

344.063 KV Special Topic:

Natural Language Processing with Deep Learning

Sequence-to-Sequence RNNs for Text Summarization



Navid Rekab-saz

navid.rekabsaz@jku.at

Agenda

- Document summarization
- Abstractive summarization with seq2seq
- Extractive summarization with RNNs

Agenda

- **Document summarization**
- Abstractive summarization with seq2seq
- Extractive summarization with RNNs

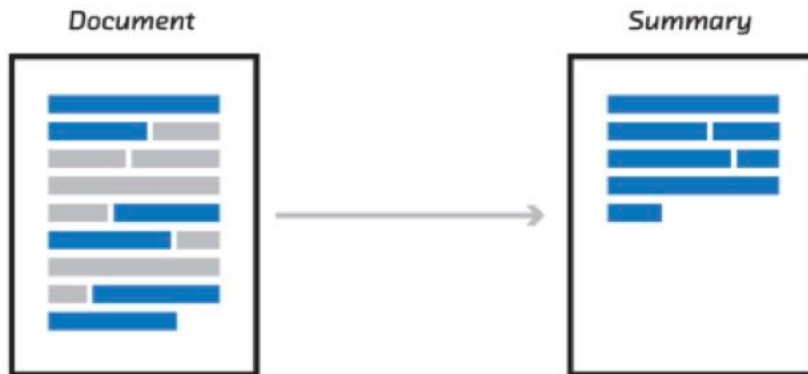
Text Summarization

- The task of summarizing the key information content of a text (document) $X = \{x_1, x_2, \dots, x_N\}$ into a summary $Y = \{y_1, y_2, \dots, y_M\}$
 - Summary is concise and (much) shorter than document
- Some datasets:
 - Gigaword: first one or two sentences of a news article
 - CNN/DailyMail: news article
 - Wikihow: full how-to article

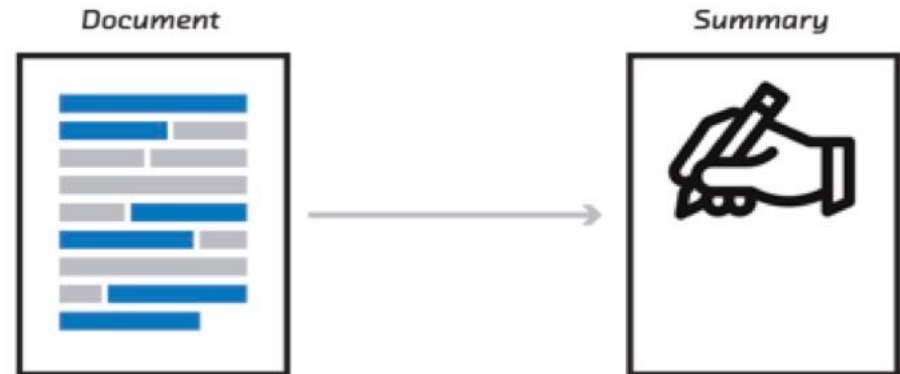
Summarization

- Extractive Summarization
 - **Selecting sections** (typically sentences) of the document
 - A model decides if a section of document should be selected for the summary
- Abstractive Summarization
 - Writing (**generating**) **new summary text** for the document
 - A language generation task conditioned on the document

Extractive Summarization



Abstractive Summarization



Abstractive Summarization

CNN/DailyMail dataset example

Document

SAN FRANCISCO, California (Reuters) -- Sony has cut the price of the PlayStation 3 by \$100, or 17 percent, in the United States, a move that should boost the video game console's lackluster sales.

Starting Monday, the current PS3 60 gigabyte model will cost \$499 -- a \$100 price drop. The PlayStation 3, which includes a 60-gigabyte hard drive and a Blu-ray high-definition DVD player, will now cost \$500, or \$20 more than the most expensive version of Microsoft's Xbox 360.

The PS3 still costs twice that of Nintendo's Wii console, whose \$250 price and motion-sensing controller have made it a best-seller despite its lack of cutting-edge graphics and hard disk.

"Our initial expectation is that sales should double at a minimum," Jack Tretton, chief executive of Sony Computer Entertainment America, said in an interview.

"We've gotten our production issues behind us on the PlayStation 3, reaching a position to pass on the savings to consumers, and our attitude is the sooner the better."

...

Summary

- Sony drops price of current 60GB PlayStation 3 console by \$100 in U.S.
- PS3 still costs twice that of Nintendo's best-selling Wii console, which is \$250
- Some expect Microsoft to respond with its first price cuts on the Xbox 360
- Sony to revise PS3 console with bigger 80GB hard drive

Summarization – Evaluation

- **ROUGE-N**: overlap of n -grams between output and reference summary
 - **ROUGE-1**: the overlap of *unigrams*
 - **ROUGE-2**: the overlap of *bigrams*
 - ...

$$\text{ROUGE-N} = \frac{|n\text{-grams}(\hat{Y}) \cap n\text{-grams}(Y)|}{|n\text{-grams}(Y)|}$$

Y and \hat{Y} are the reference and output summary, respectively. n -grams returns the set of all possible n -grams of the given text.

Summarization – Evaluation

- **ROUGE-L** is based on the length of the **longest common subsequence** between the output and reference summary

$$\text{ROUGE-L} = \frac{LCS(\hat{Y}, Y)}{|\hat{Y}|}$$

LCS is the longest common subsequence of the two given texts.

- **ROUGE-L**
 - does not require consecutive matches but **in-sequence** matches
 - Example from Wikipedia (*LCS*=3):
 - *Y* : **this is** some text that will be **changed**
 - *Ŷ* : **this is** the **changed** text
 - reflects sentence structure
 - don't need a predefined *n*-gram length

Summarization – Evaluation

- ROUGE (in the discussed definitions) is a recall-based measure
 - ROUGE can also be defined as a precision-based, as well as F-measure

Example

- Y : “police hugged the gunman”
- \hat{Y}_1 : “police hug the gunman”
- \hat{Y}_2 : “the gunman hug police”
- ROUGE-2 of both \hat{Y}_1 and \hat{Y}_2 results in the same values!
 - In both \hat{Y}_1 and \hat{Y}_2 , “the gunman” is the only common bigram with the reference
- $LCS(\hat{Y}_1, Y) = \text{“police the gunman”} \rightarrow \text{ROUGE-L}(\hat{Y}_1, Y) = 0.75$
- $LCS(\hat{Y}_2, Y) = \text{“the gunman”} \rightarrow \text{ROUGE-L}(\hat{Y}_2, Y) = 0.5$

Agenda

- Document summarization
- **Abstractive summarization with seq2seq**
- Extractive summarization with RNNs

Sequence in – sequence out!

- Several NLP tasks are defined as:
 - Given the source sequence $X = \{x^{(1)}, x^{(2)}, \dots, x^{(L)}\}$
 - Create/Generate the target sequence $Y = \{y^{(1)}, y^{(2)}, \dots, y^{(T)}\}$

X

Y

Was mich nicht umbringt, macht mich stärker.

F. Nietzsche

Machine Translation

What does not kill me makes me stronger.

Then the woman went to the bank to deposit her cash .

POS Tagging

RB DT NN VBD TO DT NN TO VB PRP\$ NN .

How tall is Stephansdom?

Semantic parsing

[Heightof, ., Stephansdom]

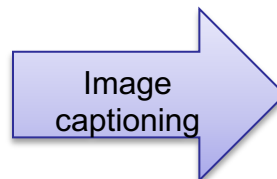
Sequence in – sequence out!

- Tasks such as:

- Machine Translation (source language → target language)
- Summarization (long text → short text)
- Dialogue (previous utterances → next utterance)
- Code generation (natural language → SQL/Python code)
- Named entity recognition
- Dependency/semantic/ POS Parsing (input text → output parse as sequence)

but also ...

- Image captioning (image → caption)
- Automatic Speech Recognition (speech → manuscript)



some elephants standing
around a tall tree

Sequence-to-sequence model

- Sequence-to-sequence model (aka **seq2seq**) is the neural network architecture to approach ...
 - given the source sequence $X = \{x^{(1)}, x^{(2)}, \dots, x^{(L)}\}$,
 - generate the target sequence $Y = \{y^{(1)}, y^{(2)}, \dots, y^{(T)}\}$
- A seq2seq model typically creates a **model** to estimate the **conditional probability**:

$$P(Y|X)$$

- and then generates a new sequence Y^* by solving:

$$Y^* = \underset{Y}{\operatorname{argmax}} P(Y|X)$$

Seq2seq model

- A seq2seq model in many cases can be as a **conditional Language Model**
- It calculates the probability of the next word of target sequence, conditioned on the previous words of target sequence and the source sequence:

for $y^{(1)} \rightarrow P(y^{(1)} | X)$

for $y^{(2)} \rightarrow P(y^{(2)} | X, y^{(1)})$

...

