

BBM 495
MACHINE TRANSLATION
ENCODER-DECODER MODELS
LECTURER: BURCU CAN



2019-2020 SPRING

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 \rightarrow 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:

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

Translation Model

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

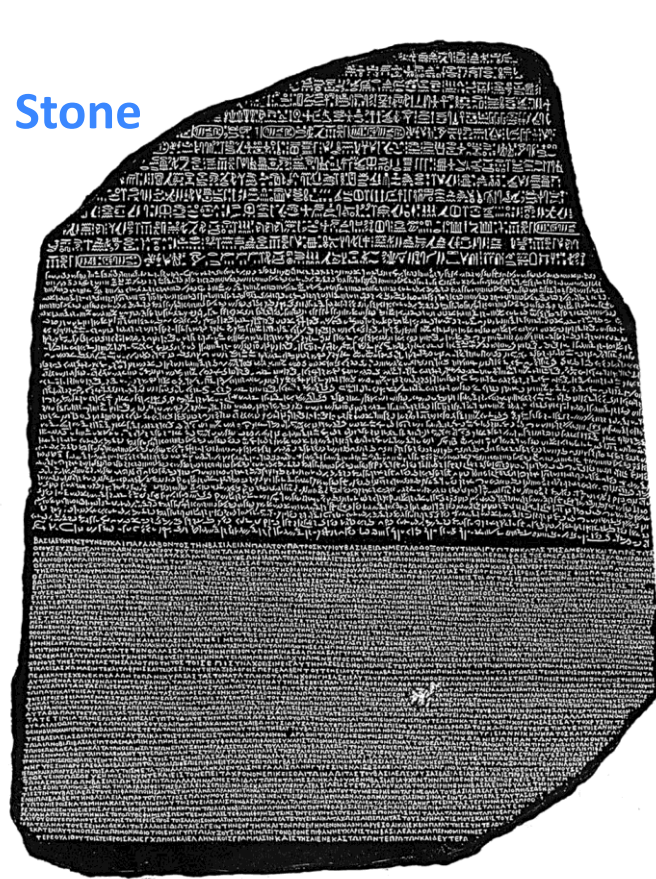
Language Model

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

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

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

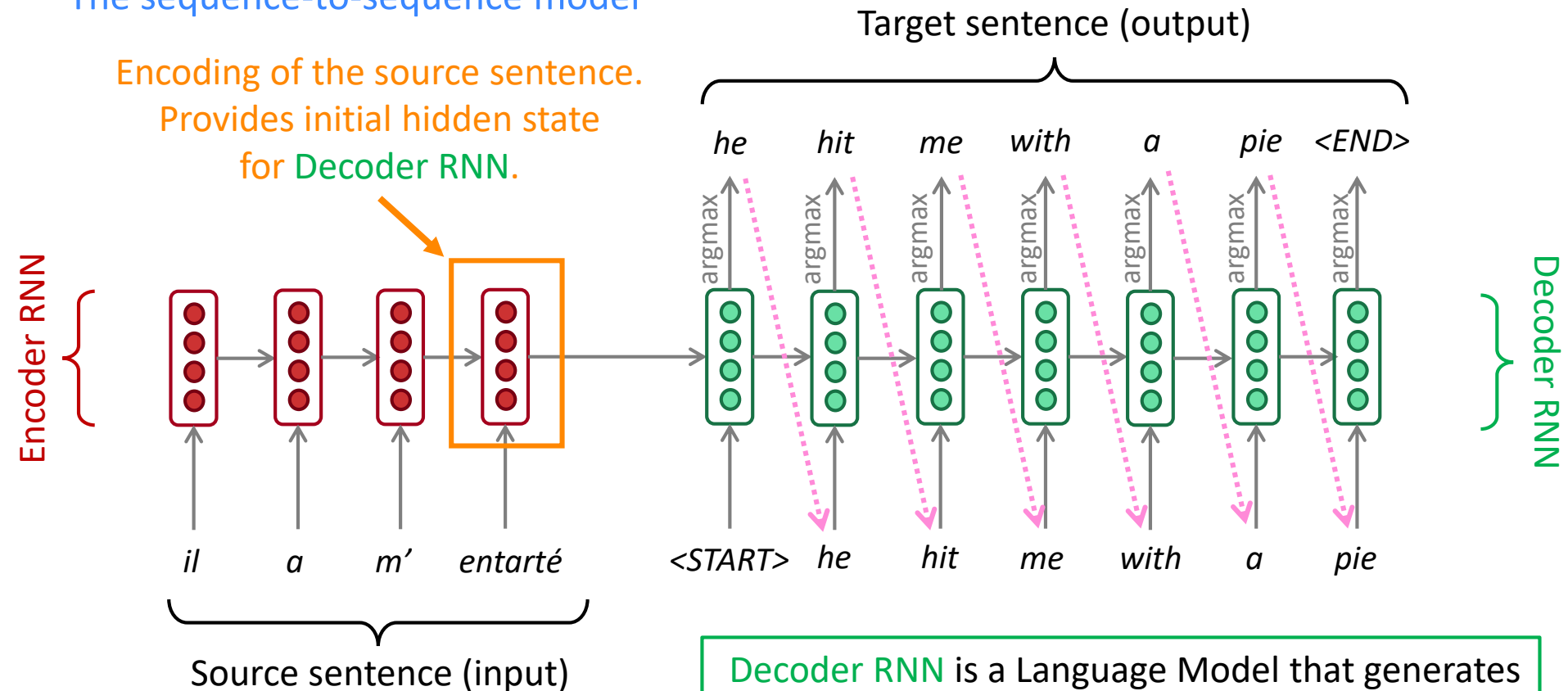
What is Neural Machine Translation?

