Scoring Phrase Pairs

- Phrase pair scoring: assign probabilities to all the phrase pairs extracted from the aligned corpus
- Scoring by relative probability:

$$\phi(f|e) = \frac{count(e,f)}{count(e)} = \frac{count(e,f)}{\sum_{f_i} count(e,f_i)}$$

Content

Decoding

Decoding is to search for the most likely target sentence e for a given source sentence f:

$$\hat{e} = \underset{e}{\operatorname{argmax}}\, p(e)p(f|e)$$

Decoding processing

Maria no dio una bofetada a la bruja verde

Build the translation from left to right:

1. Match the untranslated source words against the phrase table, and locate a matched source phrase
2. Append the target phrase to the end of the partial translation
3. Mark the source words translated
4. Iterate (1) to (3) until all source words are marked translated.

One to many translation

Many to one translation

Reordering: skip forward

Reordering: skip backward

Decoding processing

Maria no dio una bofetada a la bruja verde

Build the translation from left to right:

1. Match the untranslated source words against the phrase table, and locate a matched source phrase
2. Append the target phrase to the end of the partial translation
3. Mark the source words translated
4. Iterate (1) to (3) until all source words are marked translated.

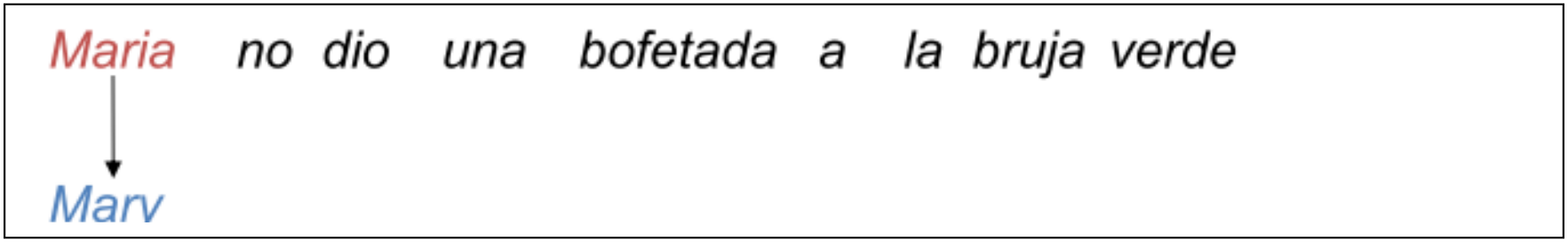
One to many translation

Many to one translation

Reordering: skip forward

Reordering: skip backward

Decoding processing



Build the translation from left to right:

1. Match the untranslated source words against the phrase table, and locate a matched source phrase
2. Append the target phrase to the end of the partial translation
3. Mark the source words translated
4. Iterate (1) to (3) until all source words are marked translated.

One to many translation

Many to one translation

Reordering: skip forward

Reordering: skip backward

Decoding processing

Maria no dio una bofetada a la bruja verde

Mary

Build the translation from left to right:

1. Match the untranslated source words against the phrase table, and locate a matched source phrase
2. Append the target phrase to the end of the partial translation
3. Mark the source words translated
4. Iterate (1) to (3) until all source words are marked translated.

One to many translation

Many to one translation

Reordering: skip forward

Reordering: skip backward

Decoding processing

Maria no dio una bofetada a la bruja verde

Mary

Build the translation from left to right:

1. Match the untranslated source words against the phrase table, and locate a matched source phrase
2. Append the target phrase to the end of the partial translation
3. Mark the source words translated
4. Iterate (1) to (3) until all source words are marked translated.

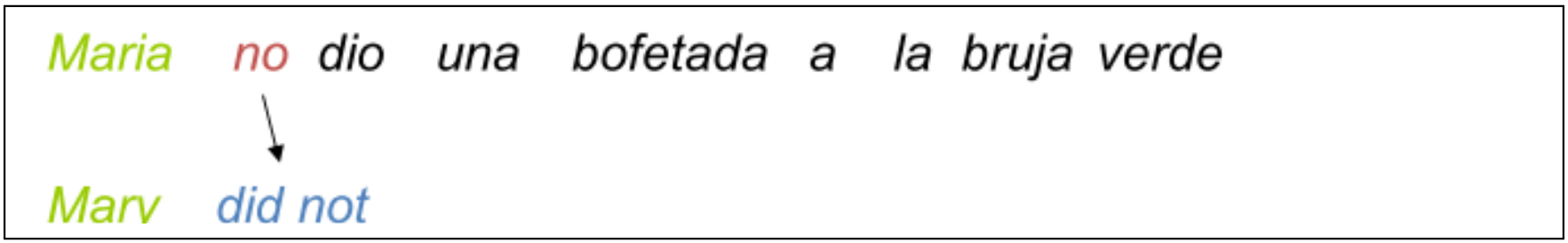
One to many translation

Many to one translation

Reordering: skip forward

Reordering: skip backward

Decoding processing



Build the translation from left to right:

1. Match the untranslated source words against the phrase table, and locate a matched source phrase
2. Append the target phrase to the end of the partial translation
3. Mark the source words translated
4. Iterate (1) to (3) until all source words are marked translated.

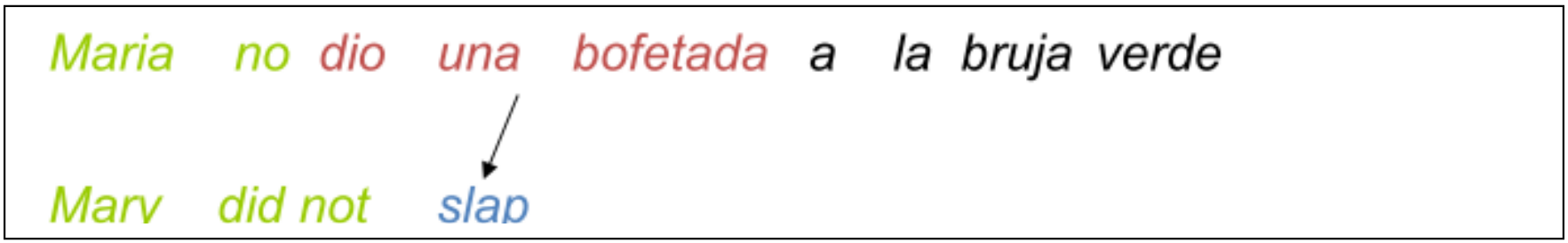
One to many translation

Many to one translation

Reordering: skip forward

Reordering: skip backward

Decoding processing



Build the translation from left to right:

1. Match the untranslated source words against the phrase table, and locate a matched source phrase
2. Append the target phrase to the end of the partial translation
3. Mark the source words translated
4. Iterate (1) to (3) until all source words are marked translated.

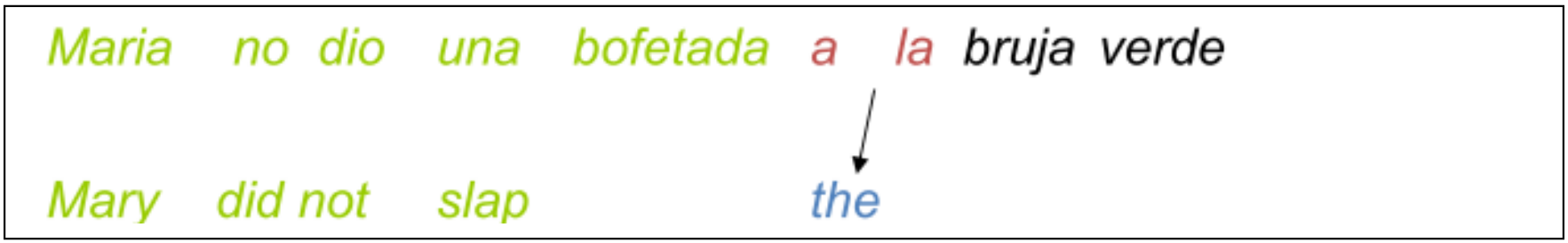
One to many translation

Many to one translation

Reordering: skip forward

Reordering: skip backward

Decoding processing



Build the translation from left to right:

1. Match the untranslated source words against the phrase table, and locate a matched source phrase
2. Append the target phrase to the end of the partial translation
3. Mark the source words translated
4. Iterate (1) to (3) until all source words are marked translated.