for $y^{(i)} \rightarrow P(y^{(i)} | X, y^{(1)}, \dots, y^{(i-1)})$

... and for **whole the target sequence**:

$$P(Y|X) = P(y^{(1)} | X) \times P(y^{(2)} | X, y^{(1)}) \times \dots \times P(y^{(T)} | X, y^{(1)}, \dots, y^{(T-1)})$$

$$P(Y|X) = \prod_{t=1}^T P(y^{(t)} | X, y^{(1)}, \dots, y^{(t-1)})$$

Seq2seq – steps

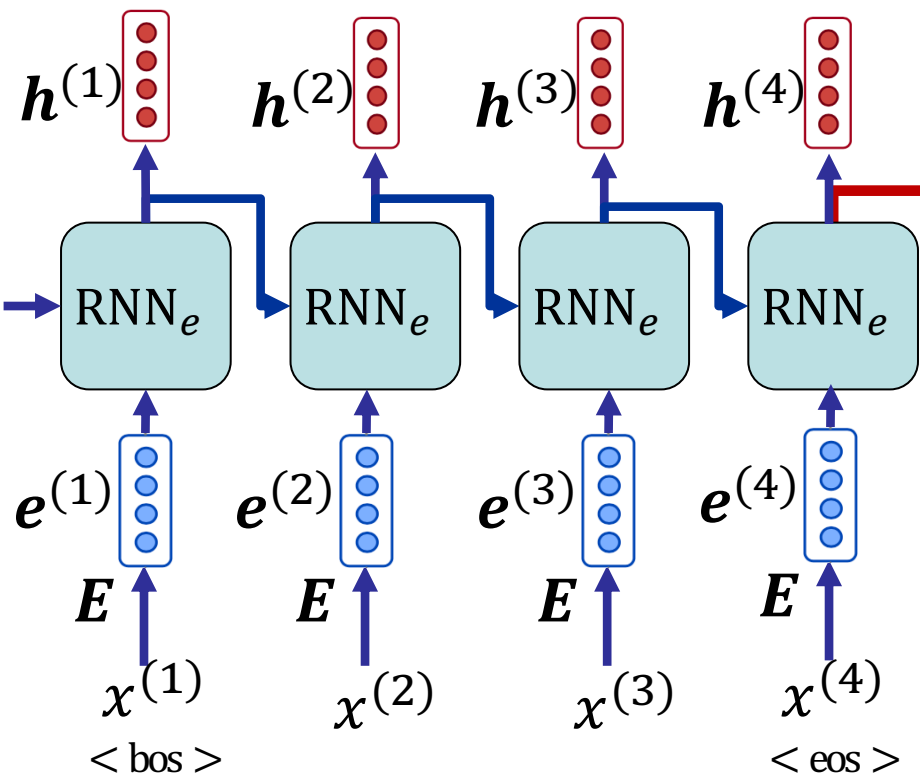
- Like Language Modeling, we ...
- ... **design** a model that predicts the **probabilities of the next words** of the target sequence, one after each other (in **auto-regressive** fashion): $P(y^{(i)} | X, y^{(1)}, \dots, y^{(i-1)})$
- We **train** the model by **maximizing** these probabilities for the correct next words, appearing in training data
- At inference time (or during **decoding**), we use the model to generate new target sequences, that have high **generation probabilities**: $P(Y|X)$

Seq2seq with two RNNs

ENCODER

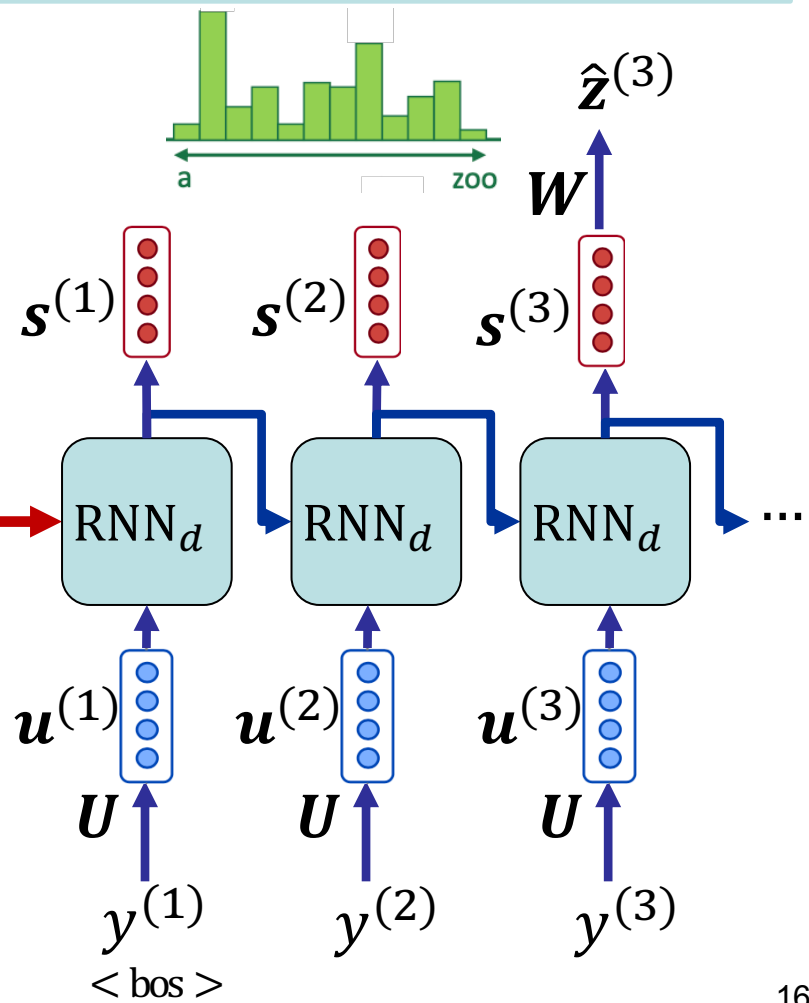
Probability of appearance of the next target word:

$$P(y^{(4)}|X, y^{(1)}, y^{(2)}, y^{(3)}) = \hat{z}_{y^{(4)}}^{(3)}$$



DECODER

$\hat{z}^{(i)}$: predicted probability distribution of the next target word, given the source sequence and first i target words



Seq2seq with two RNNs – formulation

- Two sets of vocabularies
 - \mathbb{V}_e is the set of vocabularies for source sequences
 - \mathbb{V}_d is the set of vocabularies for target sequences

Encoder

- From words to word embeddings:
 - Encoder embeddings of source words (\mathbb{V}_e) \rightarrow **E**
 - Embedding of the source word $x^{(l)}$ (at time step l) $\rightarrow \mathbf{e}^{(l)}$
- Encoder RNN:

$$\mathbf{h}^{(l)} = \text{RNN}_e(\mathbf{h}^{(l-1)}, \mathbf{e}^{(l)})$$

Parameters are shown in red

Seq2seq with two RNNs – formulation

Decoder

- From words to word embeddings:
 - Decoder embeddings of target words (\mathbb{V}_d) at input \rightarrow **U**
 - Embedding of the target word $y^{(t)}$ at time step $t \rightarrow \mathbf{u}^{(t)}$
- Decoder RNN: $\mathbf{s}^{(t)} = \text{RNN}_d(\mathbf{s}^{(t-1)}, \mathbf{u}^{(t)})$
 - where the **initial hidden state** of the decoder RNN is set to the **last hidden state** of the encoder RNN: $\mathbf{s}^{(0)} = \mathbf{h}^{(L)}$
- Decoder output prediction
 - Predicted probability distribution of words at the next time step:

$$\hat{\mathbf{z}}^{(t)} = \text{softmax}(\mathbf{W}\mathbf{s}^{(t)} + \mathbf{b}) \in \mathbb{R}^{|\mathbb{V}_d|}$$

- Probability of the next target word (at time step $t + 1$):

$$P(y^{(t+1)} | X, y^{(1)}, \dots, y^{(t-1)}, y^{(t)}) = \hat{z}_{y^{(t+1)}}^{(t)}$$

Training Seq2seq

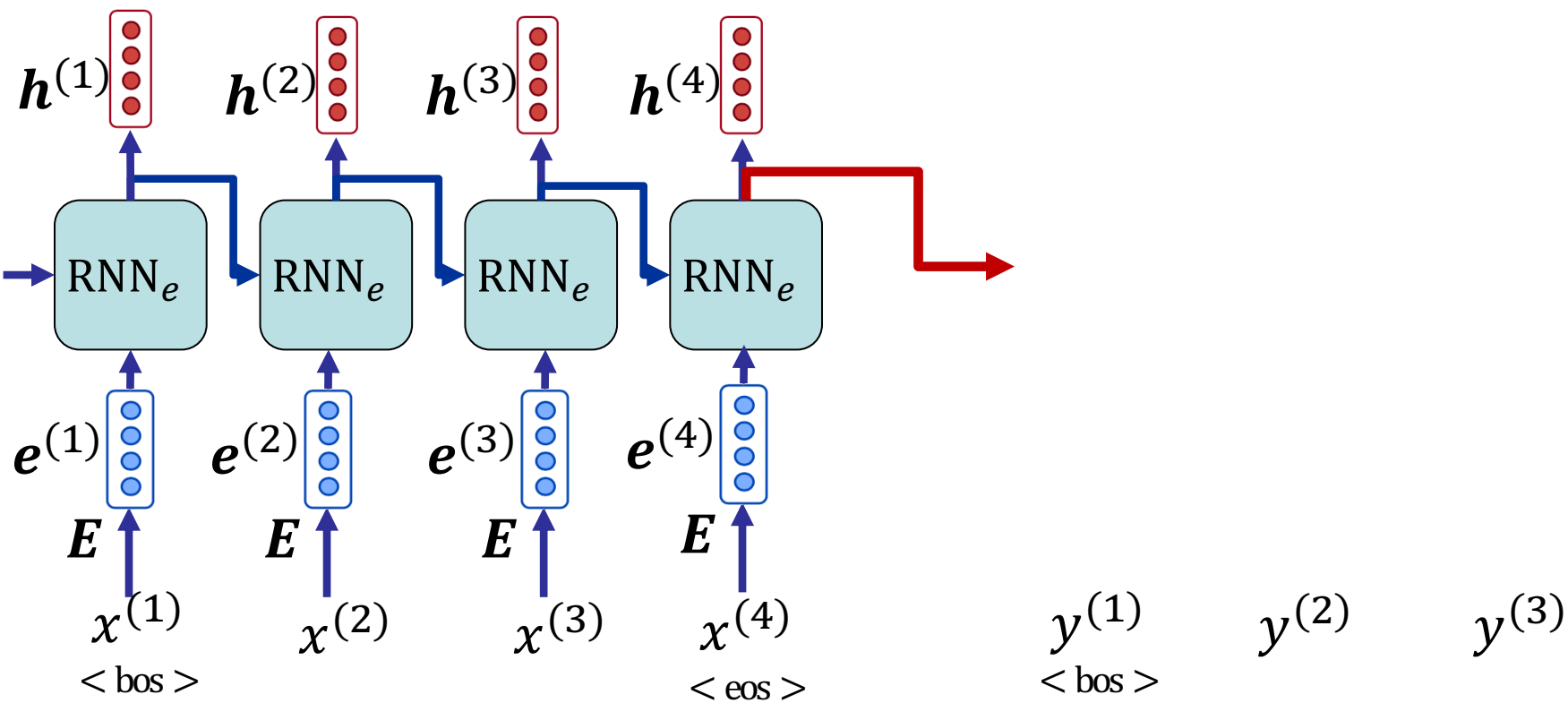
- Training a seq2seq is the same as a Language Model
 - Predict next word, calculate loss, backpropagate, and update parameters
 - Since seq2seq is an end-to-end model, gradient flows from the loss to all parameters, namely both RNNs and all embeddings
- Loss function: Negative Log Likelihood of the **predicted probability** of the correct **next target word** $y^{(t+1)}$

$$\mathcal{L}^{(t)} = -\log \hat{z}_{y^{(t+1)}}^{(t)} = -\log P(y^{(t+1)} | X, y^{(1)}, \dots, y^{(t)})$$