- 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

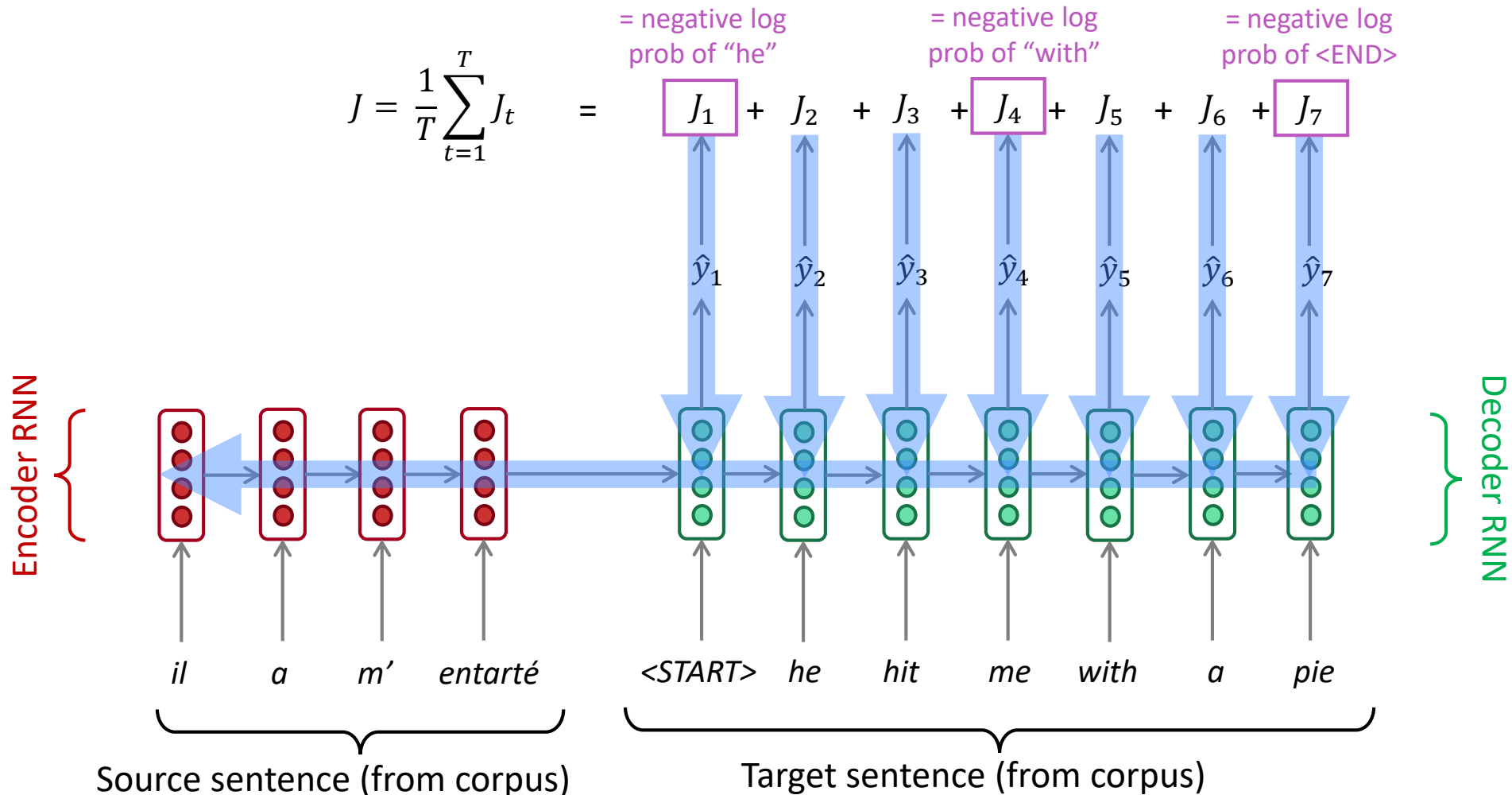
- NMT directly calculates $P(y|x)$:

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

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...

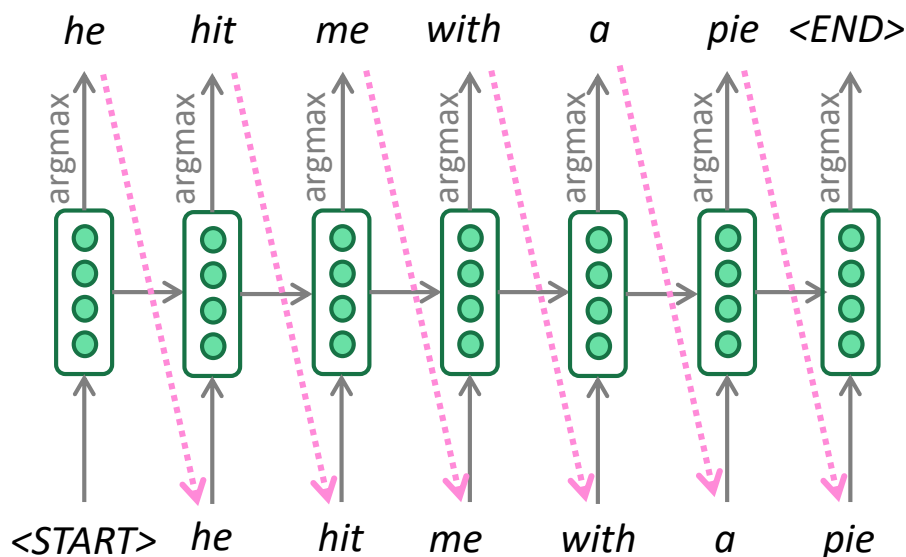
Training a Neural Machine Translation system



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

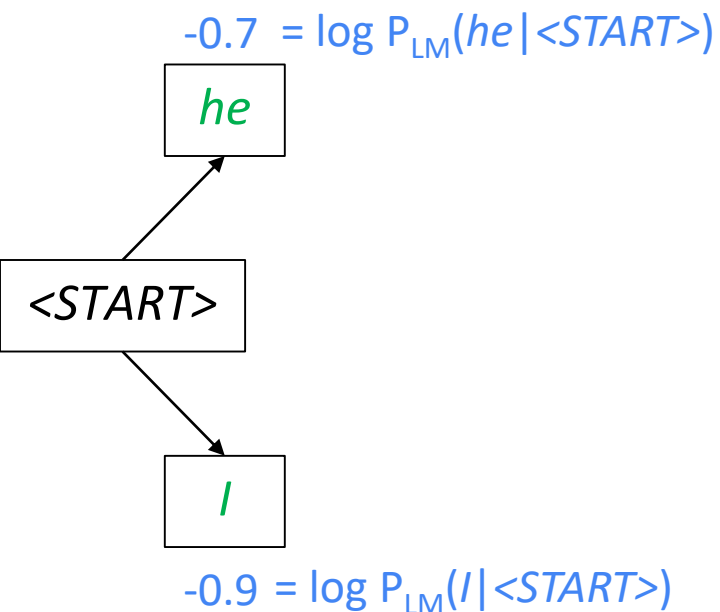
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$