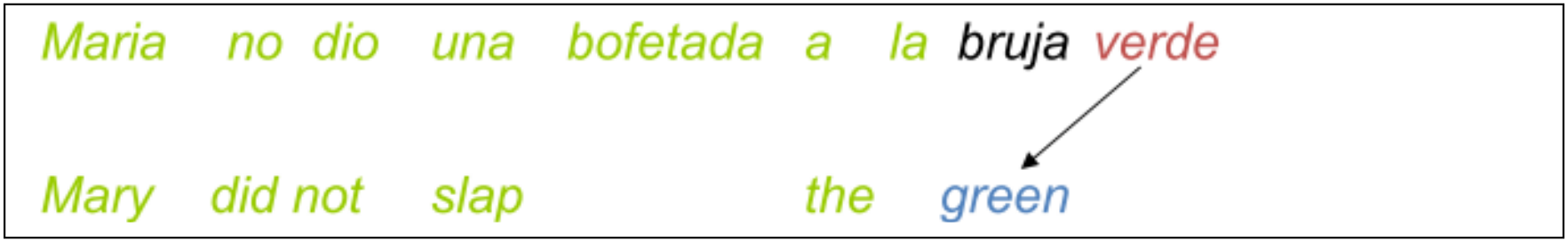
One to many translation

Many to one translation

Reordering: skip forward

Reordering: skip backward

Decoding processing



Build the translation from left to right:

1. Match the untranslated source words against the phrase table, and locate a matched source phrase
2. Append the target phrase to the end of the partial translation
3. Mark the source words translated
4. Iterate (1) to (3) until all source words are marked translated.

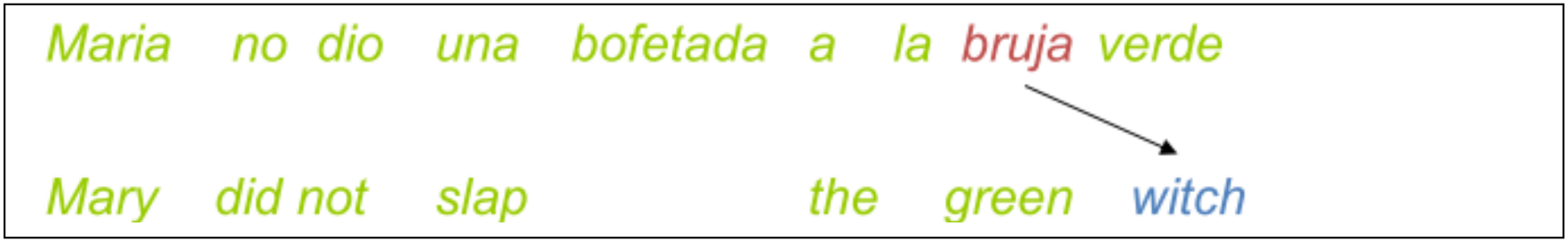
One to many translation

Many to one translation

Reordering: skip forward

Reordering: skip backward

Decoding processing



Build the translation from left to right:

1. Match the untranslated source words against the phrase table, and locate a matched source phrase
2. Append the target phrase to the end of the partial translation
3. Mark the source words translated
4. Iterate (1) to (3) until all source words are marked translated.

One to many translation

Many to one translation

Reordering: skip forward

Reordering: skip backward

Scoring hypotheses

- According to the noisy channel model, the score of a hypothesis includes two components:
  - Language model score
  - Phrase-based translation model score
- Shortcomings of the above scoring mechanism:
  - The importance of language model and translation model may be different
  - More factors should be considered in the decisions, for example, word reordering (to select a next source phrase to translate, or skip forwards or backwards), etc.

Log-linear Framework

- A log-linear framework was proposed to replace the noisy channel framework to overcome the above shortcomings:

$$p(e|f) = \frac{\exp(\sum_{i=1}^{n} \lambda_i h_i(e, f))}{\sum_{e'} \exp(\sum_{i=1}^{n} \lambda_i h_i(e', f))}$$

$$\hat{e} = \underset{e}{\operatorname{argmax}} \sum_{i=1}^{n} \lambda_i h_i(e, f)$$

Log-linear Framework

- Advantages of the log-linear framework:
  - Abitrary number of user-defined features (h_i) can be added in the model.
  - Weights (λ_i) can be tuned to balance the importance among the features.

Fine-tuning the log-linear model

- A held-out data set (usually called development set) is used to train the parameters (λ_i) for the log-linear model;
- The training process of the log-linear model is called fune-tuning in SMT;
- Unlike in neural network training, the gradient descend algorithm is not applicable here, because the input data is discrete symbols and the function is not differetiable.
- Fune-tuning algorithms, e.g. MERT, or MIRA, were developed to fine-tune the log-linear model for SMT.

Features for log-linear model in SMT

- Under the noisy channel framework, only two features are considered:
  - Target language model.
  - Backward translation model.
- Under the log-linear framework, more features can be considered, typically including:
  - Forward translation model.
  - Forward and backward lexicalized translation models.
  - Additional language models. (e.g. with different ngram orders)
  - Word reordering models.
  - Length of the target sentence.
  - User-defined dictionaries.

Content

What is Neural Machine Translation?

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

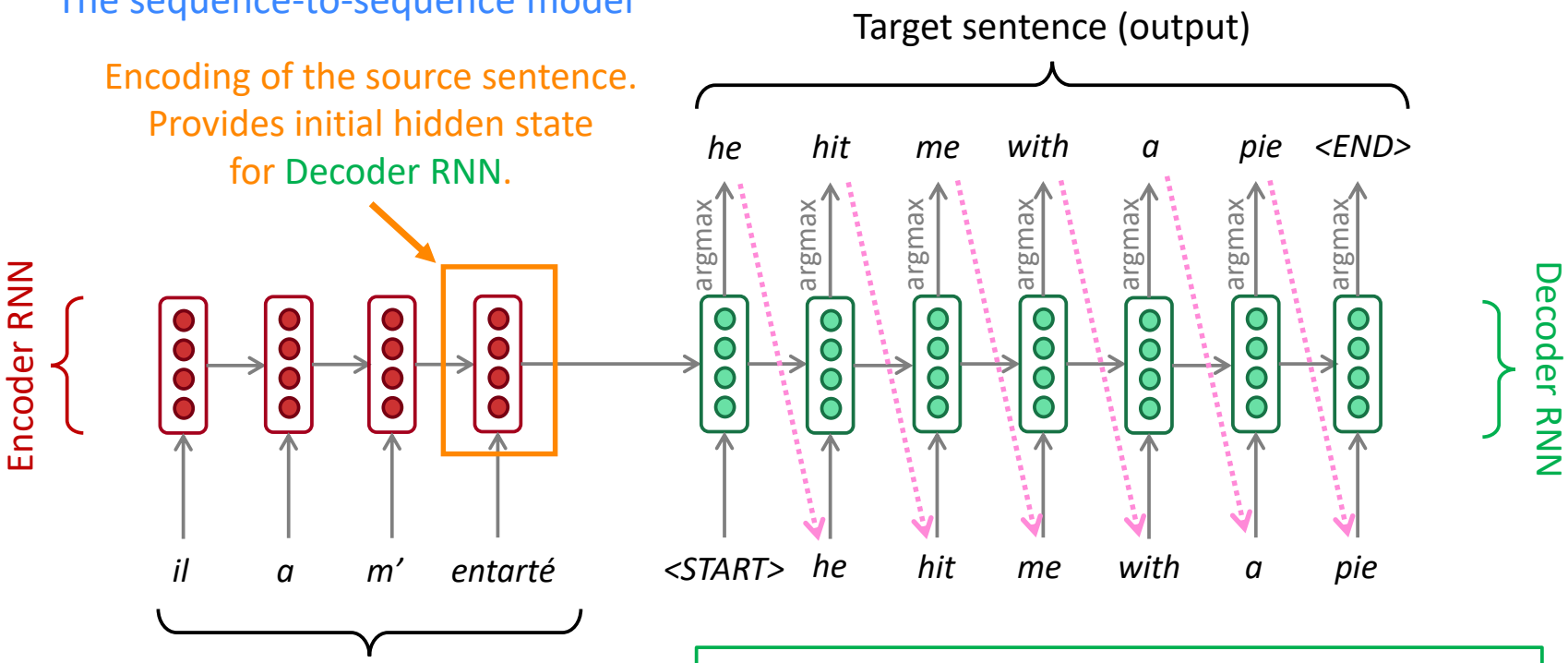
- The neural network architecture is called sequence-to-sequence (aka seq2seq) and it involves *two* RNNs.