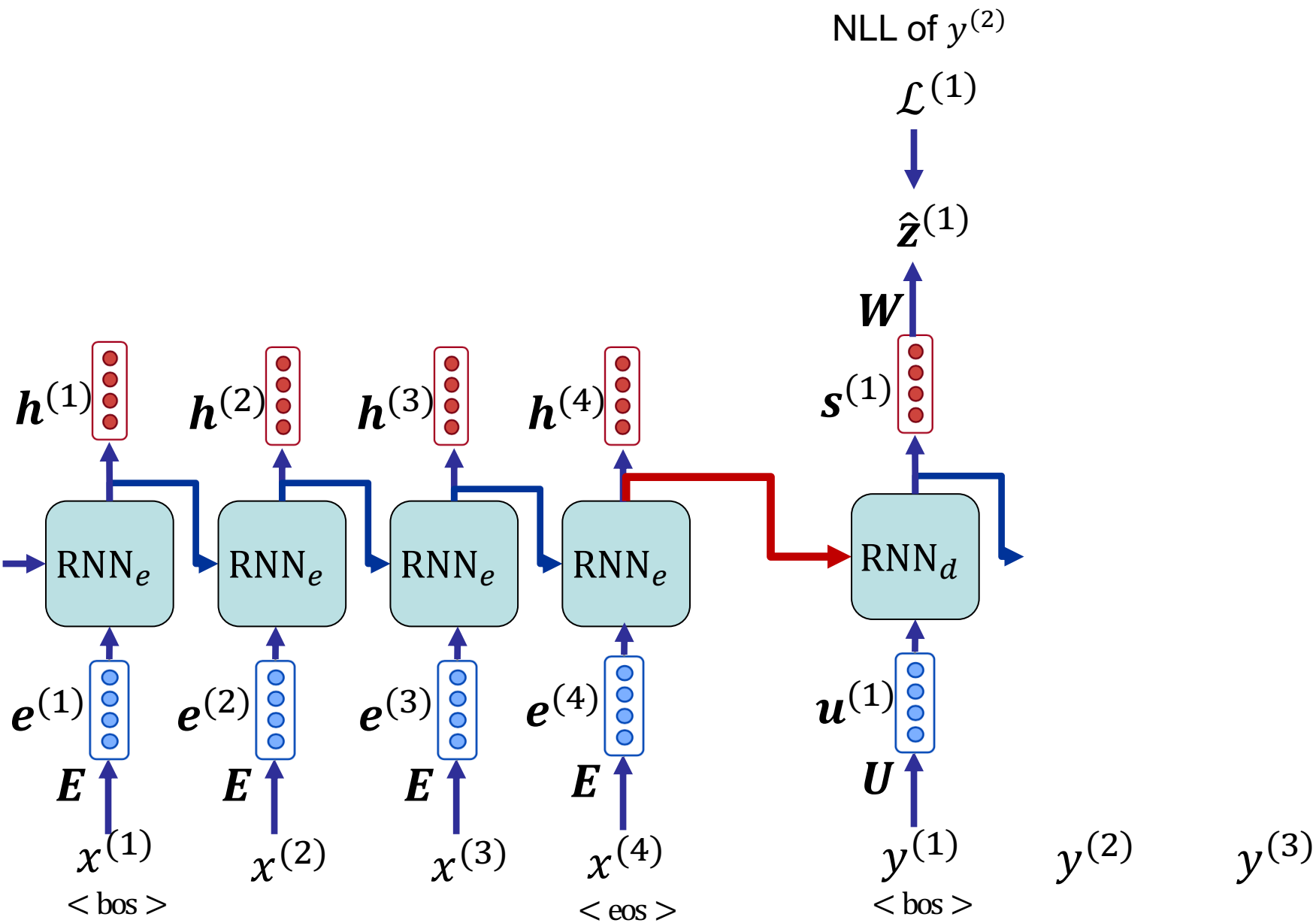
- Overall loss is the average of loss values over the training data in a batch:

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^T \mathcal{L}^{(t)}$$

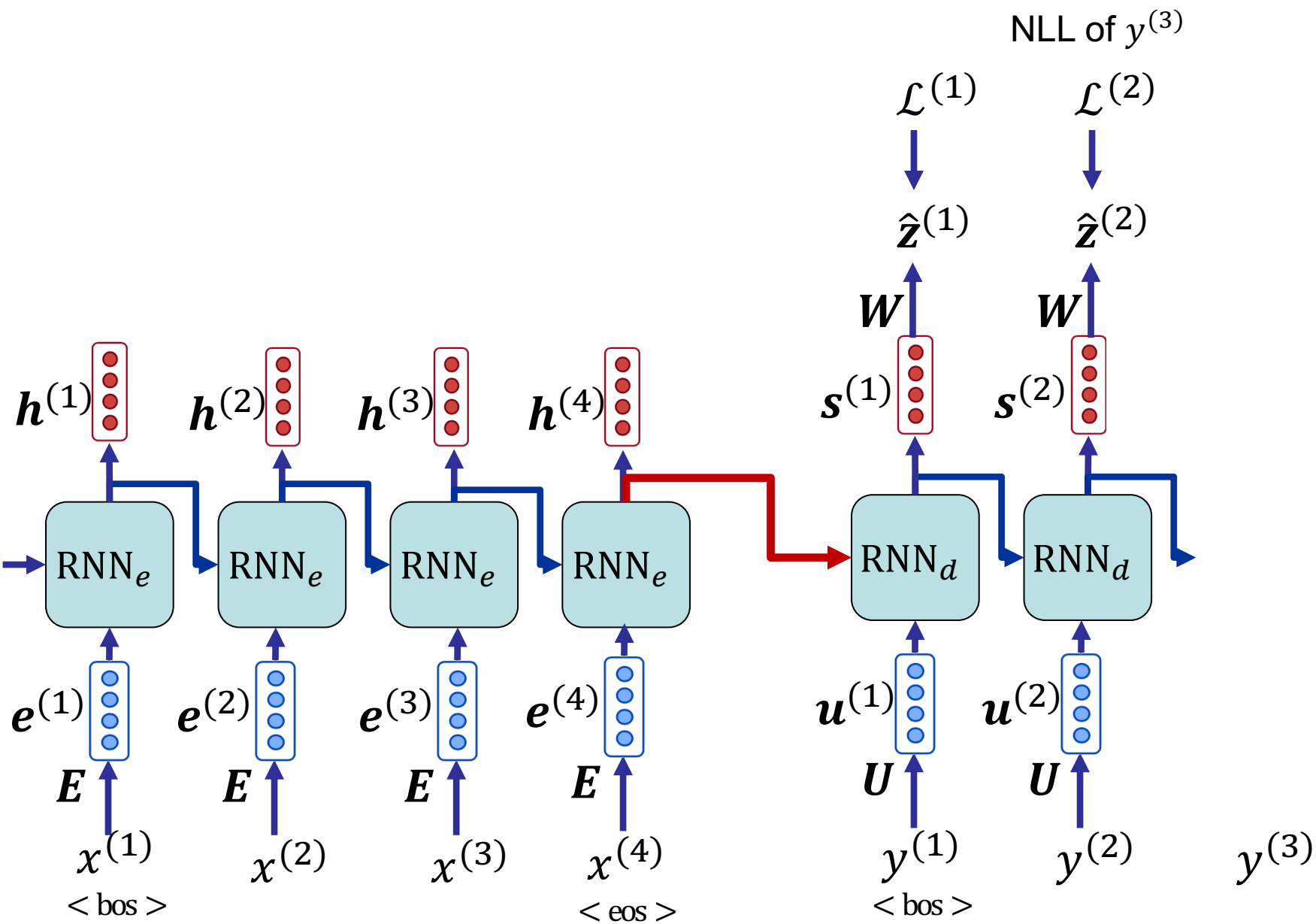
Training Seq2seq



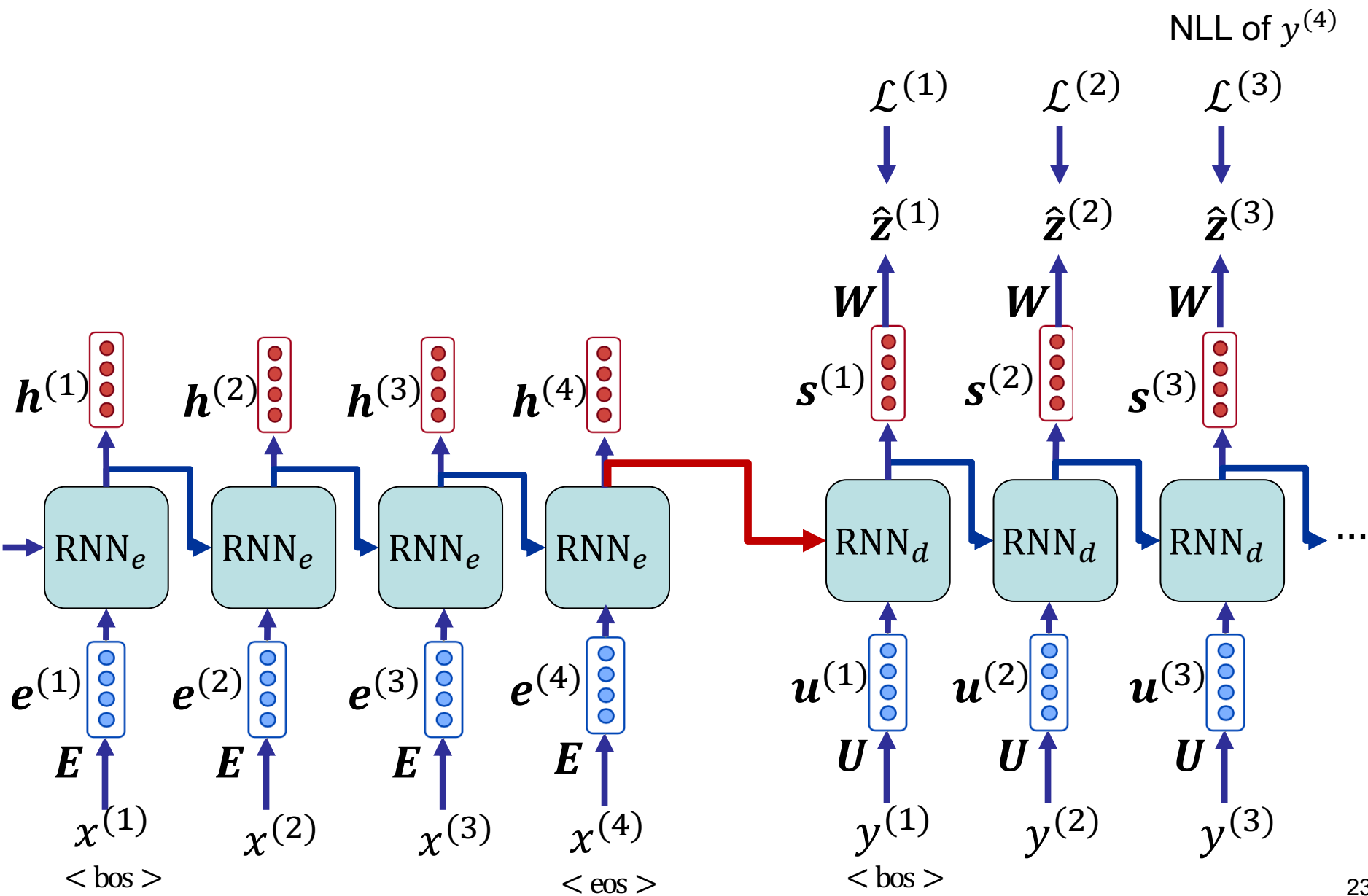
Training Seq2seq



Training Seq2seq



Training Seq2seq



Parameters

- Encoder embeddings $\mathbf{E} \rightarrow |\mathbb{V}_e| \times d_e$
 - Encoder RNN parameters
 - Decoder embeddings $\mathbf{U} \rightarrow |\mathbb{V}_d| \times d_u$
 - Decoder RNN parameters
 - Decoder output projection $\mathbf{W} \rightarrow d_w \times |\mathbb{V}_d|$
-
- bias terms are discarded
 - d_e, d_u, d_w are embedding dimensions
 - RNNs can be an LSTM, GRU, or vanilla (Elman) RNN

Practical points: vocabs & embeddings

- In summarization
 - Encoder and decoder vocabularies are typically the same set, as they are in the **same language**
 - It is different for example in neural machine translation, as there, encoder and decoder vocabularies belong to **two different languages**
 - Encoder and decoder embeddings (E and U) can also share parameters
- Weight tying
 - can be done by **sharing the parameters** of U and W in decoder

Decoding

Recap

- After training, we use the model to **generate** a target sequence given the source sequence (**decoding**). We aim to find the **optimal output sequence** Y^* that maximizes $P(Y|X)$:

$$Y^* = \operatorname{argmax}_Y P(Y|X)$$

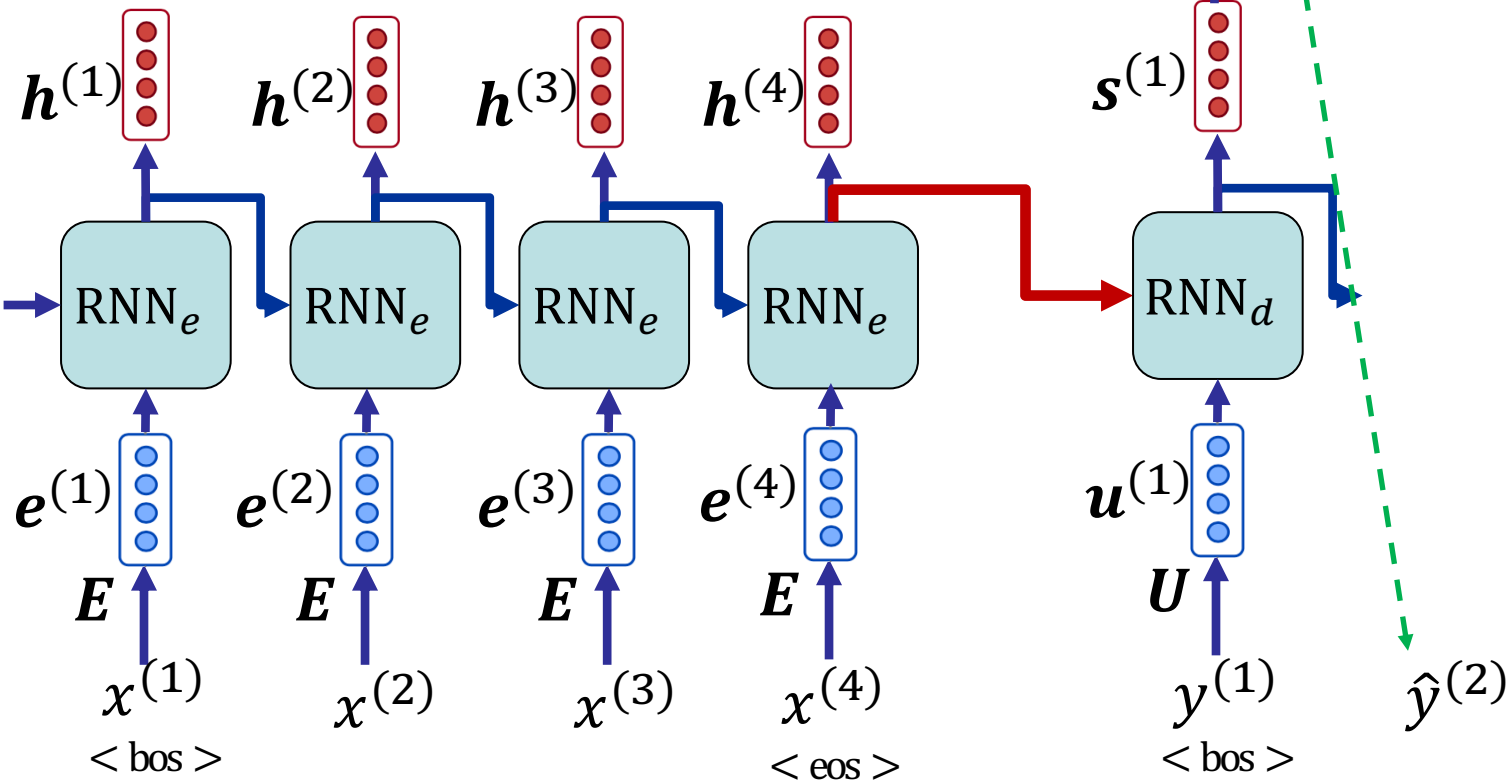
where $P(Y|X)$ for any arbitrary $Y = \{y^{(1)}, y^{(2)}, \dots, y^{(T)}\}$ is:

$$P(Y|X) = \prod_{t=1}^T P(y^{(t)} | X, y^{(1)}, \dots, y^{(t-1)})$$

- ***Question:** among all possible Y sequences, how can we find Y^* ?*

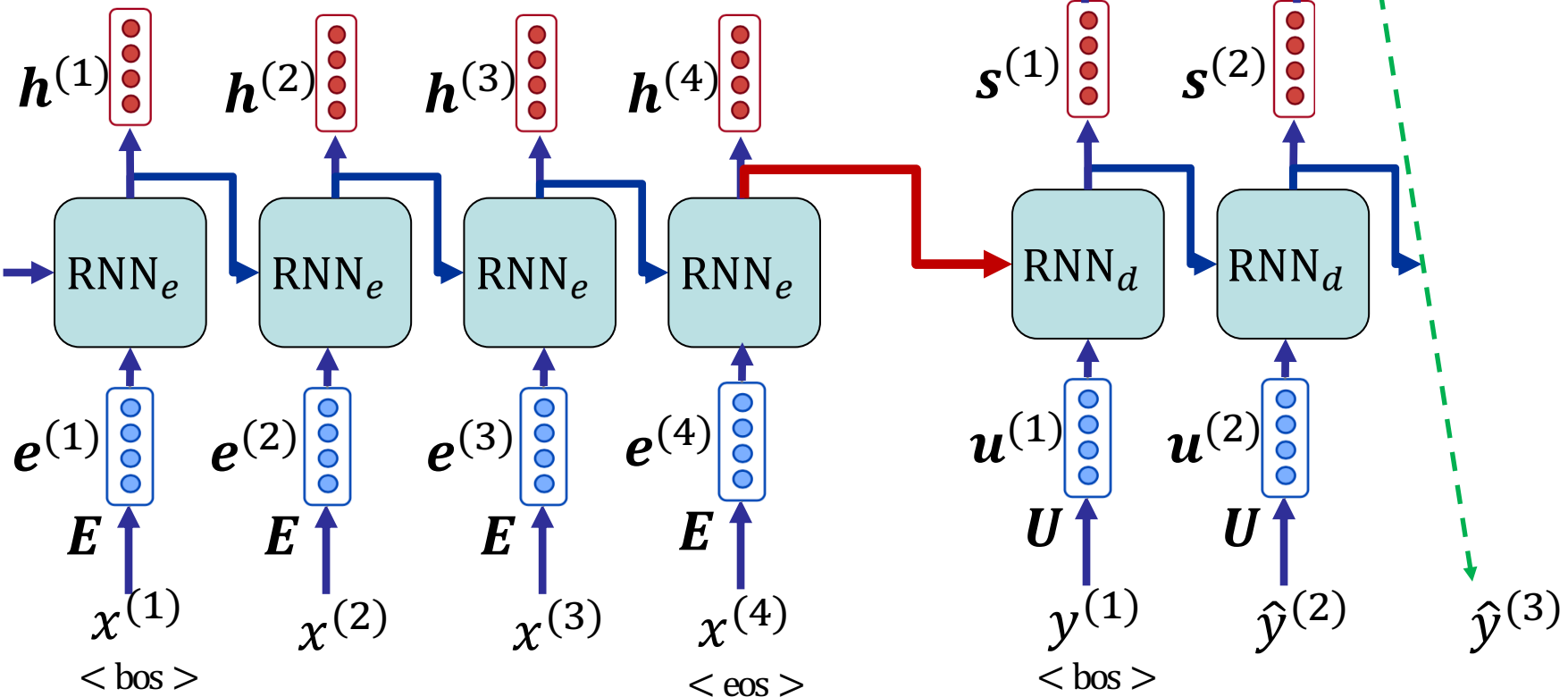
A first approach: Greedy decoding

- In each step, take the **most probable word**
- Use the generated word for the next step, and continue



