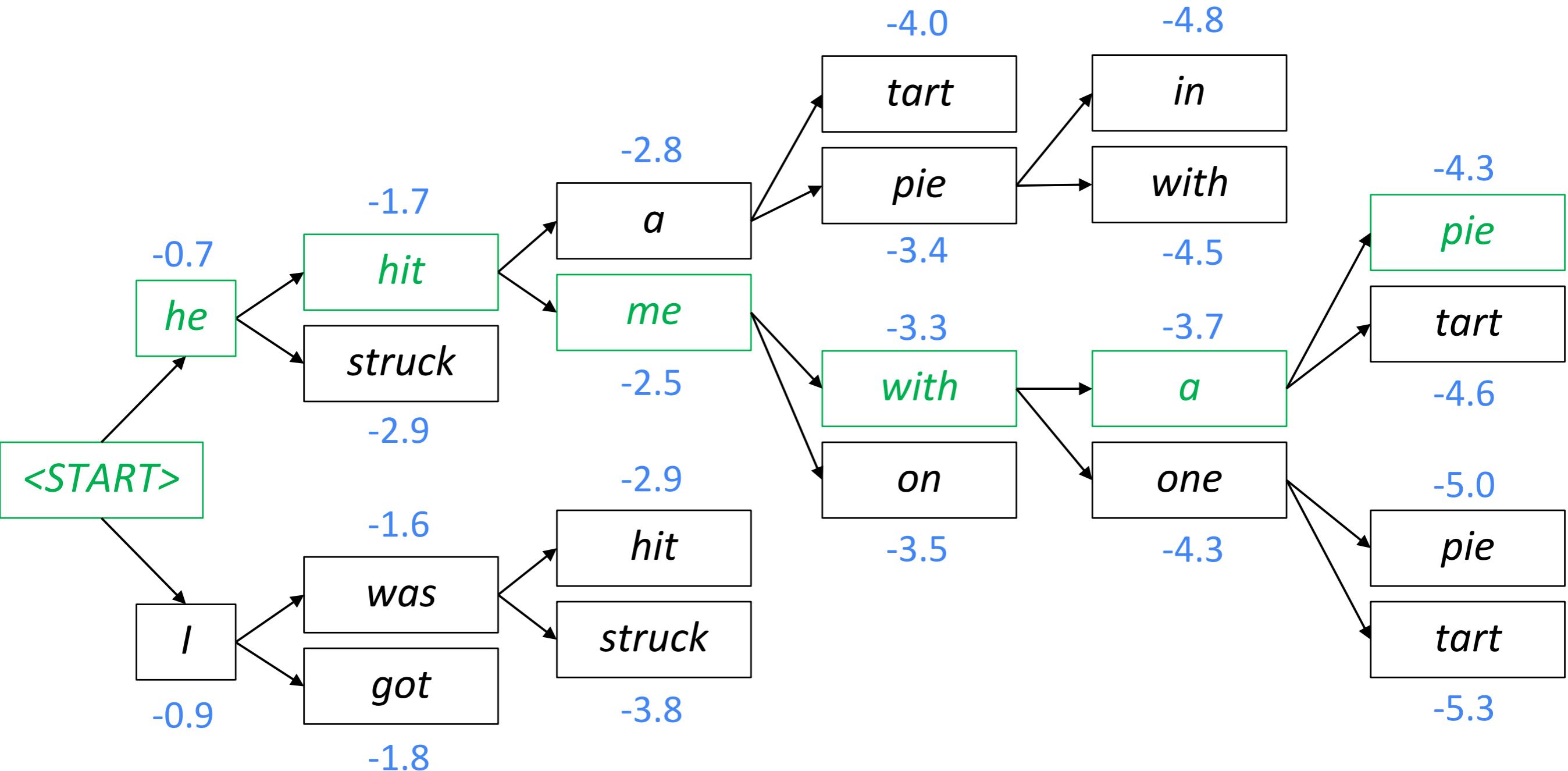
Beam search decoding: example

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



Beam search decoding: example

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



Backtrack to obtain the full hypothesis

Beam search decoding: stopping criterion

- In **greedy decoding**, usually we decode until the model produces a **<END> token**
 - For example: *<START> he hit me with a pie <END>*
- 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 timestep T (where T is some pre-defined cutoff), or
 - We have at least n completed hypotheses (where n is pre-defined cutoff)