23

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Neural Machine Translation (NMT)

The sequence-to-sequence model



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

Encoder RNN produces an encoding of the source sentence.

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

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

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)

25

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

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 $P(y|x)$:

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

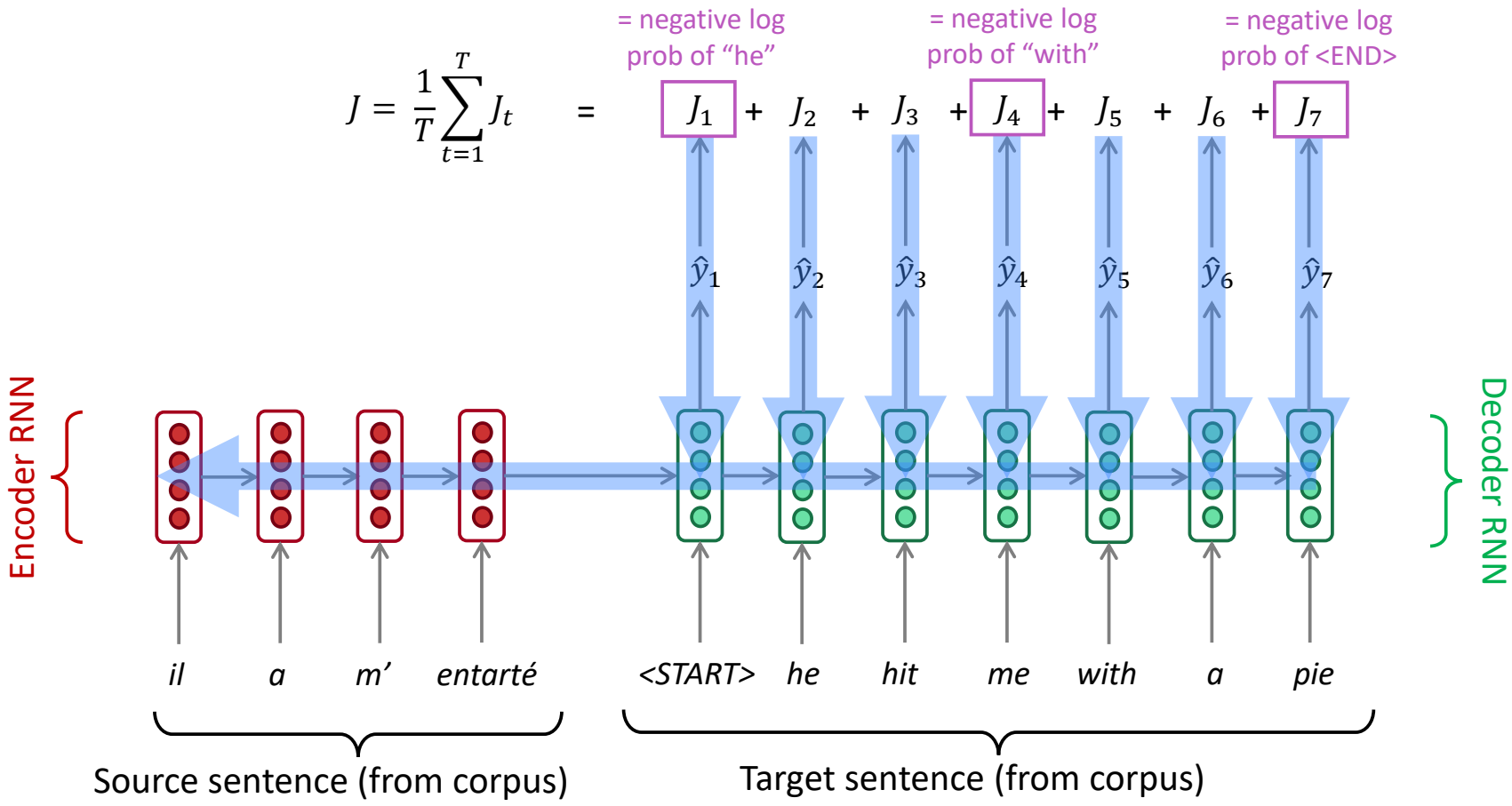
Probability of next target word, given target words so far and source sentence *x*

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

26

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Training a Neural Machine Translation system



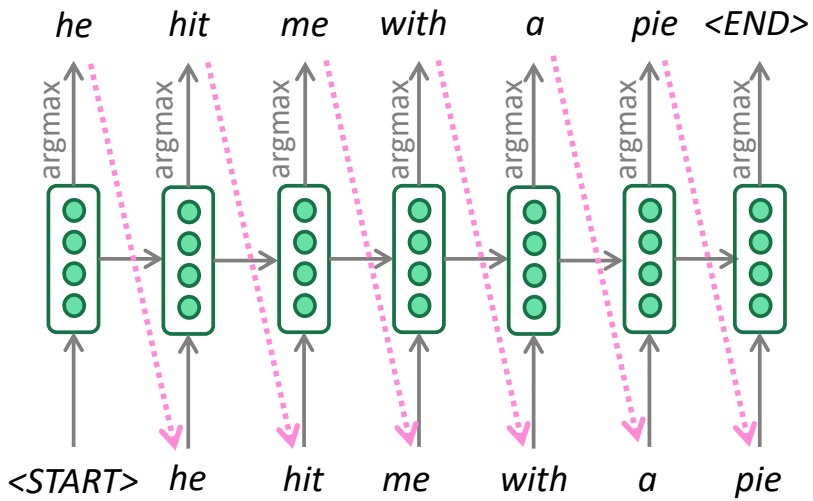
Seq2seq is optimized as a **single system.**
Backpropagation operates "*end-to-end*".

27

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

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?**

28

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

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?

29

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Exhaustive search decoding

- 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*
  - 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!

30

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

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 $y_1, \ldots, y_t$ has a score which is its log probability:

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

  - 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!

31

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

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>

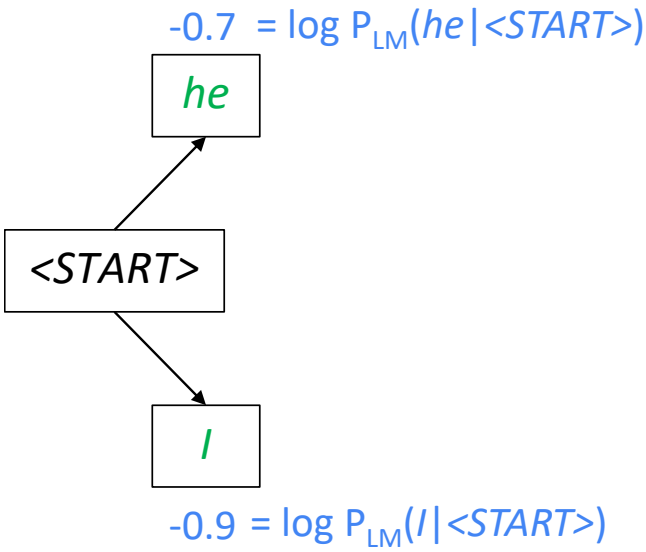
32

Calculate prob dist of next word

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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