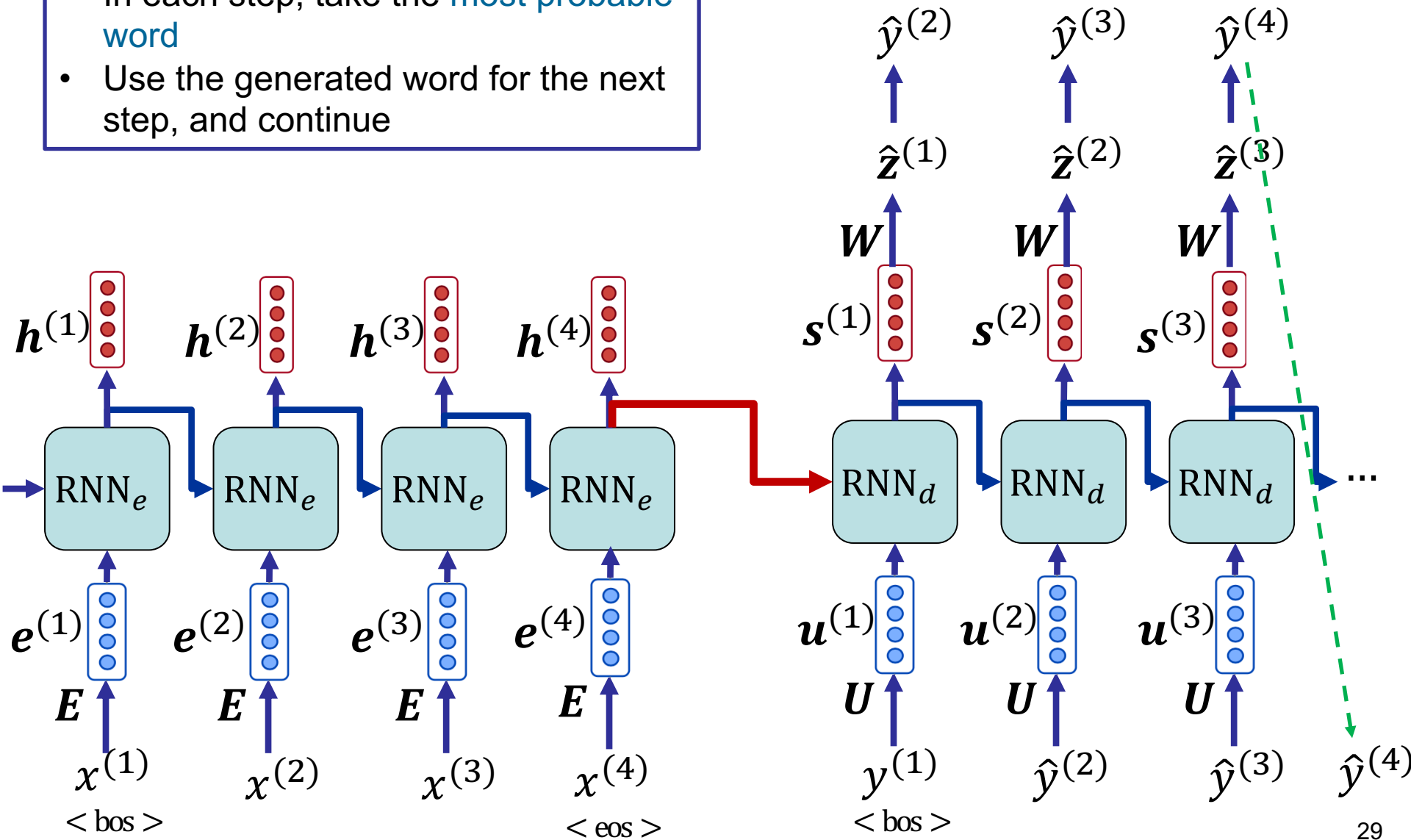
A first approach: Greedy decoding

- In each step, take the **most probable word**
- Use the generated word for the next step, and continue



A first approach: Greedy decoding

- In each step, take the **most probable word**
- Use the generated word for the next step, and continue



Decoding

- Greedy decoding
 - Fast but ...
 - ... decisions are only based on immediate **local knowledge**
 - A non-optimal local decision can get propagated
 - It does not explore other decoding possibilities
- Exhaustive search decoding
 - We *can* compute all possible decodings
 - It means a decoding tree with $|\mathbb{V}_d| \times T$ leaves!
 - Far too **expensive**!
- Beam search decoding
 - A compromise between exploration and exploitation!

Beam search decoding

- Core idea: on each time step of decoding, keep **only k most probable** intermediary sequences (**hypotheses**)
 - k is the beam size (in practice around 5 to 10)

- To do it, beam search calculates of the following **score** for each hypothesis **till time step l** (denoted as $y^{(1...l)}$) :

$$\text{score}(y^{(1...l)}) = \log P(y^{(1...l)} | X) = \sum_{i=1}^l \log P(y^{(i)} | X, y^{(1)}, \dots, y^{(i-1)})$$

- In each decoding step, we only keep k **hypotheses** with the highest scores, and don't continue the rest