<START>

Calculate prob
dist of next word

Beam search decoding: example

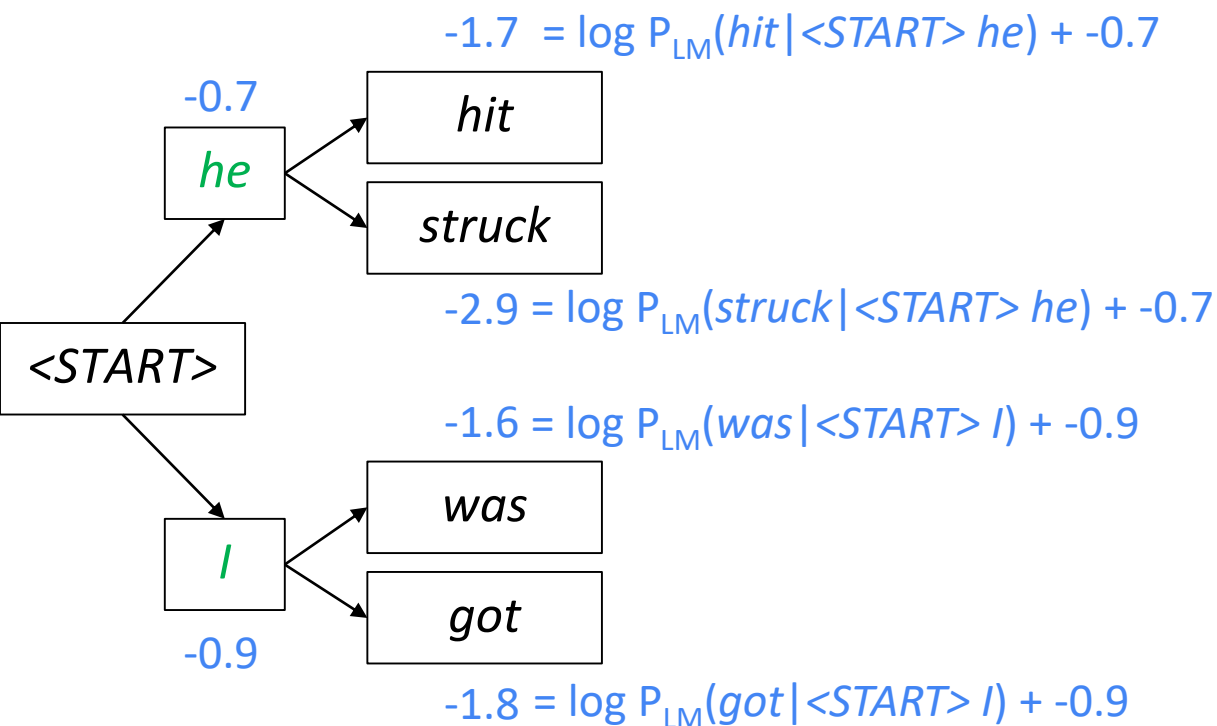
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Take top k words
and compute scores

Beam search decoding: example

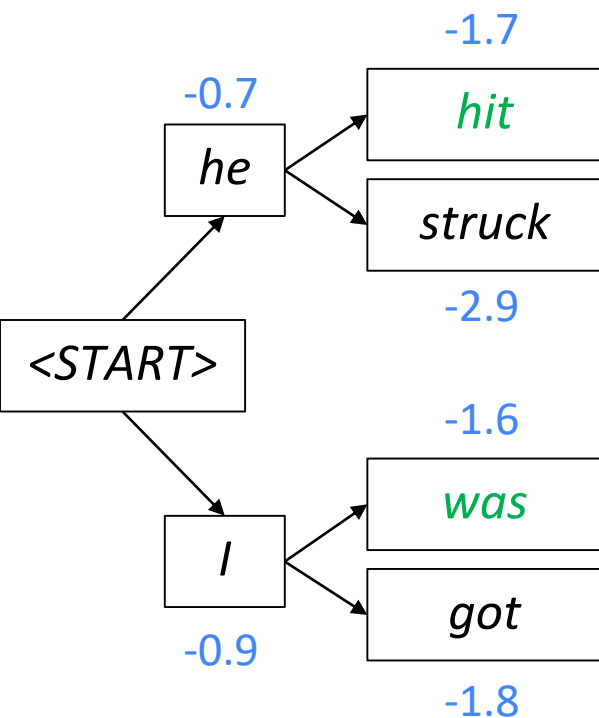
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



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

Beam search decoding: example

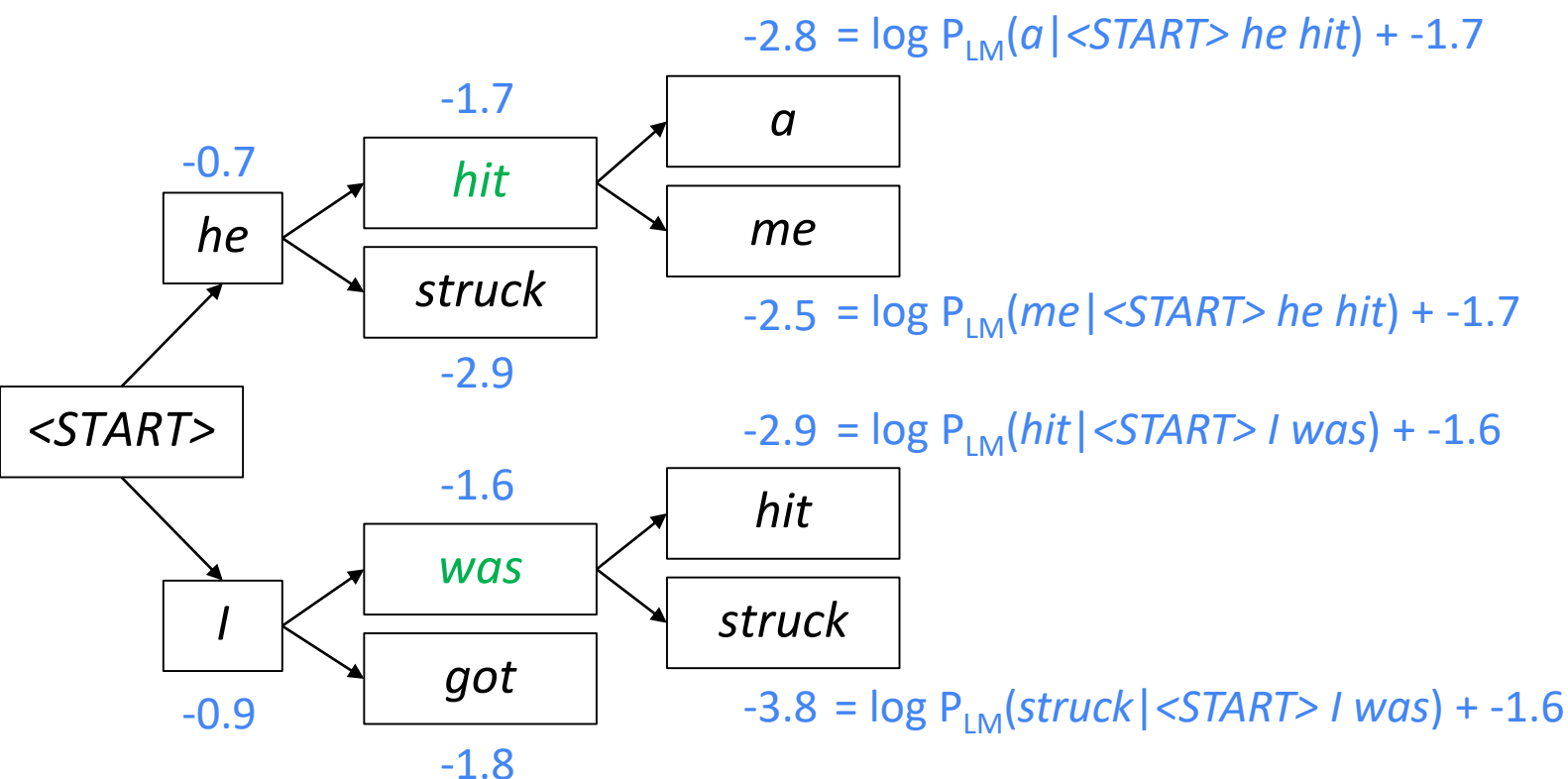
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