Beam search decoding: finishing up

- We have our list of completed hypotheses.
- How to select top one with highest score?
- Each hypothesis y_1, \dots, y_t on our list has a score

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

- Problem with this: longer hypotheses have lower scores
- Fix: Normalize by length. Use this to select top one instead:

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

Advantages of NMT

Compared to SMT, NMT has many **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

Disadvantages of NMT?

Compared to SMT:

- 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!

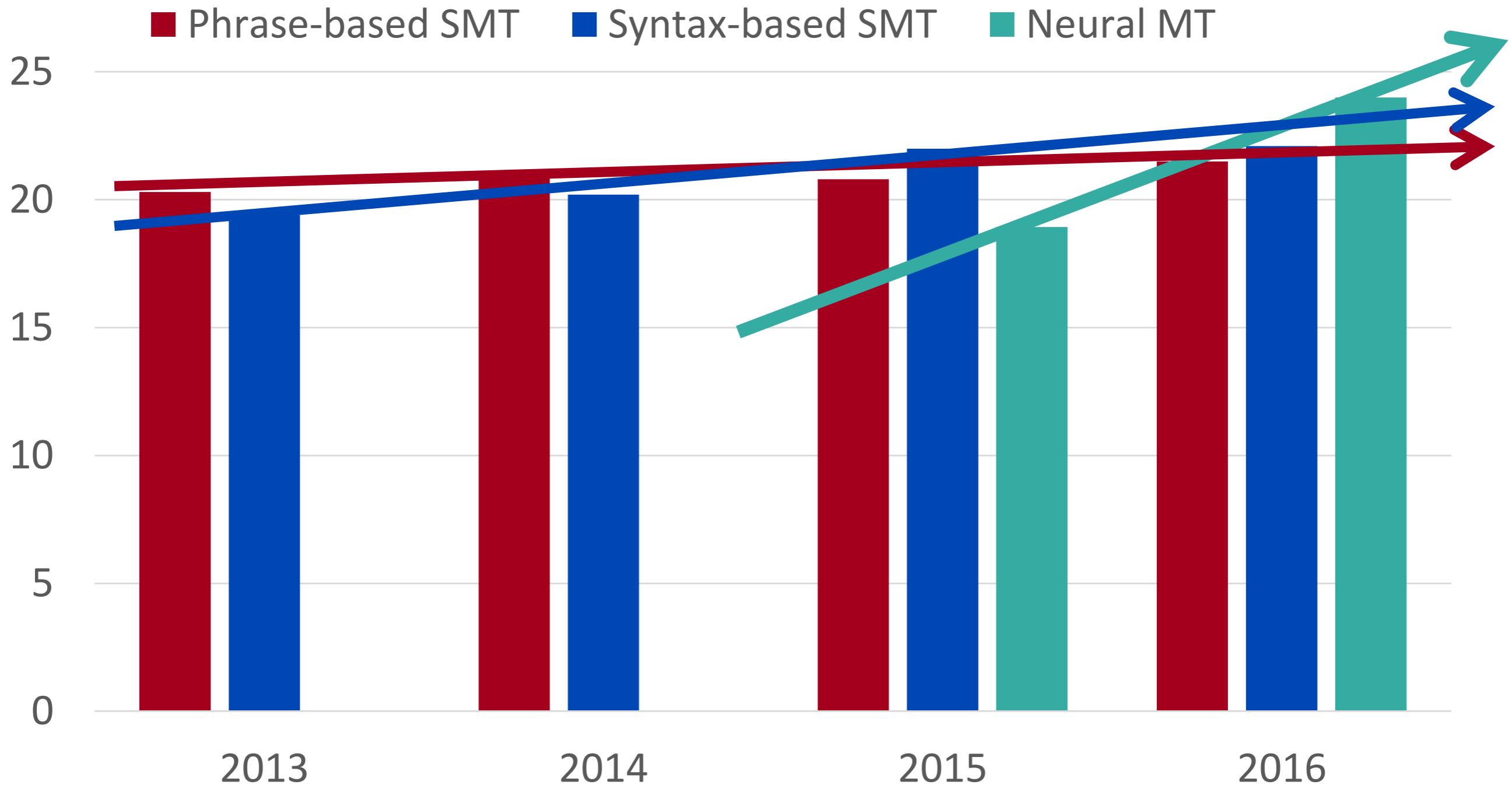
How do we evaluate Machine Translation?