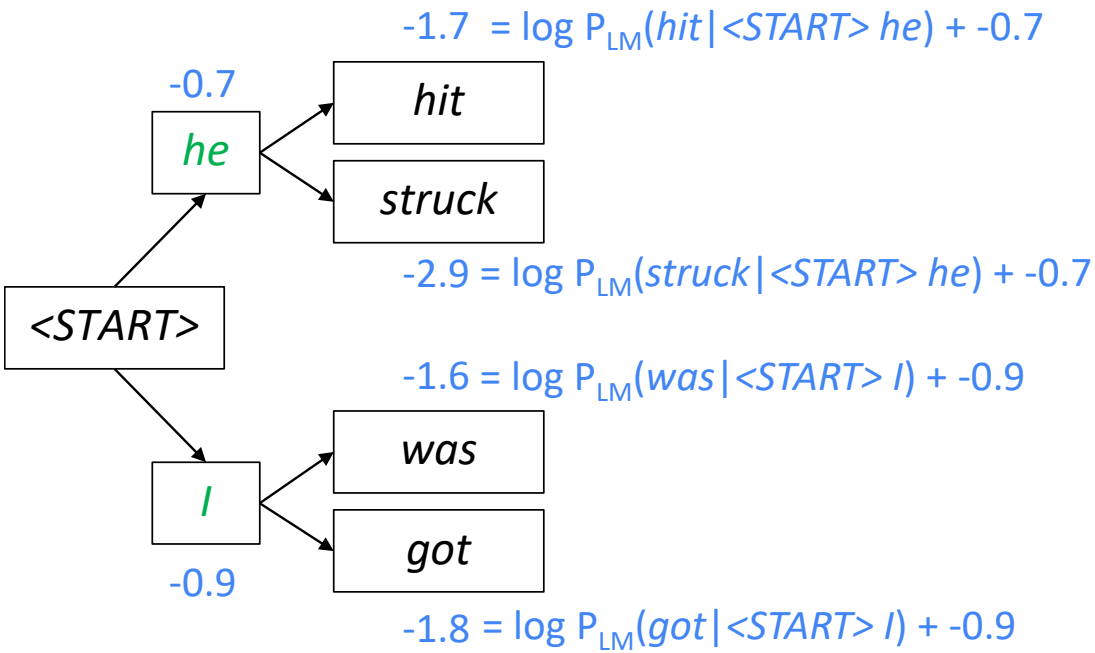


33

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



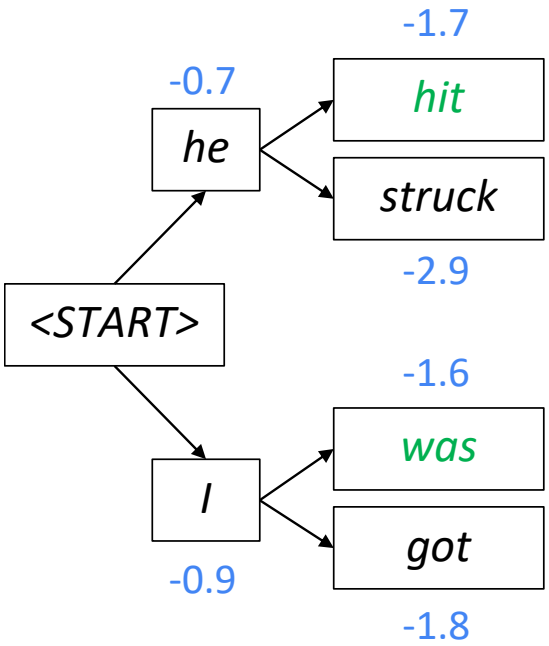
For each of the *k* hypotheses, find top *k* next words and calculate scores

34

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



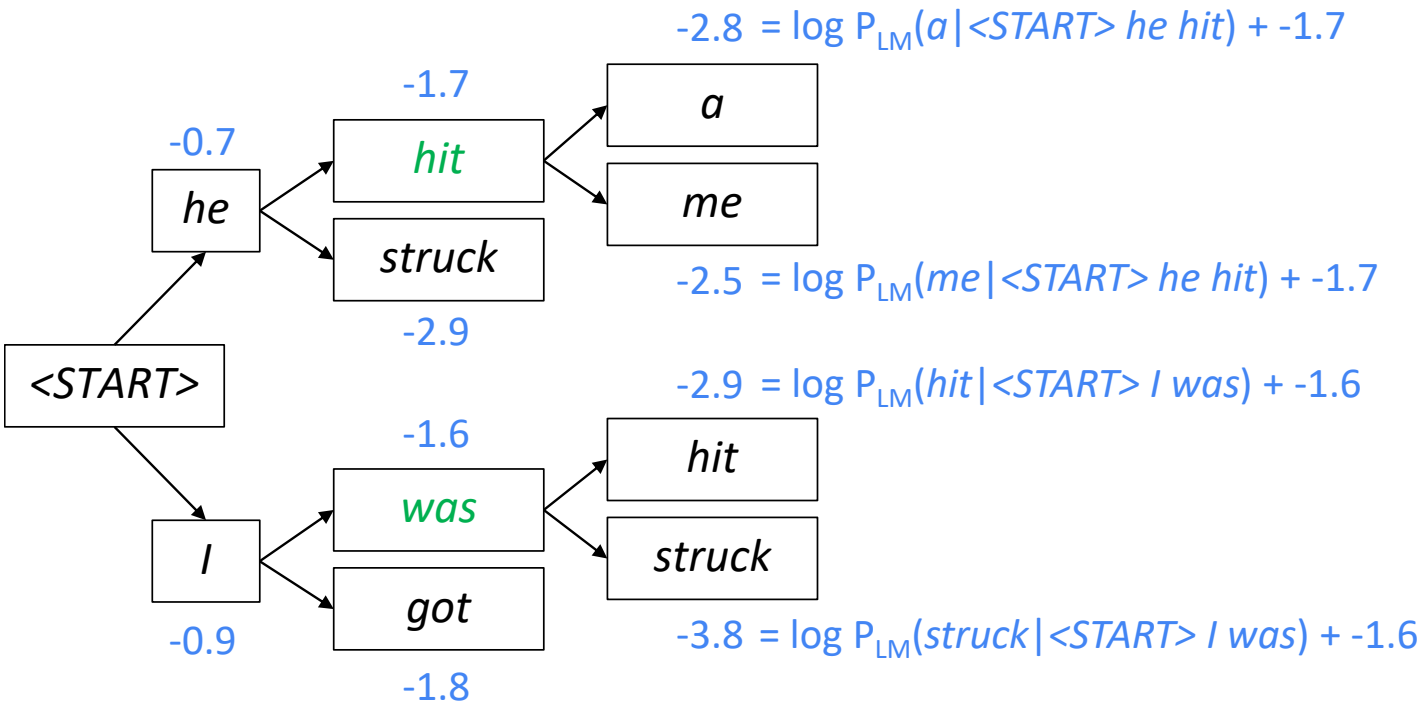
Of these k^2 hypotheses, just keep k with highest scores

35

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



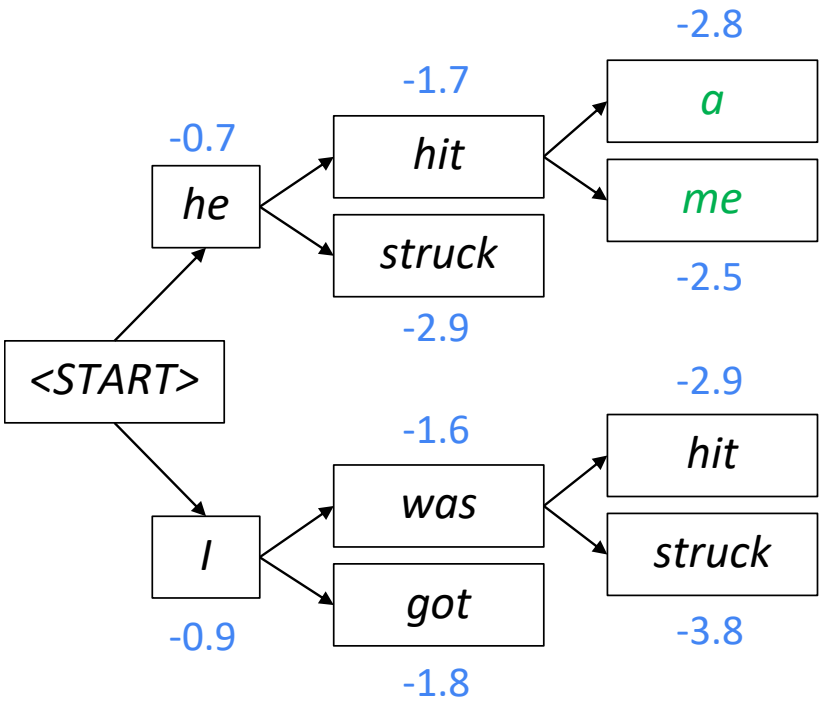
For each of the *k* hypotheses, find top *k* next words and calculate scores