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

Beam search decoding: example

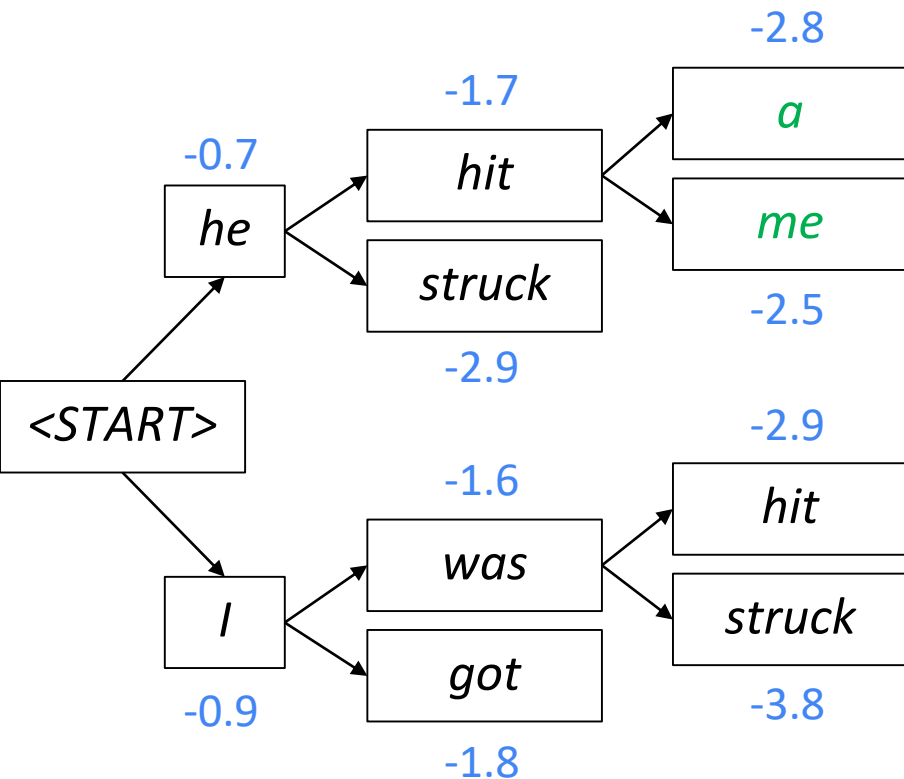
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



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

Beam search decoding: example

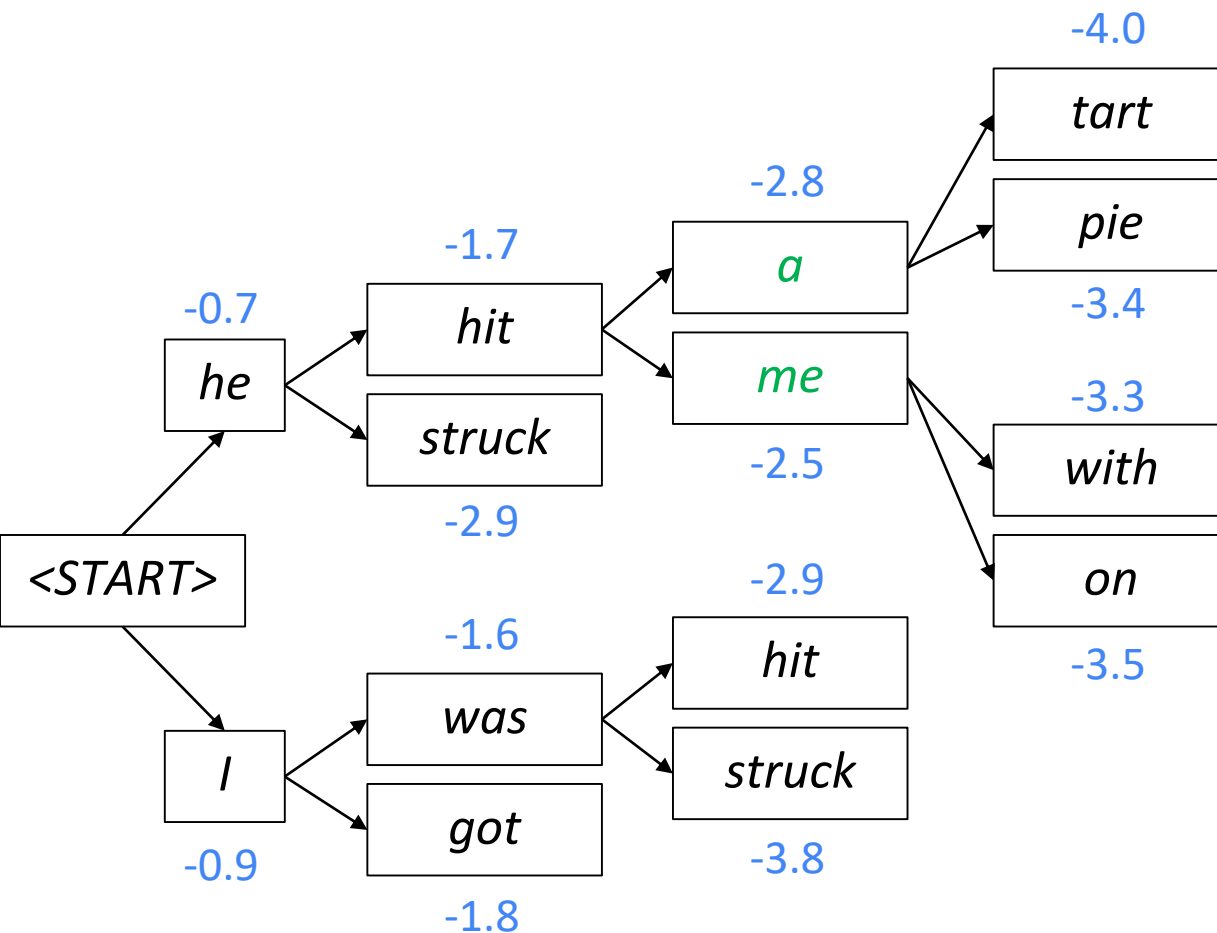
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