Beam search decoding – example

<START>

Calculate prob
dist of next word

Beam search decoding – example

$$-0.7 = \log P_{LM}(he | \langle START \rangle)$$

he

<START>

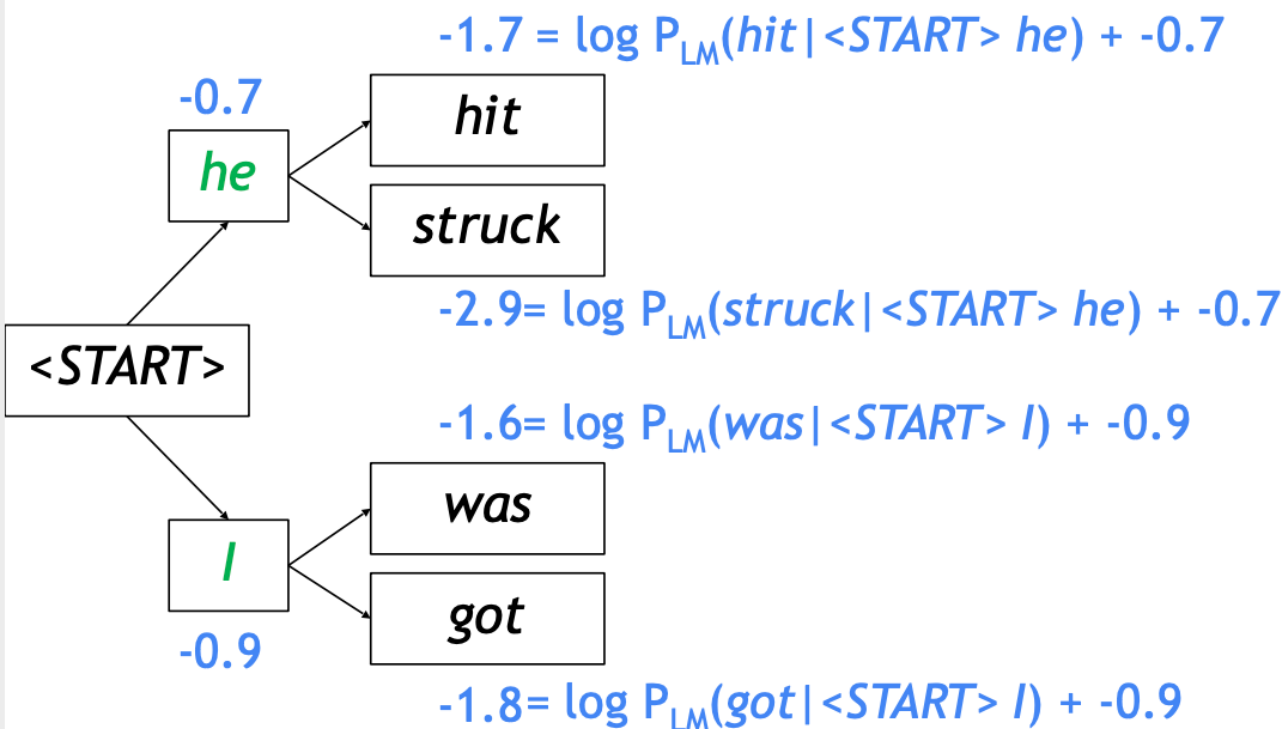
I

$$-0.9 = \log P_{LM}(I | \langle START \rangle)$$

Take top k words
and compute scores

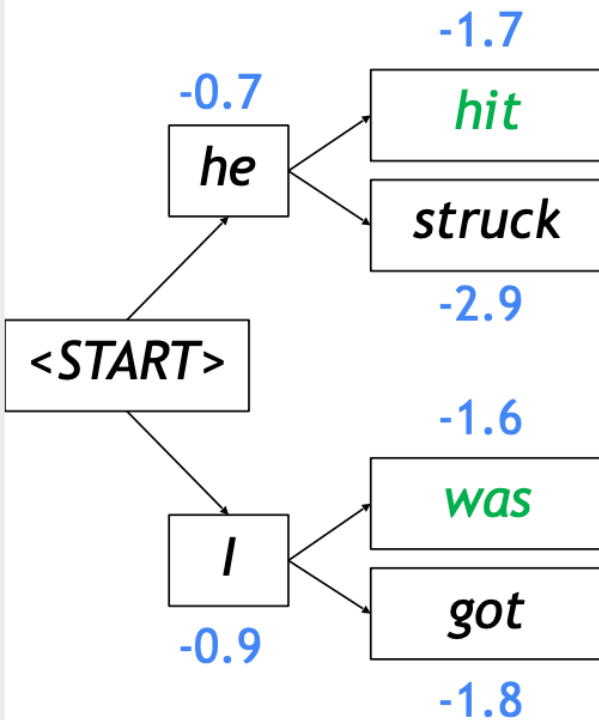


Beam search decoding – example



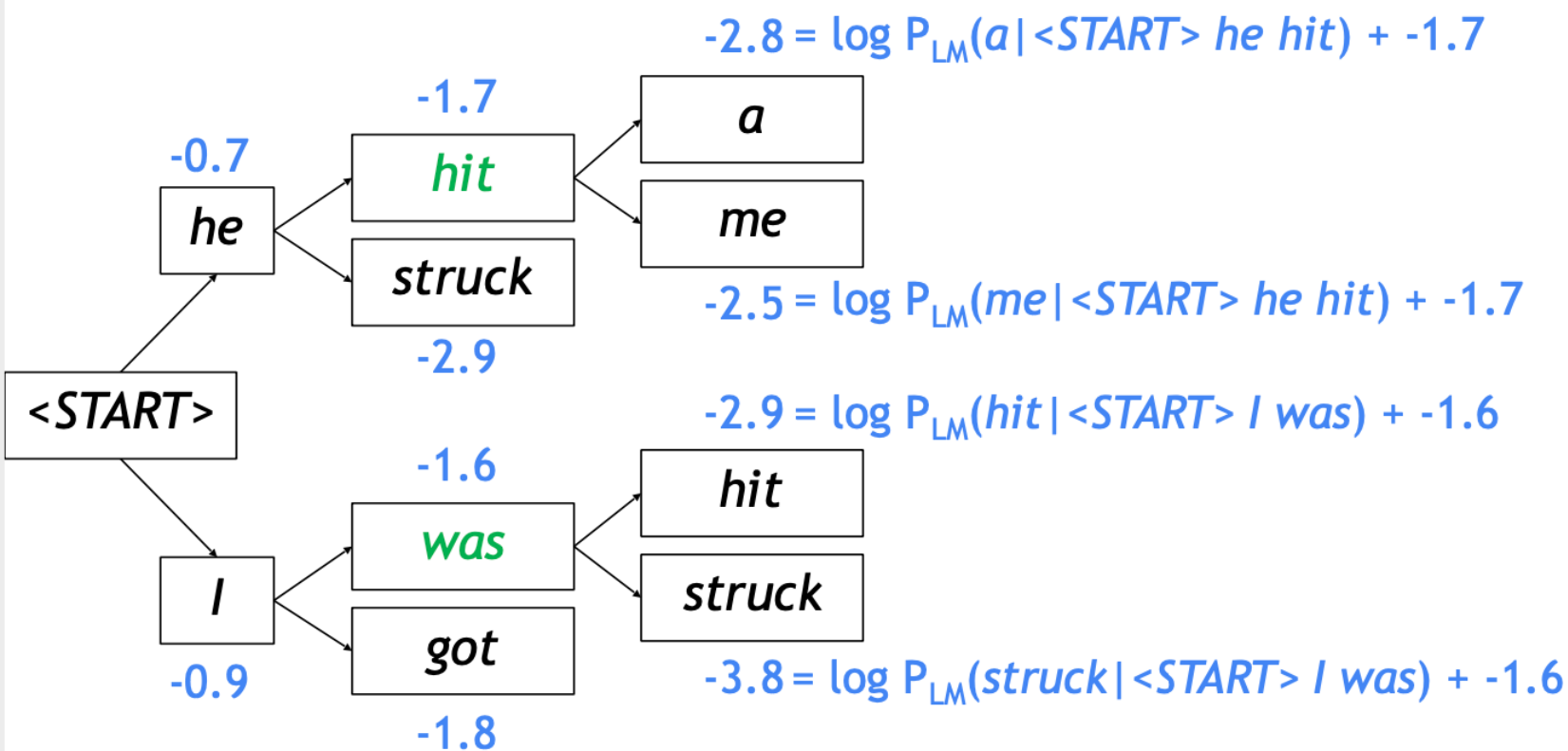
For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding – example