36

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



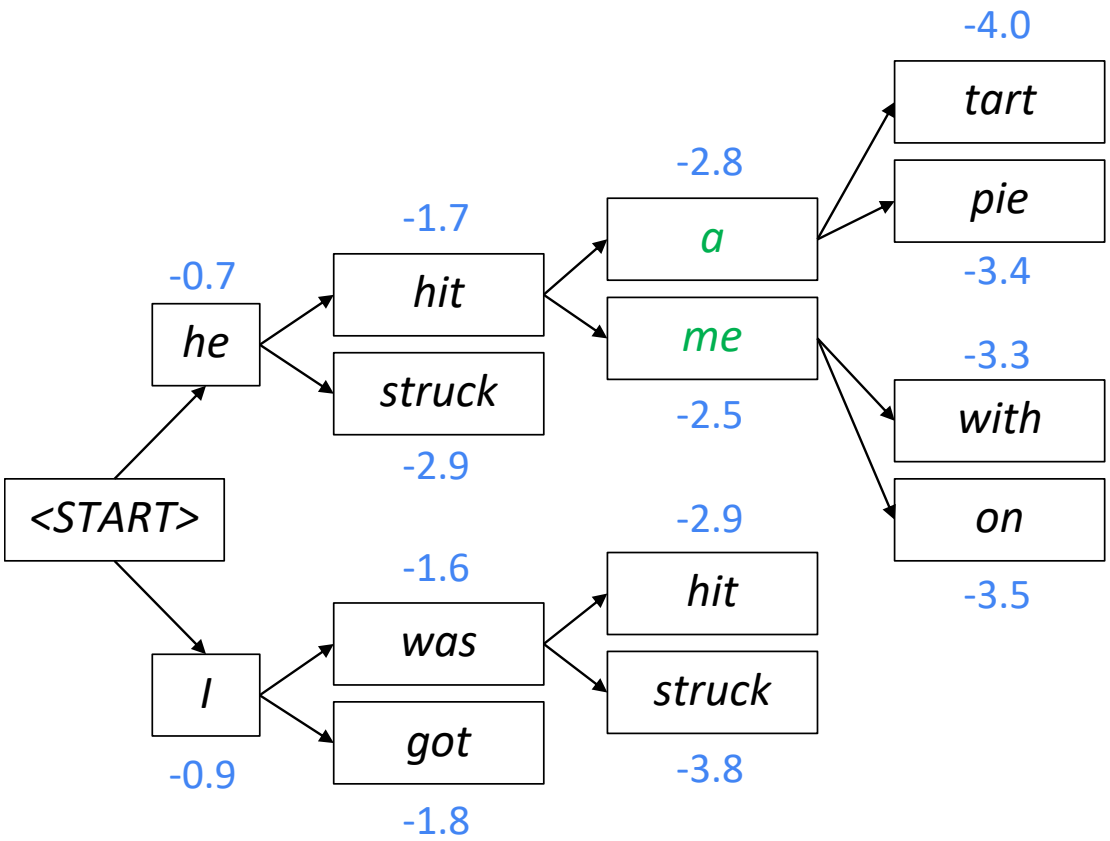
Of these k^2 hypotheses, just keep k with highest scores

37

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



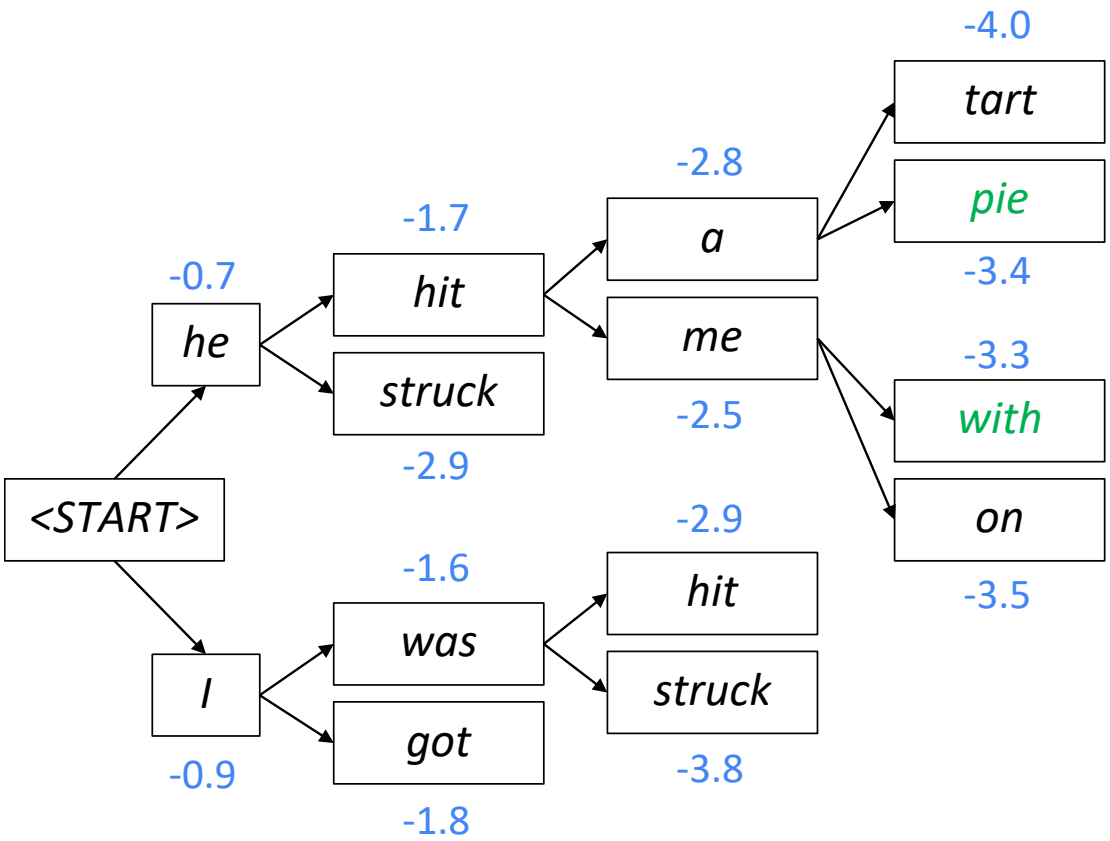
For each of the *k* hypotheses, find top *k* next words and calculate scores

38

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



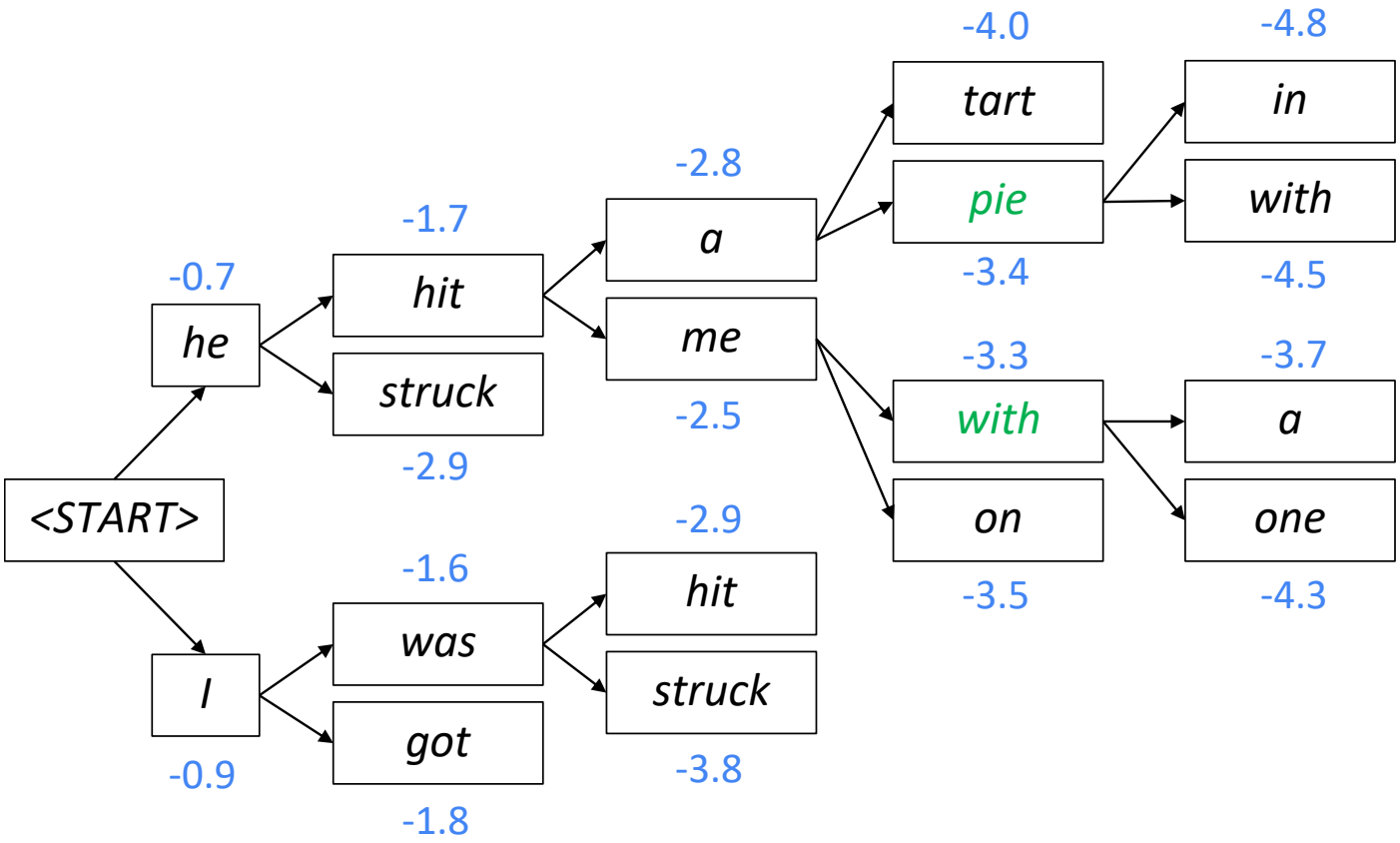
Of these k^2 hypotheses, just keep k with highest scores