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

Beam search decoding: example

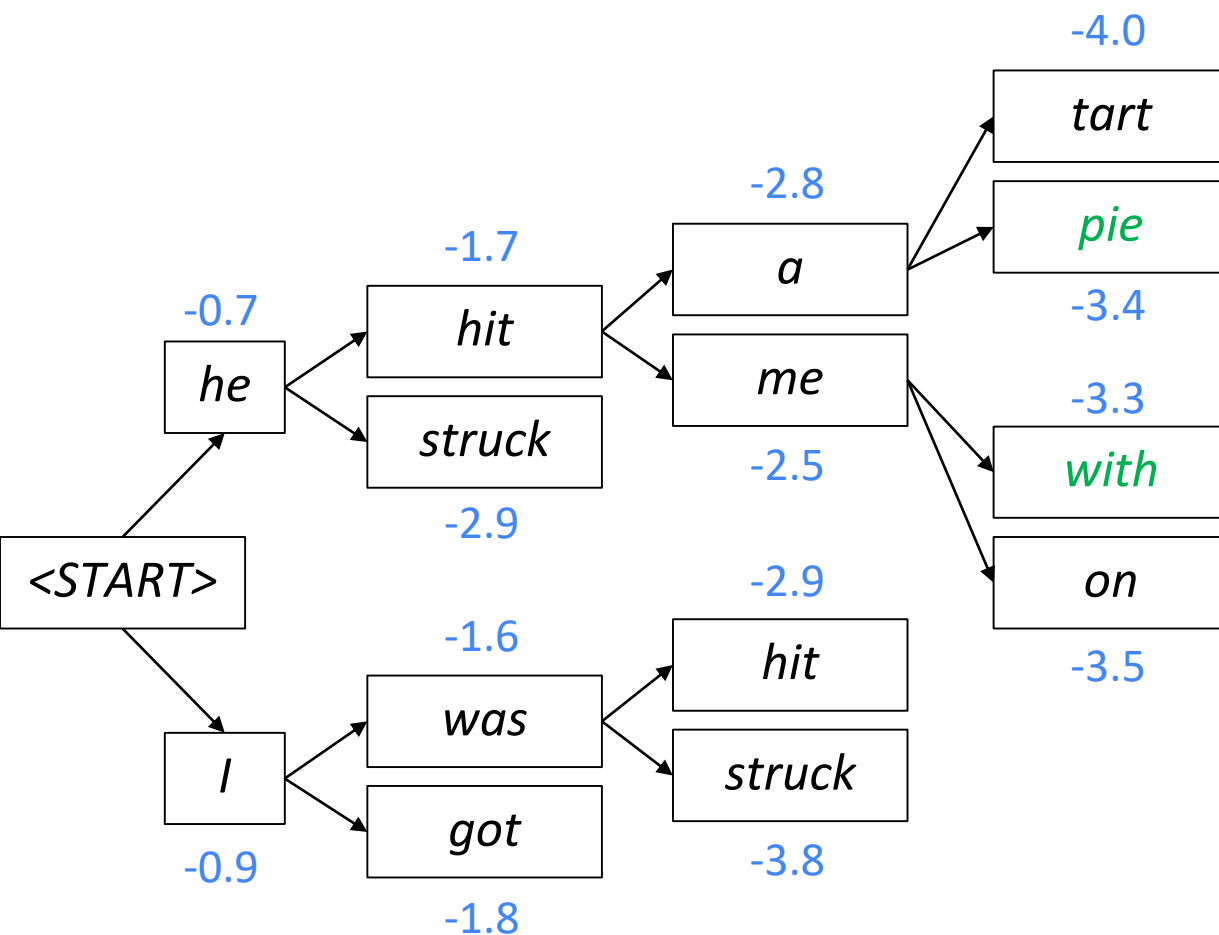
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



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

Beam search decoding: example

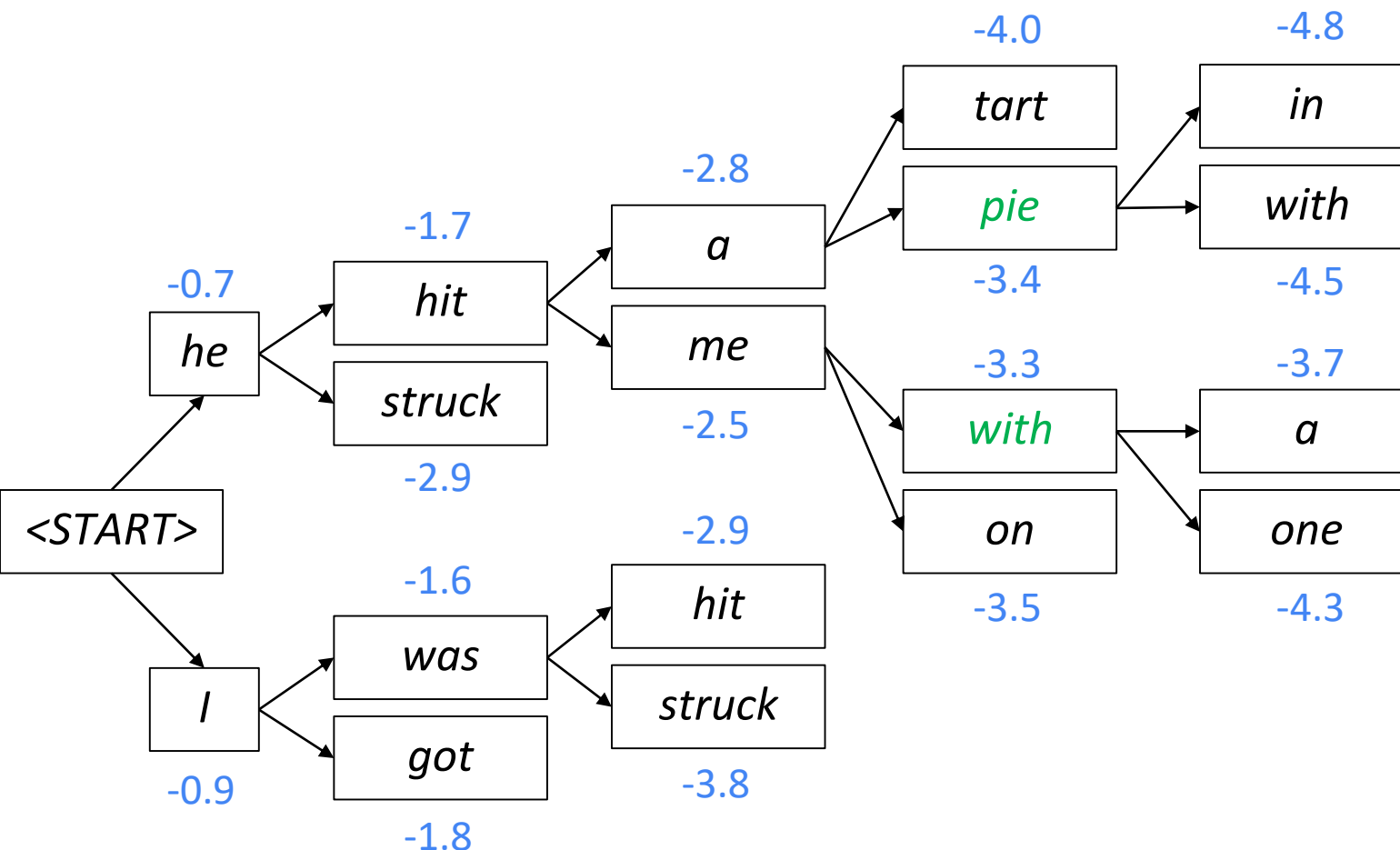
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