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

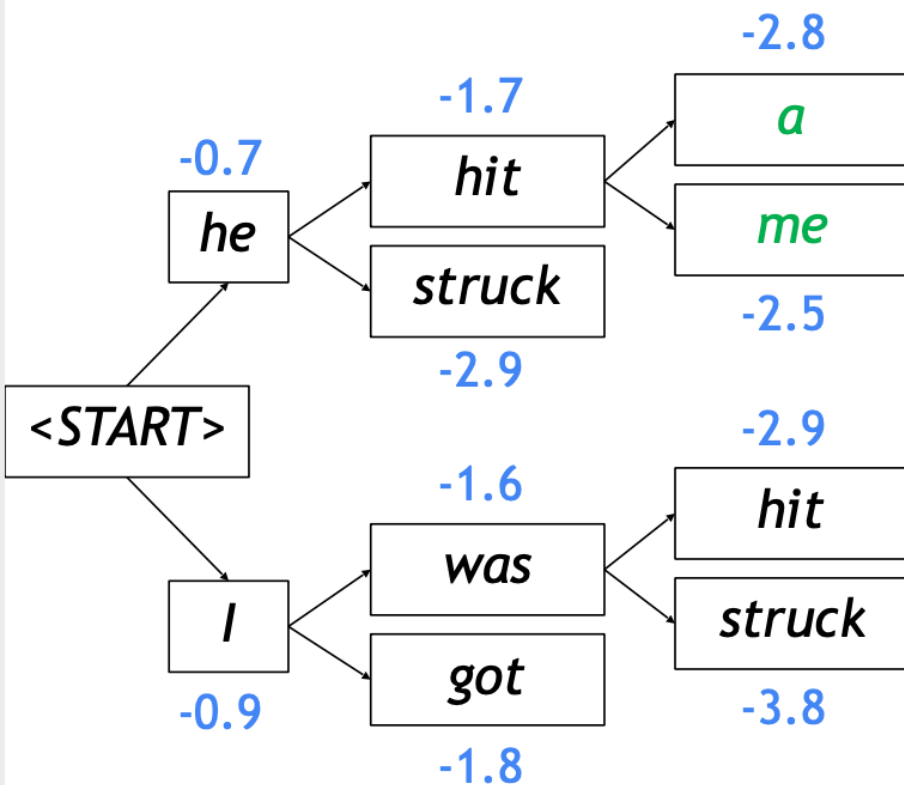
Beam search decoding – example



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

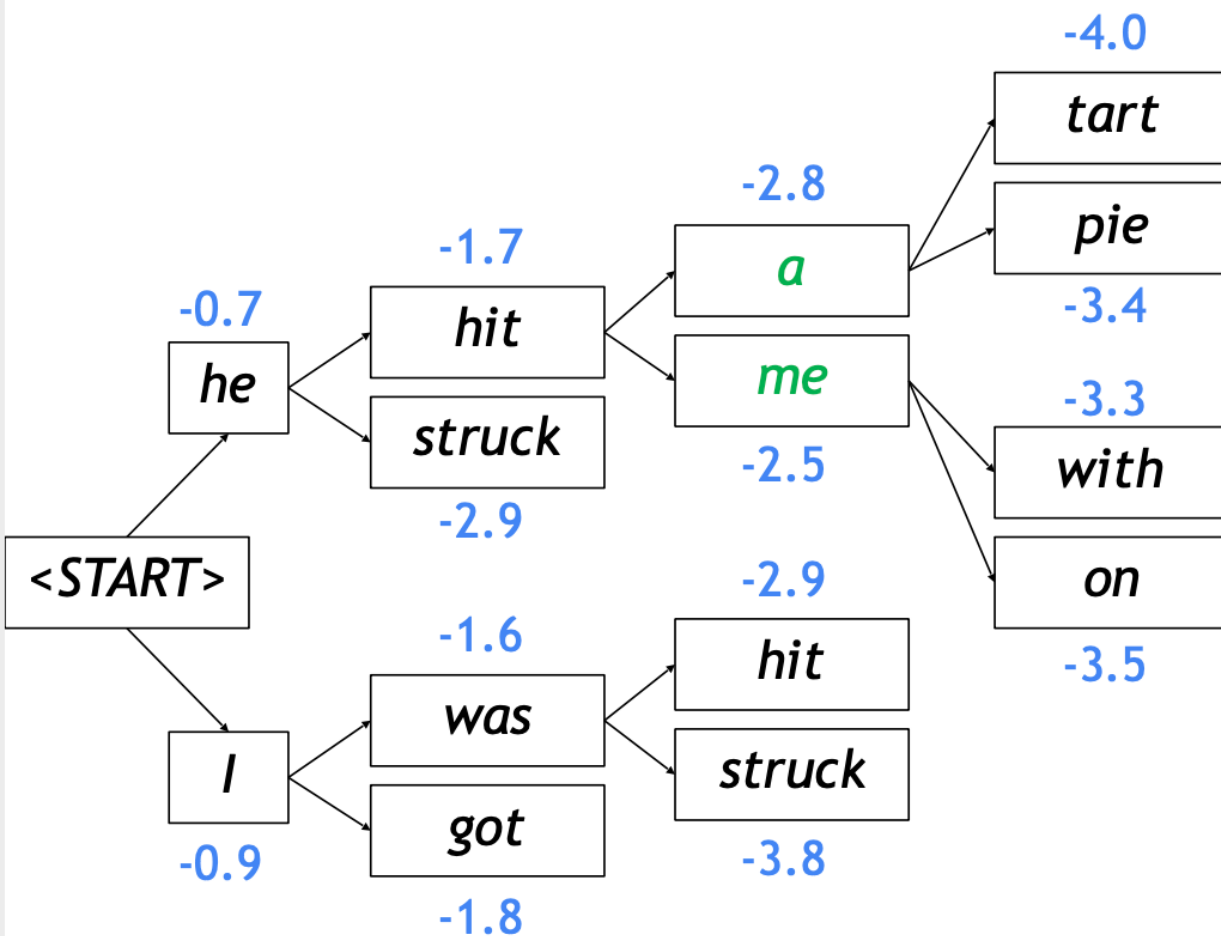


Beam search decoding – example



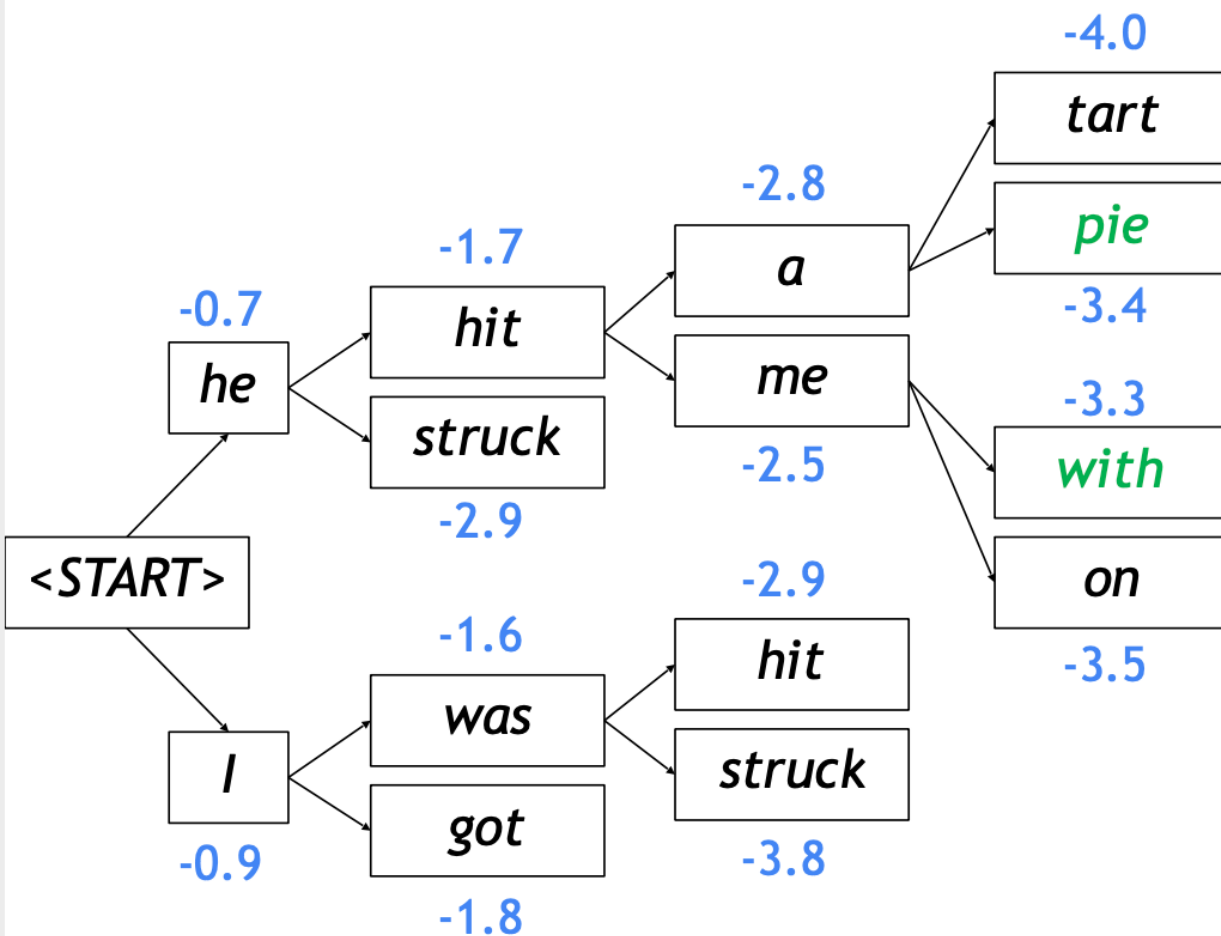
Of these k^2 hypotheses,
just keep k with highest scores