BLEU (Bilingual Evaluation Understudy)

- BLEU compares the machine-written translation to one or several human-written translation(s), and computes a **similarity score** based on:
 - ***n*-gram precision** (usually for 1, 2, 3 and 4-grams)
 - Plus a penalty for too-short system translations
- BLEU is **useful** but **imperfect**
 - There are many valid ways to translate a sentence
 - So a **good** translation can get a **poor** BLEU score because it has low *n*-gram overlap with the human translation ☹

MT progress over time

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal]



NMT: the biggest success story of NLP Deep Learning

Neural Machine Translation went from a **fringe research activity** in **2014** to the **leading standard method** in **2016**

- **2014:** First seq2seq paper published
- **2016:** Google Translate switches from SMT to NMT
- This is amazing!
 - **SMT** systems, built by **hundreds** of engineers over many **years**, outperformed by NMT systems trained by a **handful** of engineers in a few **months**

So is Machine Translation solved?

- **Nope!**
- Many difficulties remain:
 - Out-of-vocabulary words
 - Domain mismatch between train and test data
 - Maintaining context over longer text
 - Low-resource language pairs

Further reading: “Has AI surpassed humans at translation? Not even close!”

https://www.skynettoday.com/editorials/state_of_nmt

So is Machine Translation solved?

- **Nope!**
- Using common sense is still hard

The image shows a screenshot of the Google Translate interface. On the left, under 'English', the text 'paper jam' is displayed with an 'Edit' link below it. On the right, under 'Spanish', the text 'Mermelada de papel' is shown. Both sections include language selection dropdowns ('English▼' and 'Spanish▼'), microphone and speaker icons, and a refresh arrow icon. At the bottom of the interface, there are 'Open in Google Translate' and 'Feedback' links.



So is Machine Translation solved?

- **Nope!**
- NMT picks up **biases** in training data

Malay - detected ▾

English ▾

Dia bekerja sebagai jururawat.

Dia bekerja sebagai pengaturcara. Edit

She works as a nurse.

He works as a programmer.

Didn't specify gender

NMT research continues

NMT is the **flagship task** for NLP Deep Learning

- NMT research has **pioneered** many of the recent **innovations** of NLP Deep Learning
- In **2019**: NMT research continues to **thrive**
 - Researchers have found ***many, many improvements*** to the “vanilla” seq2seq NMT system we’ve presented today
 - But **one improvement** is so integral that it is the new vanilla...

ATTENTION

Summary of today's lecture

- We learned some history of Machine Translation (MT)
- Since 2014, Neural MT rapidly replaced intricate Statistical MT
- Sequence-to-sequence is the architecture for NMT (uses 2 RNNs)
- Attention is a way to *focus on particular parts* of the input
 - Improves sequence-to-sequence a lot!



slide credits

slides that look like this

come from

1990s-2010s: Statistical Machine Translation

- Core idea: Learn a **probabilistic model** from **data**
- Suppose we're translating French → English.
- We want to find **best English sentence y , given French sentence x**
$$\operatorname{argmax}_y P(y|x)$$
- Use Bayes Rule to break this down into **two components** to be learnt separately:

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



Chris Manning &
Abigail See's Stanford
cs224n course, 2019
edition

7

and their use is gratefully acknowledged. I try to make any modifications obvious, but if there are errors on a slide, assume that I added them.