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

Beam search decoding: example

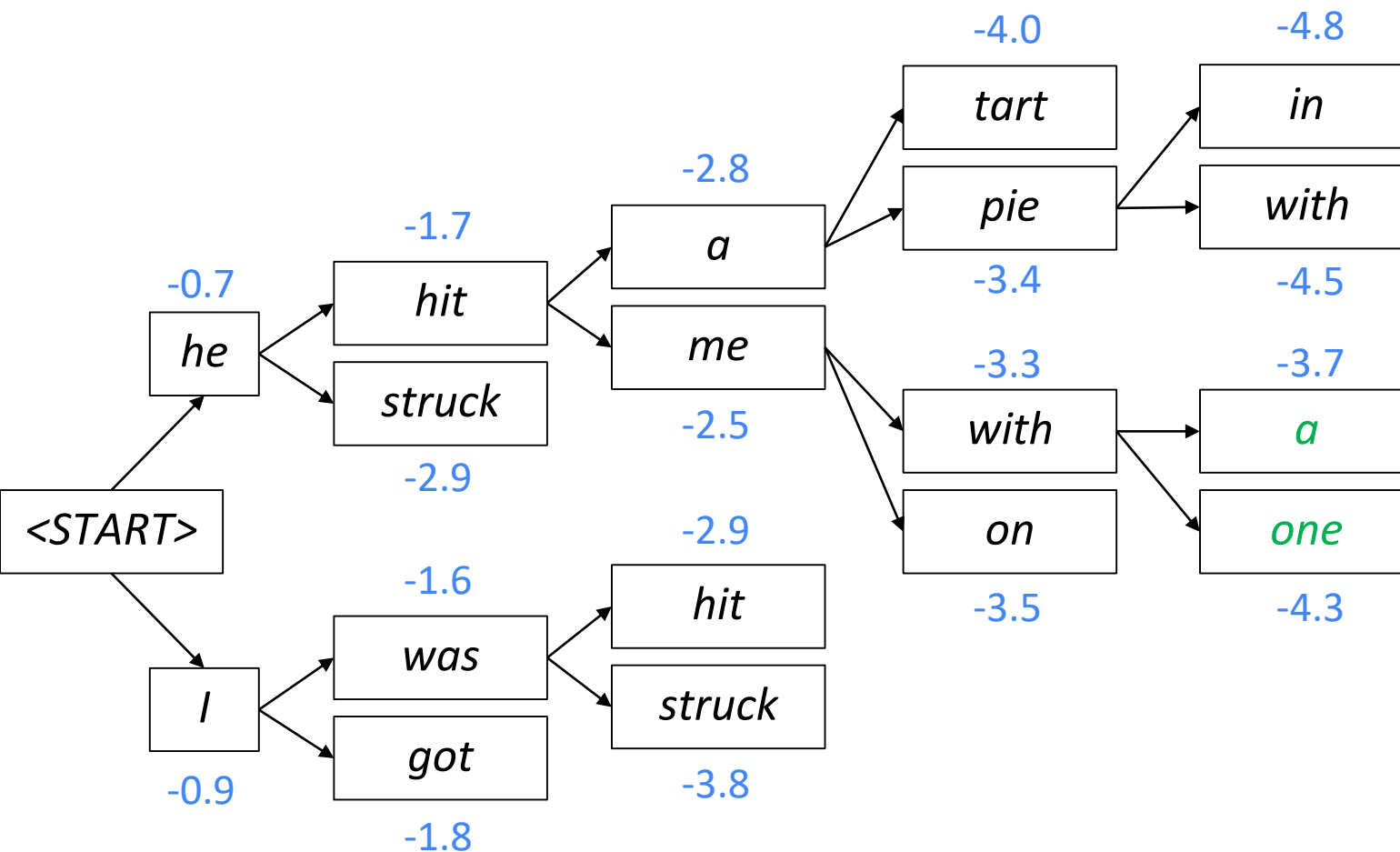
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



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

Beam search decoding: example

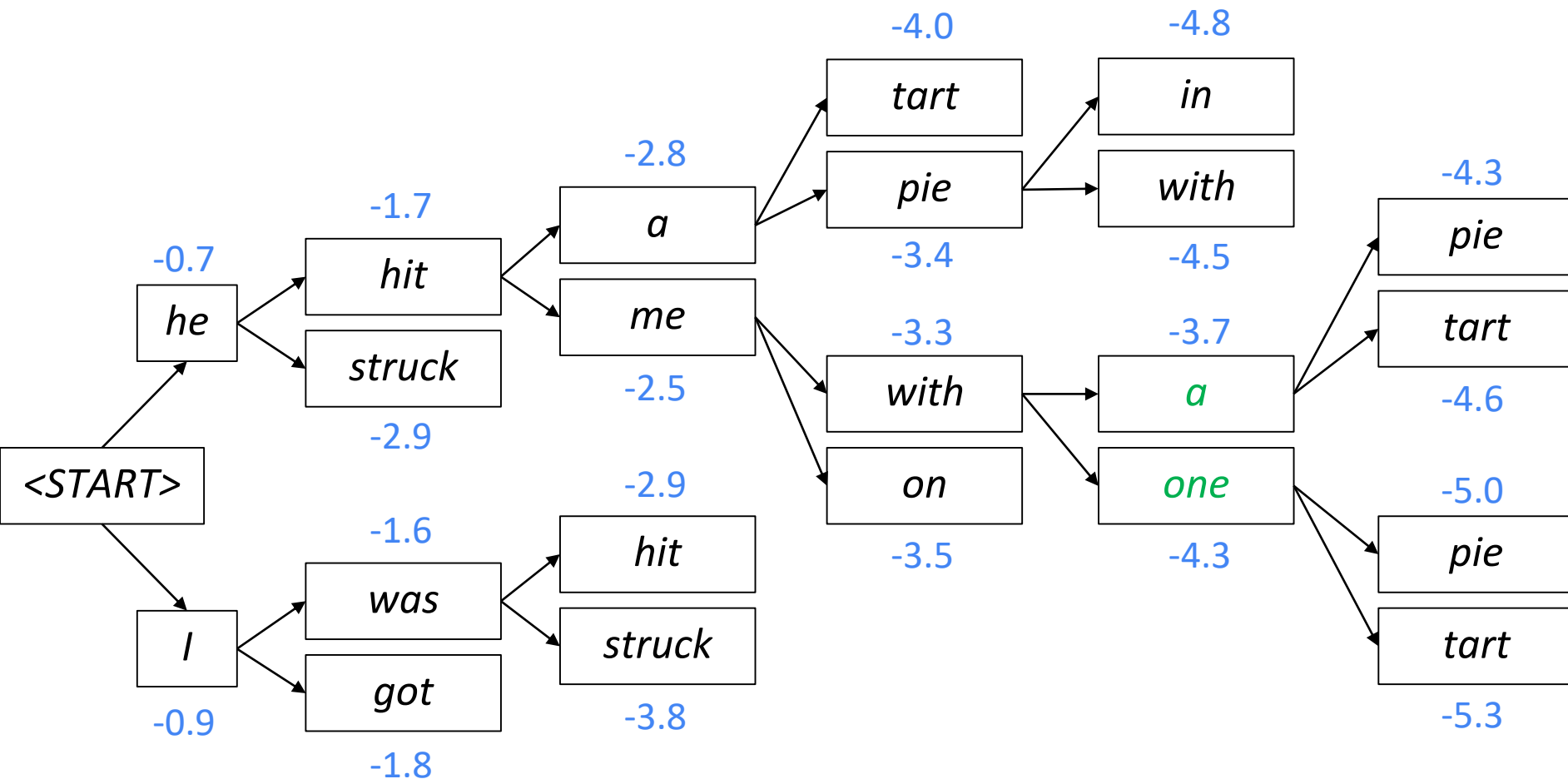
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