39

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



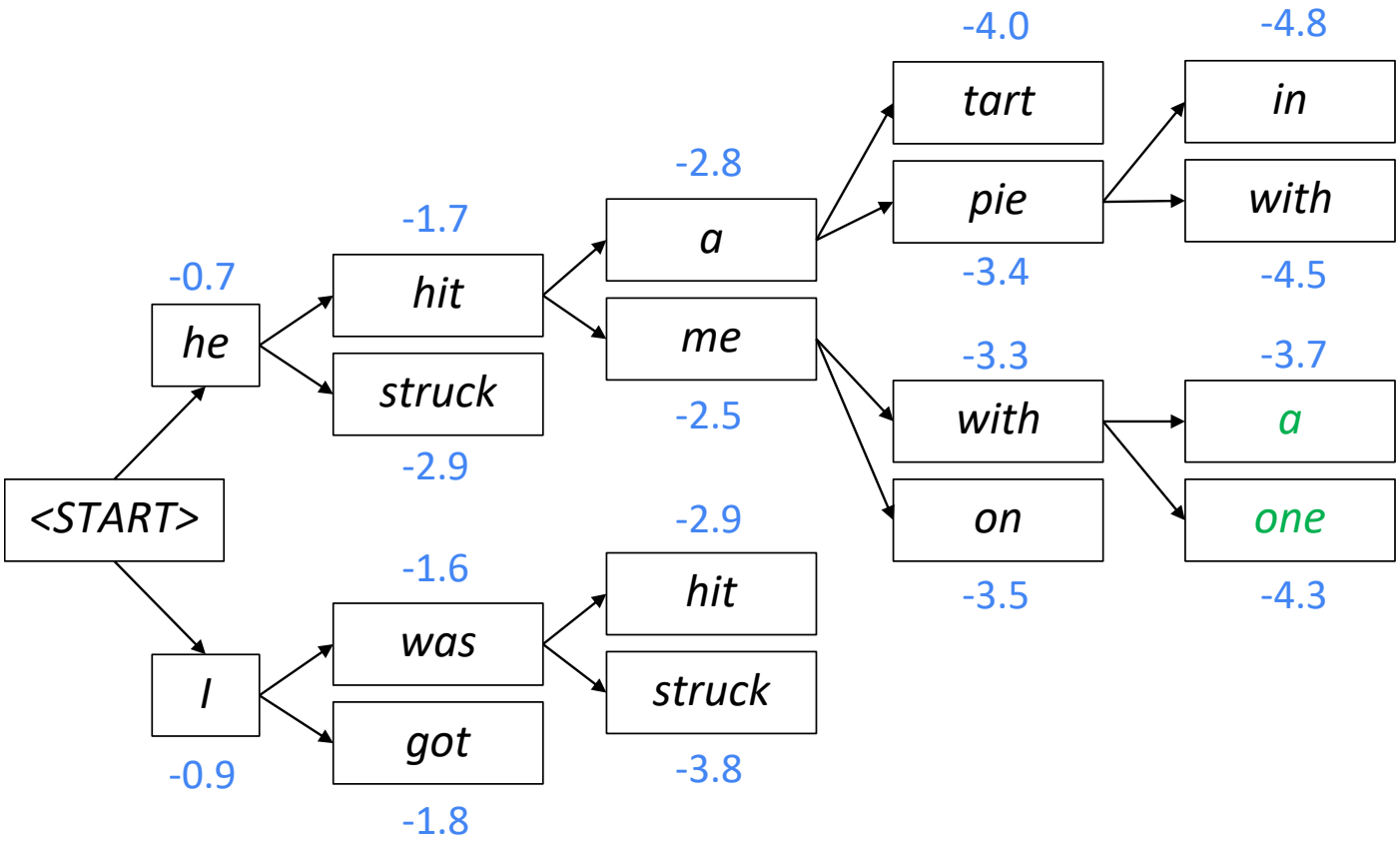
For each of the *k* hypotheses, find top *k* next words and calculate scores

40

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



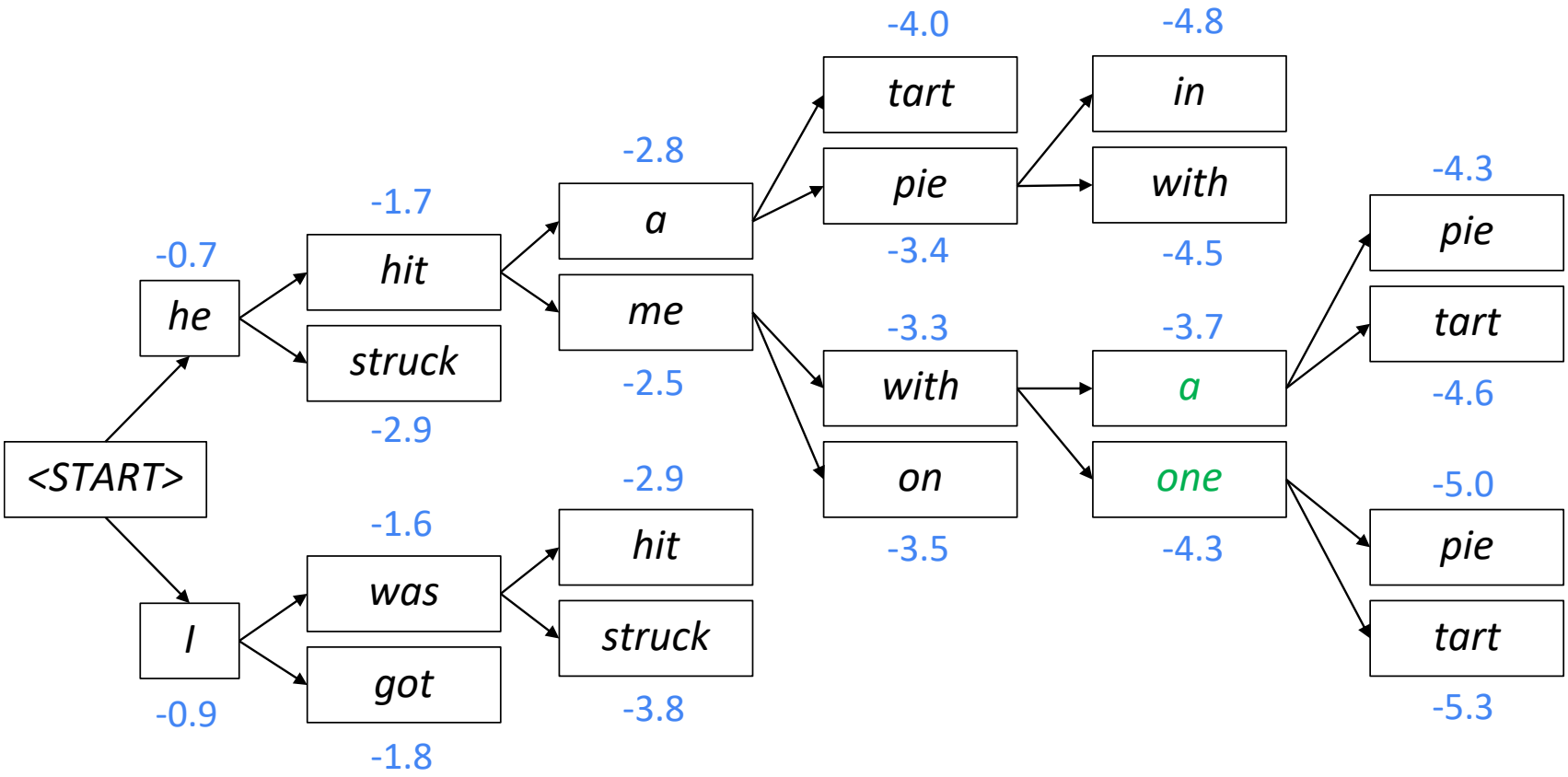
Of these k^2 hypotheses, just keep k with highest scores