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

Beam search decoding: example

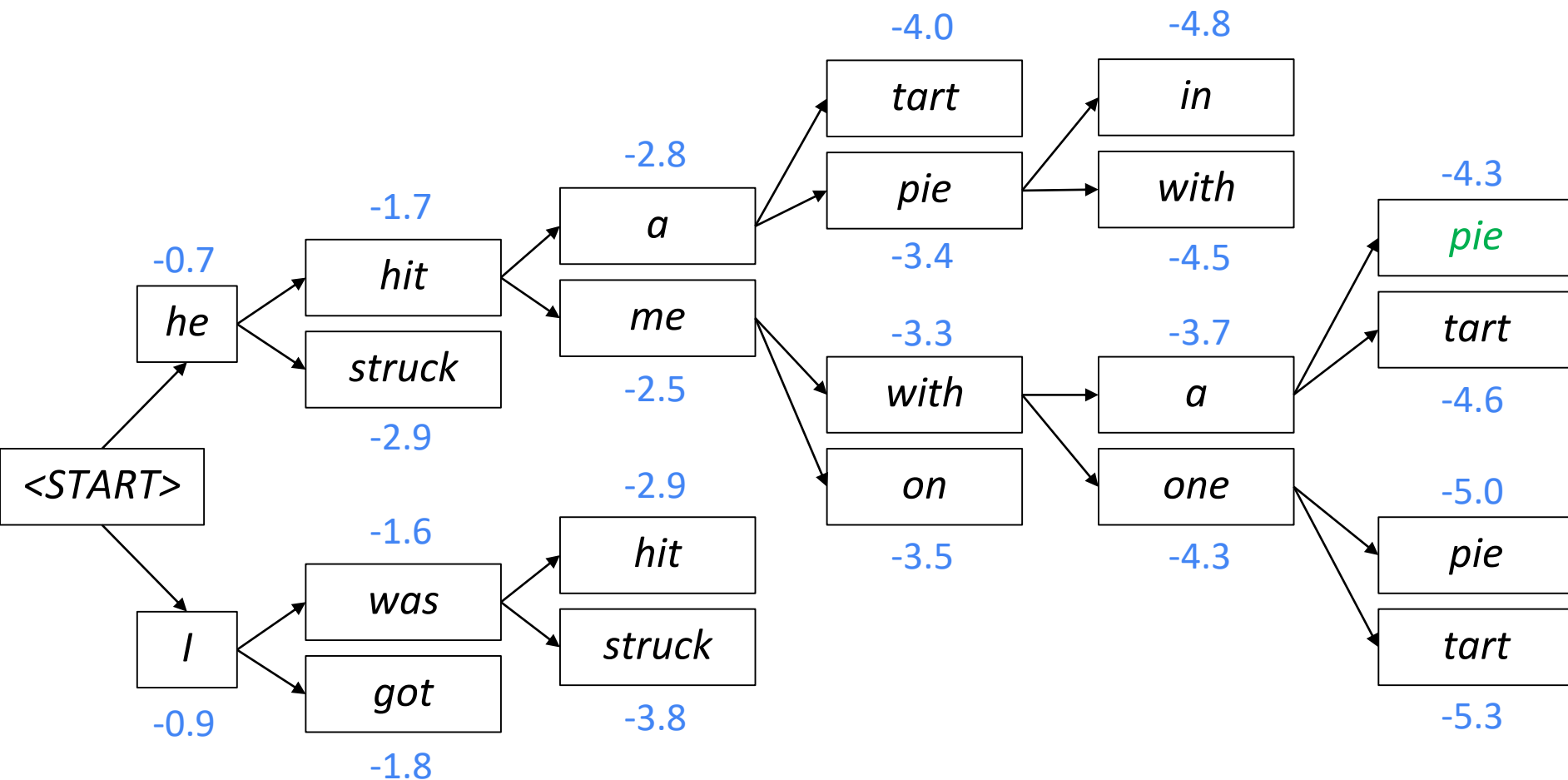
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



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

Beam search decoding: example

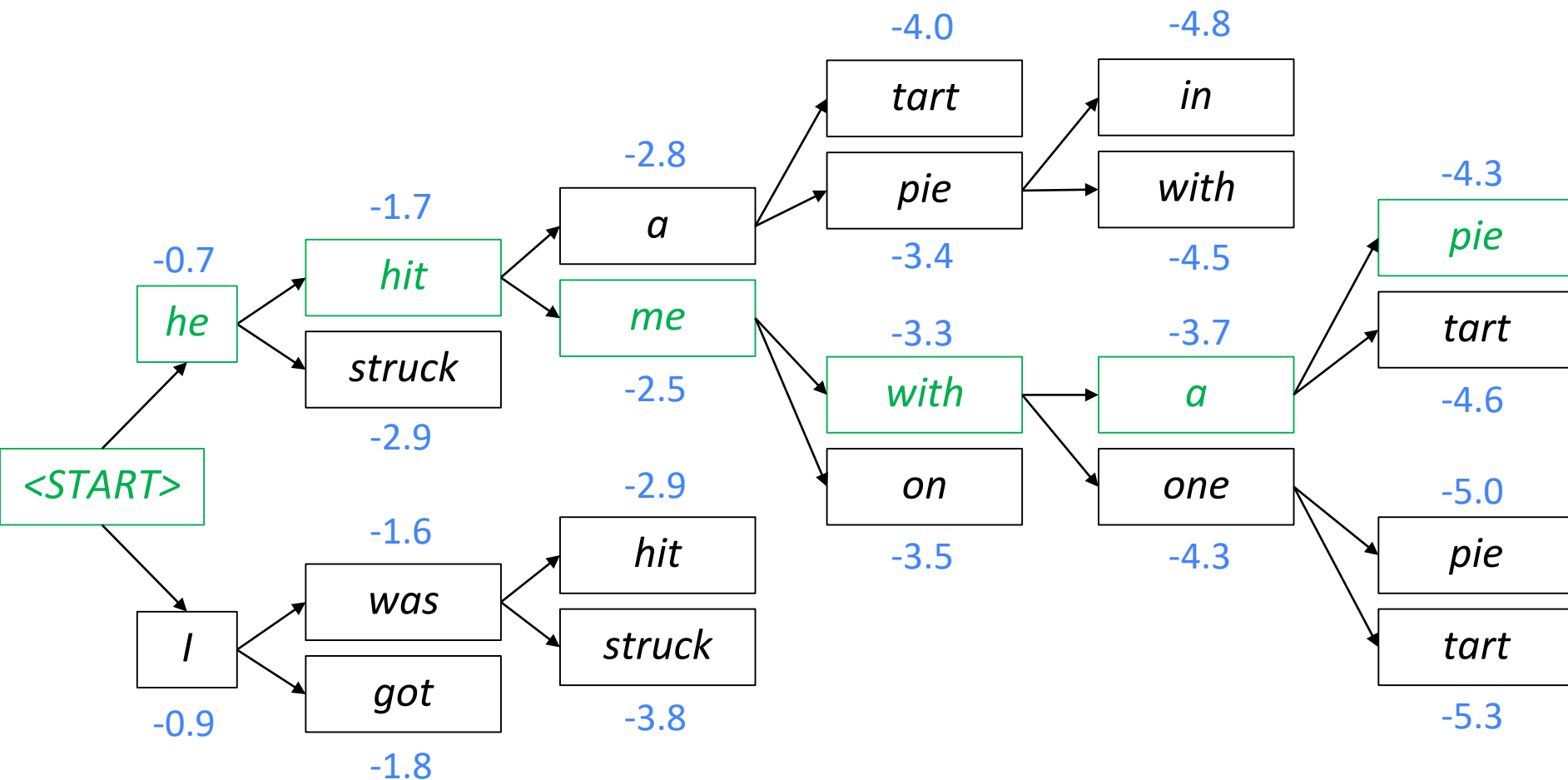
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



This is the top-scoring hypothesis!

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)

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!

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

English ▼	Spanish ▼
paper jam <small>Edit</small>	Mermelada de papel

[Open in Google Translate](#)

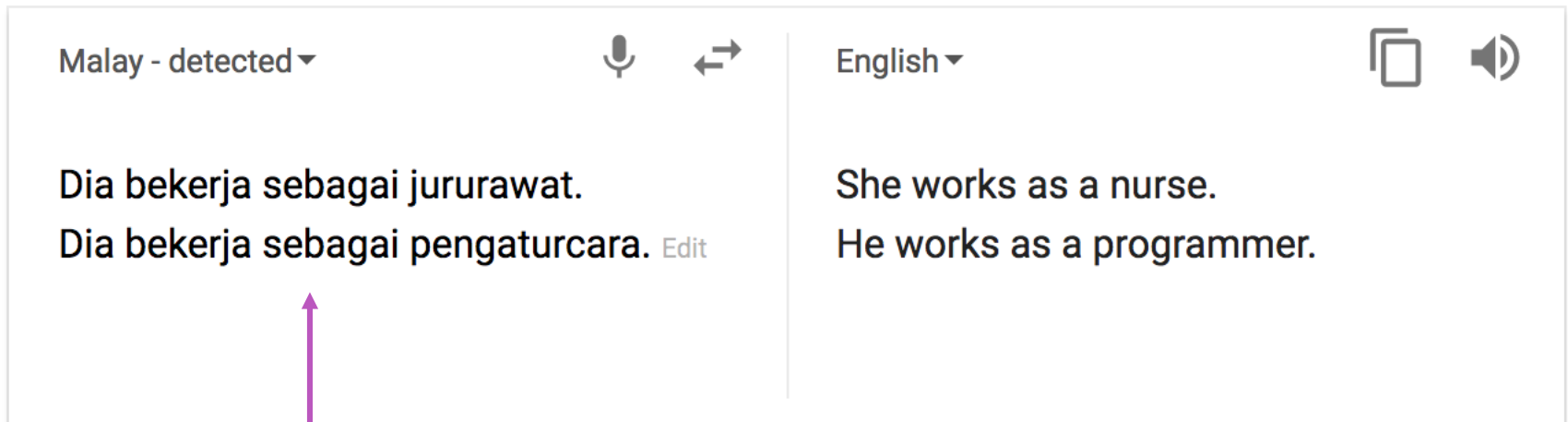
[Feedback](#)



?

So is Machine Translation solved?

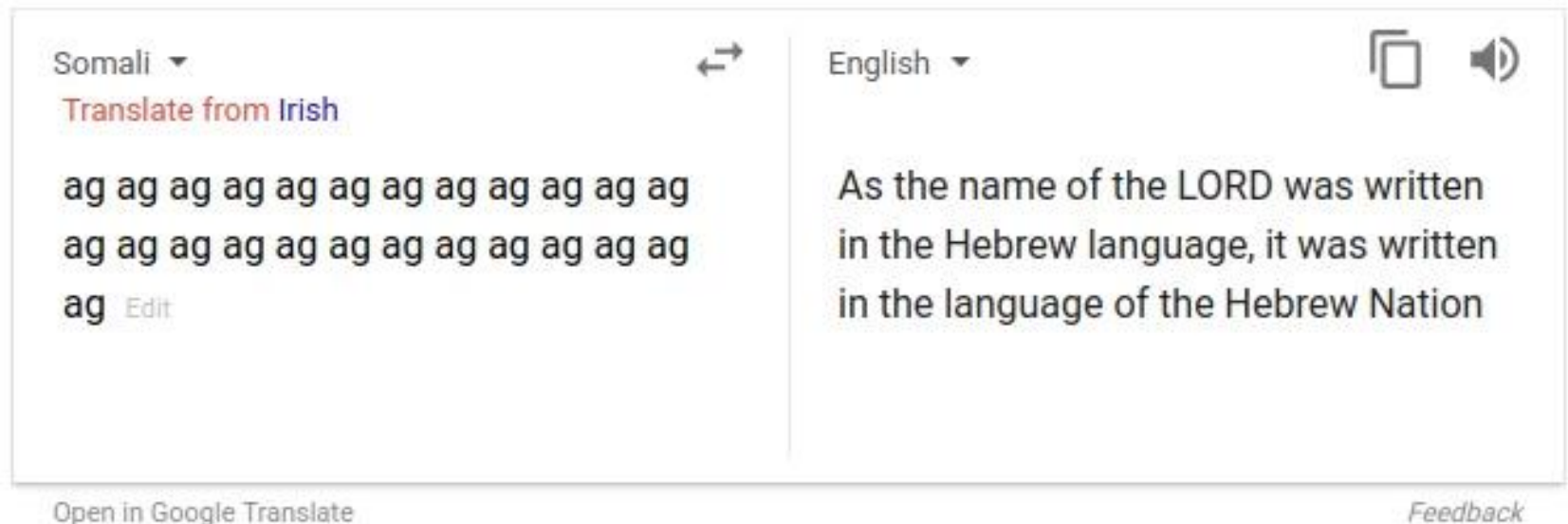
- **Nope!**
- NMT picks up **biases** in training data



Didn't specify gender

So is Machine Translation solved?

- Nope!
- Uninterpretable systems do strange things



Picture source: https://www.vice.com/en_uk/article/j5npeg/why-is-google-translate-spitting-out-sinister-religious-prophecies

Explanation: <https://www.skynettoday.com/briefs/google-nmt-prophecies>

References

- Dense Vectors, Dan Jurafsky
- Representation for Language: from Word Embeddings to Sentence Meanings, Christopher Manning, Stanford University, 2017
- Natural Language Processing with Deep Learning, Richard Socher, Stanford University
- More Word Vectors, Richard Socher, Stanford University
- Improving Distributional Similarity with Lessons Learned from Word Embeddings, Omer Levy,