41

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



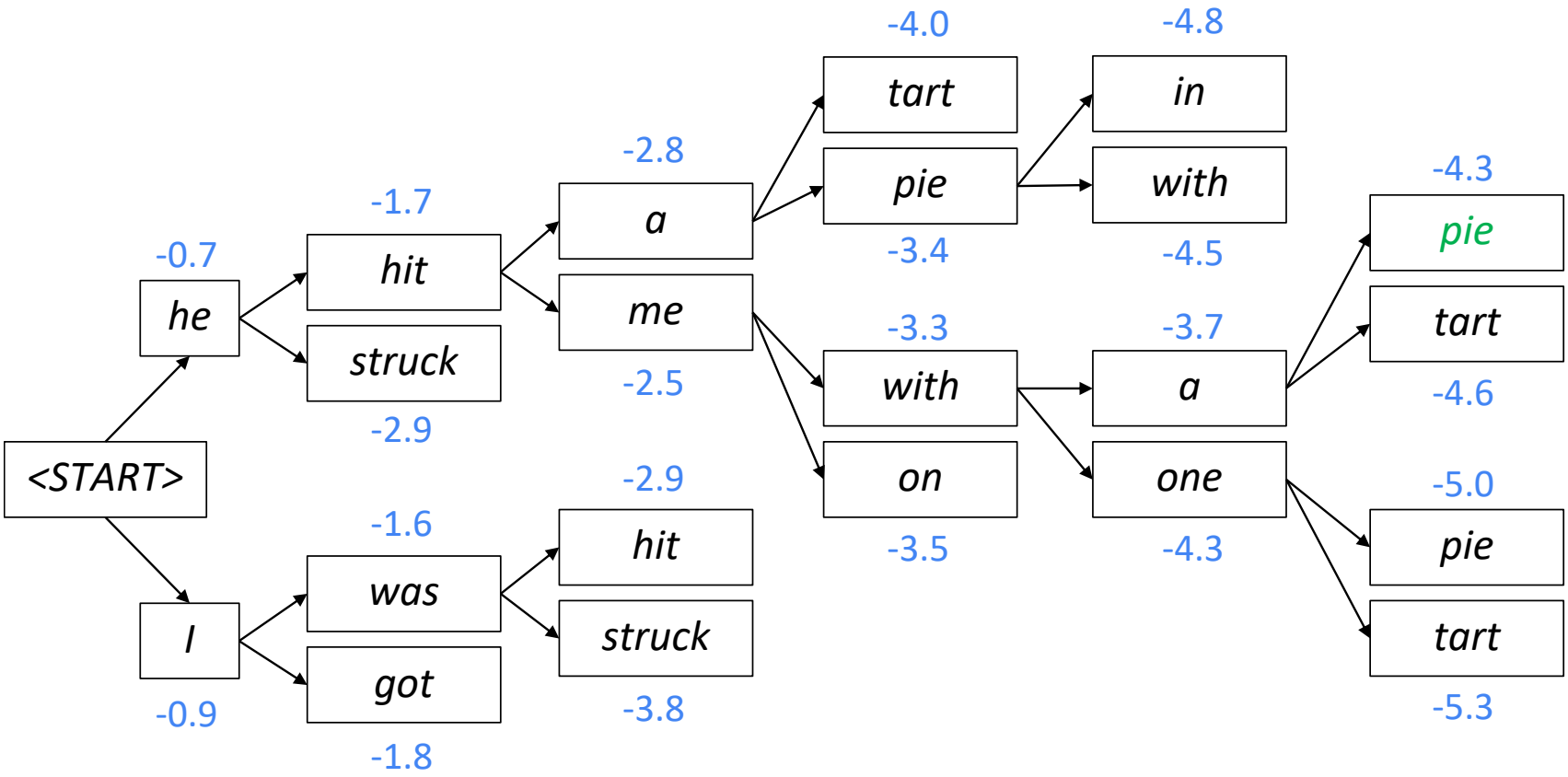
For each of the *k* hypotheses, find top *k* next words and calculate scores

42

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



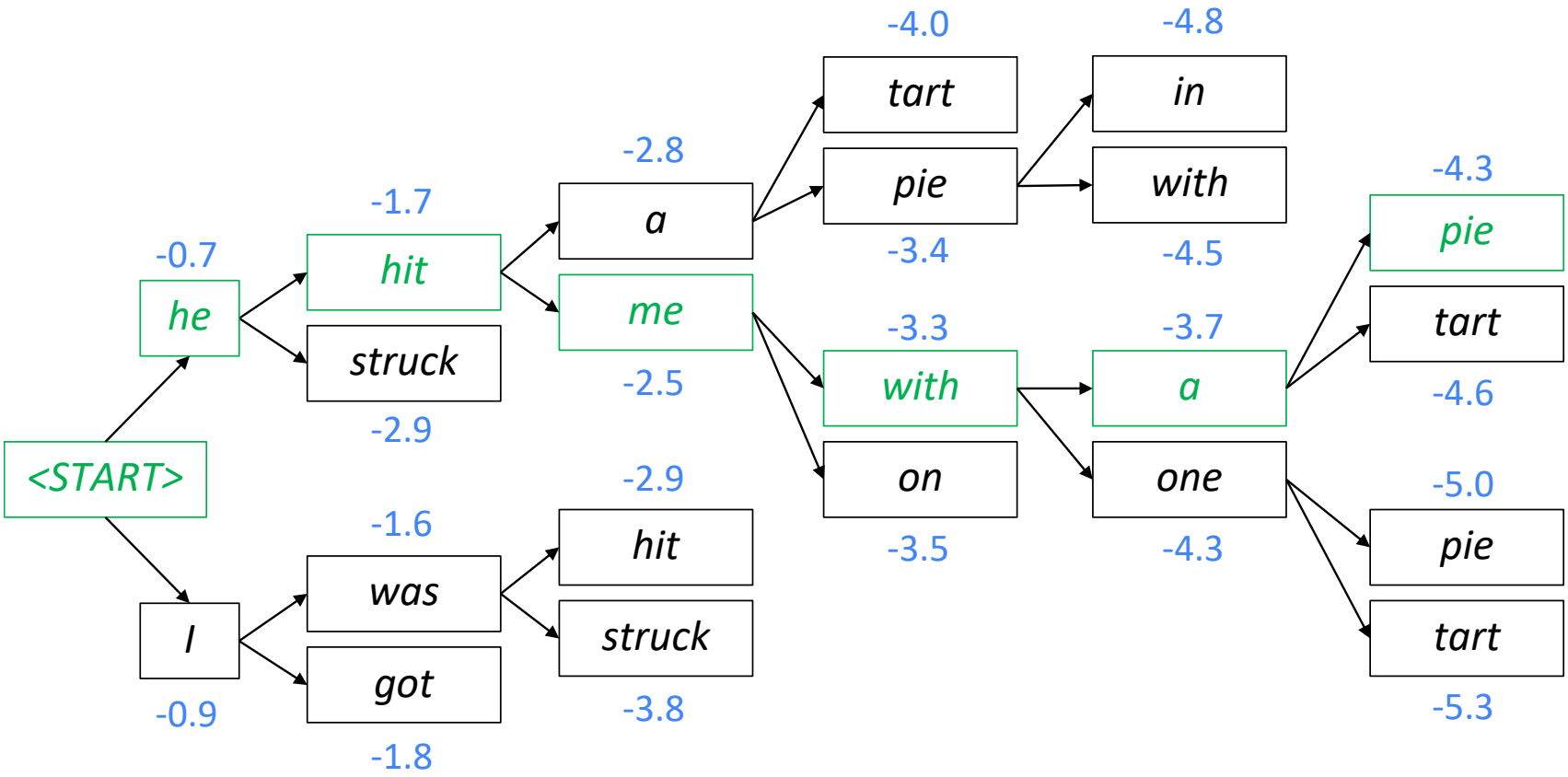
This is the top-scoring hypothesis!

43

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: example

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



Backtrack to obtain the full hypothesis

44

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

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)

45

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Beam search decoding: finishing up

- We have our list of completed hypotheses.
- How to select top one with highest score?

- Each hypothesis $y_1, \ldots, y_t$ 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 with this: longer hypotheses have lower scores

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

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

46

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

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

47

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

Disadvantages of NMT?

Compared to SMT:

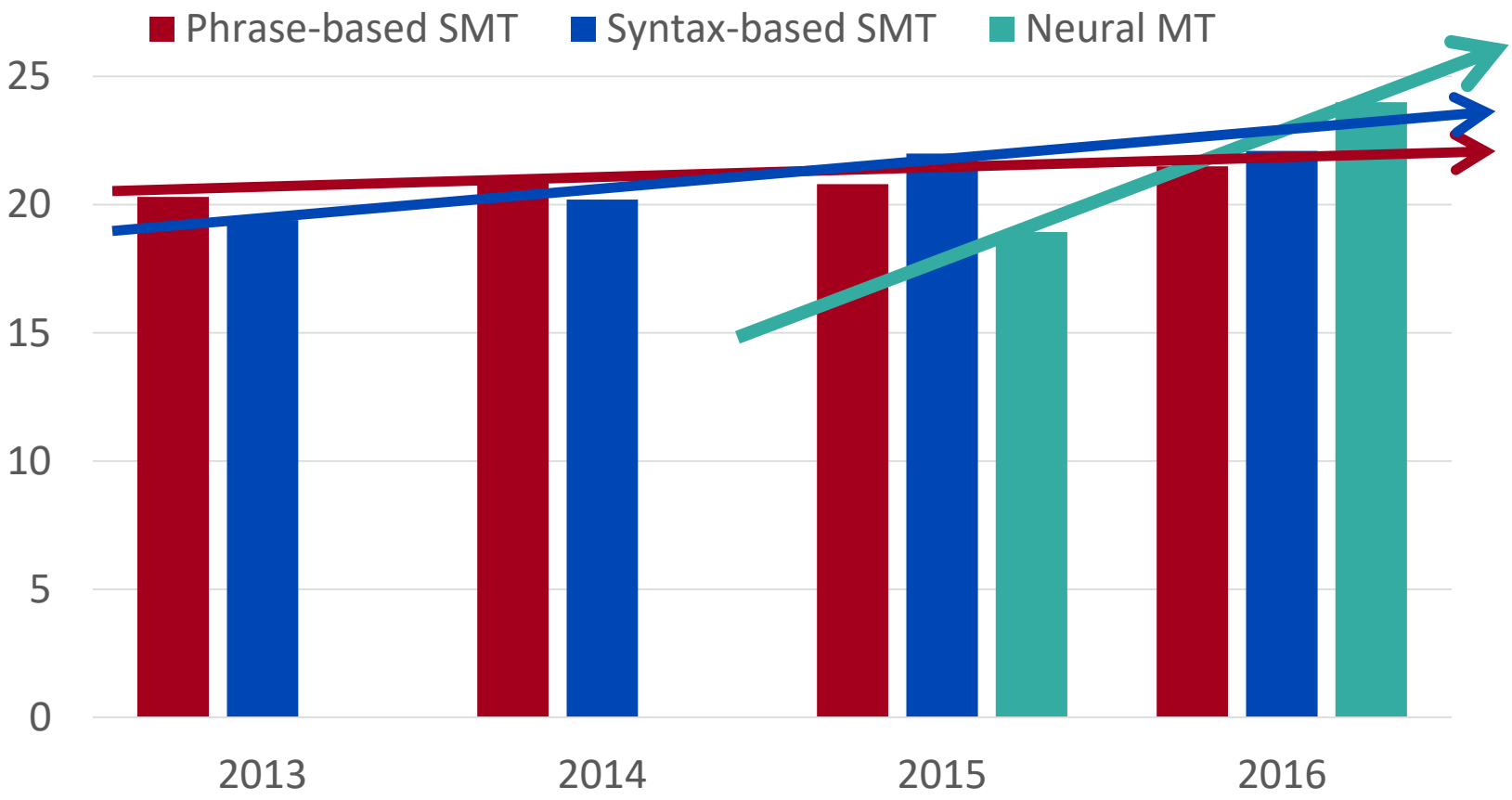
- NMT is less interpretable
  - Hard to debug
- NMT is difficult to control
  - For example, can't easily specify rules or guidelines for translation
  - Safety concerns!

48

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

MT progress over time

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



50

Source: http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

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

51

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

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

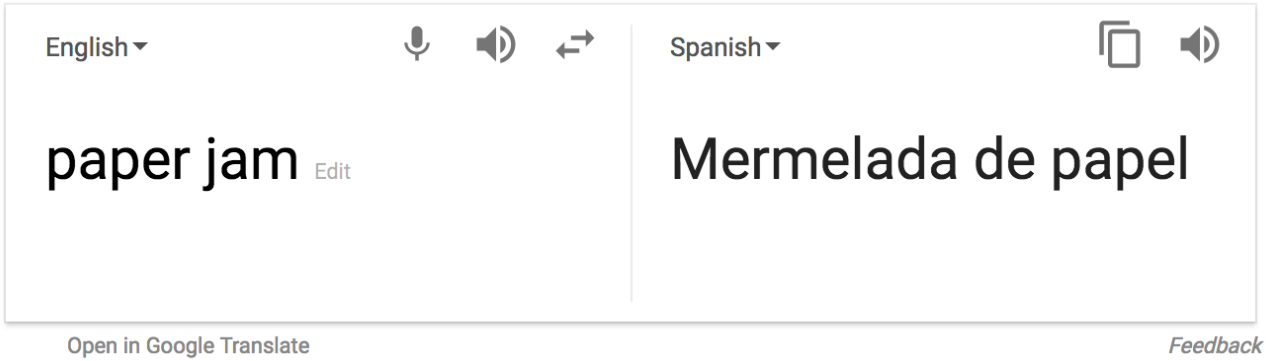
Further reading: *"Has AI surpassed humans at translation? Not even close!"*
https://www.skynettoday.com/editorials/state_of_nmt

52

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

So is Machine Translation solved?

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



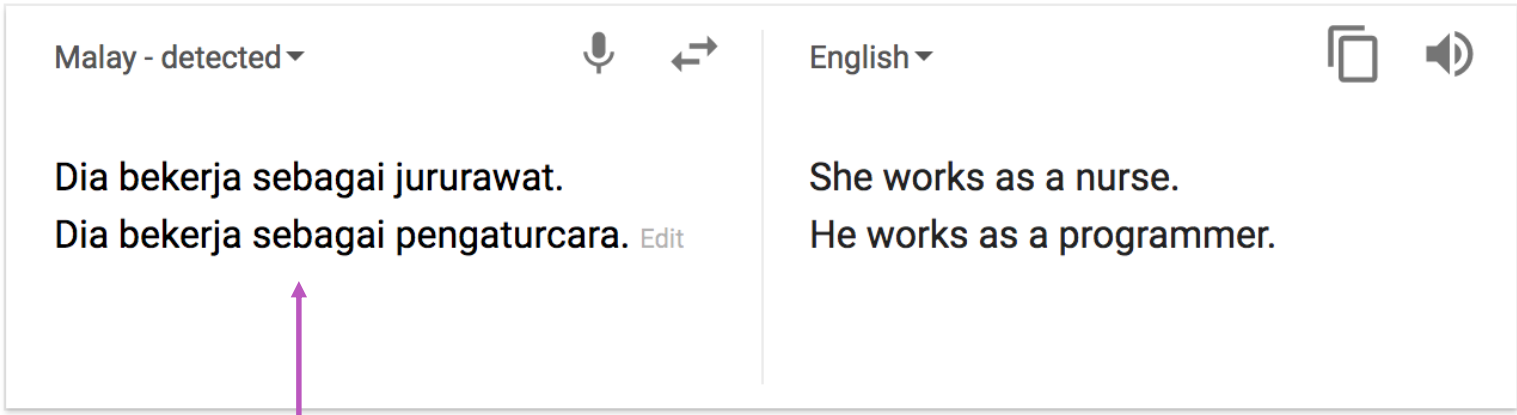
?

53

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)

So is Machine Translation solved?

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



Didn't specify gender

Source: https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c

Christopher Manning, Natural Language Processing with Deep Learning, 2019 (slides)