Beam search decoding – example



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

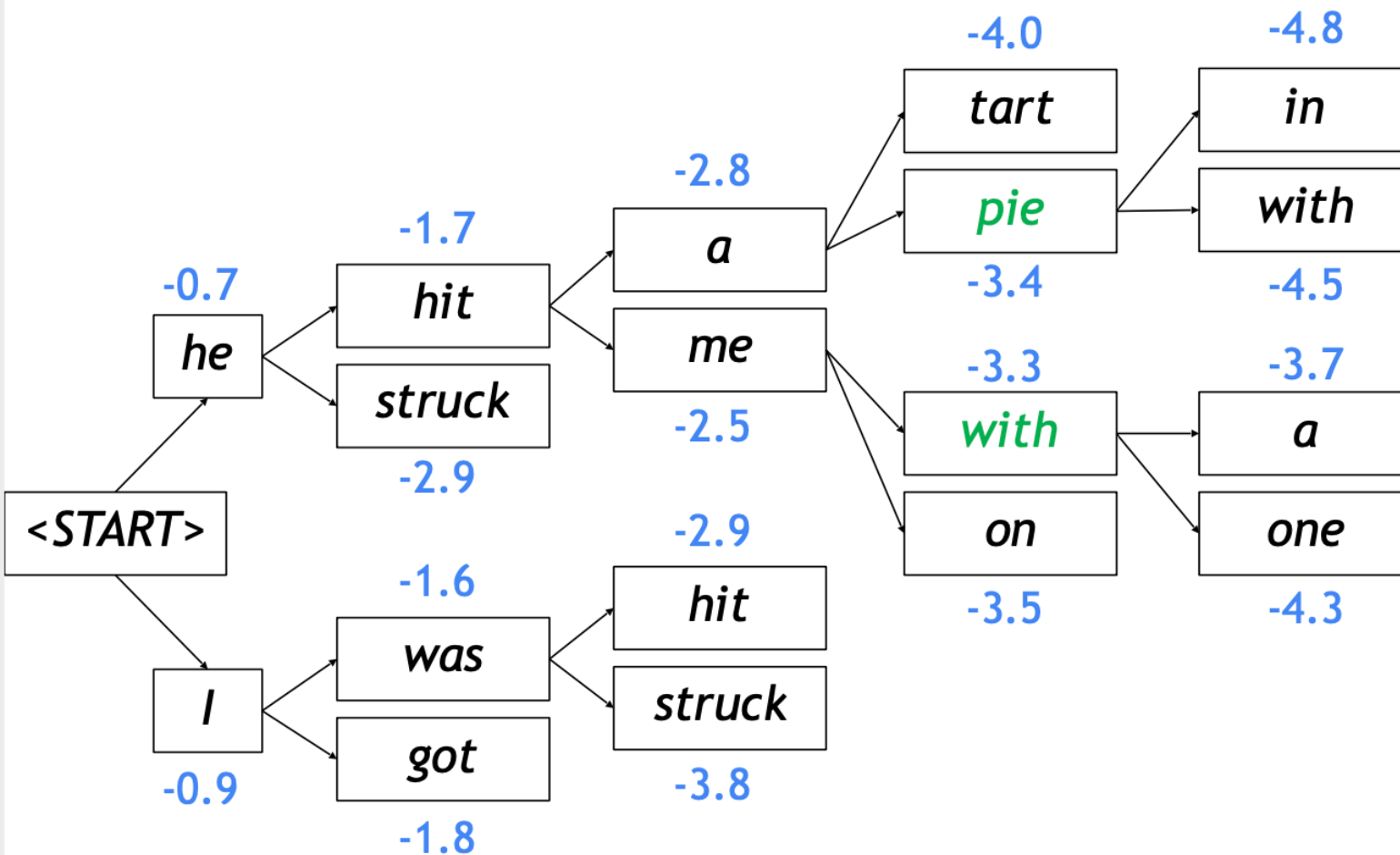
Beam search decoding – example



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

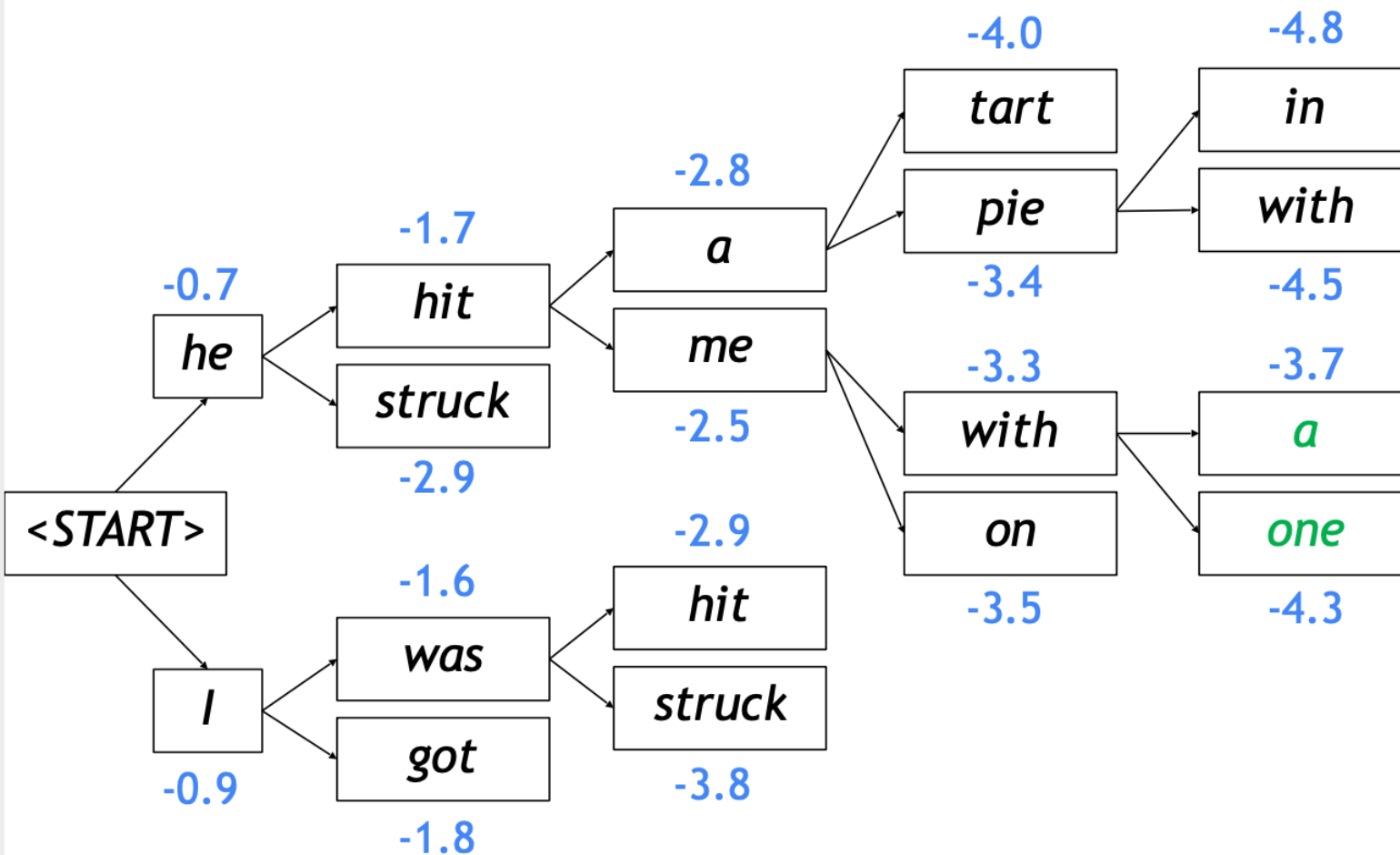


Beam search decoding – example



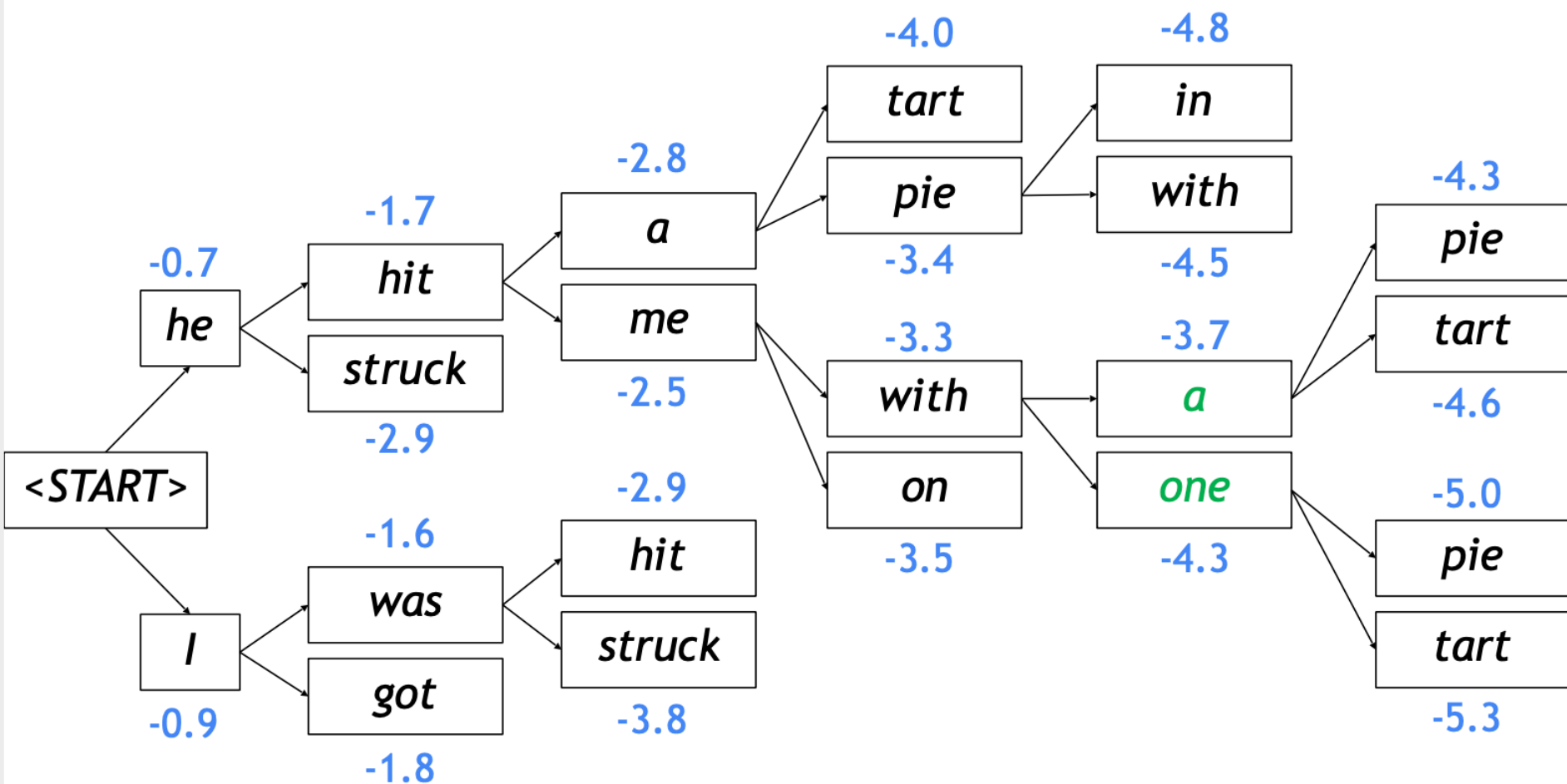
For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding – example



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

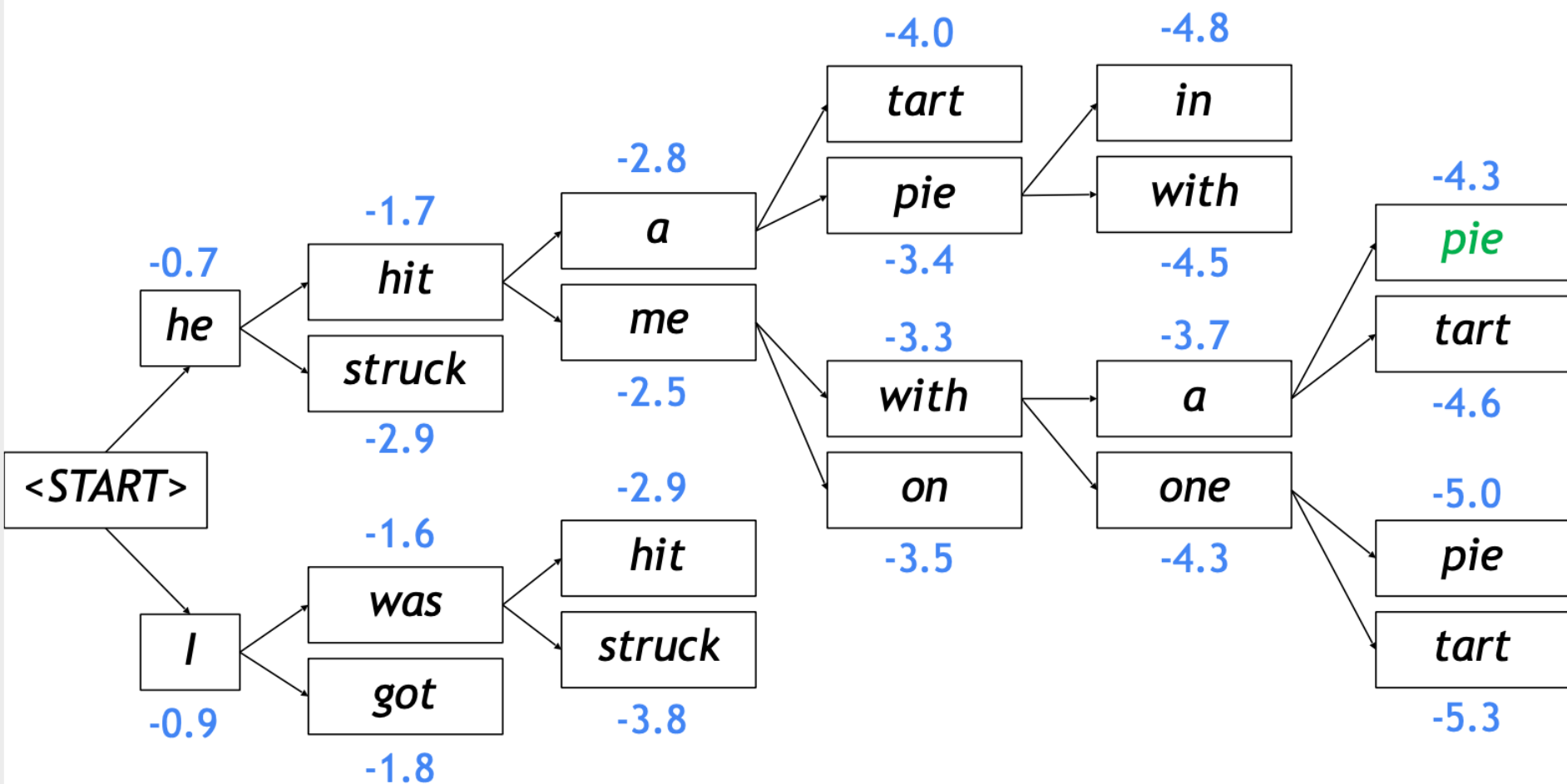
Beam search decoding – example



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

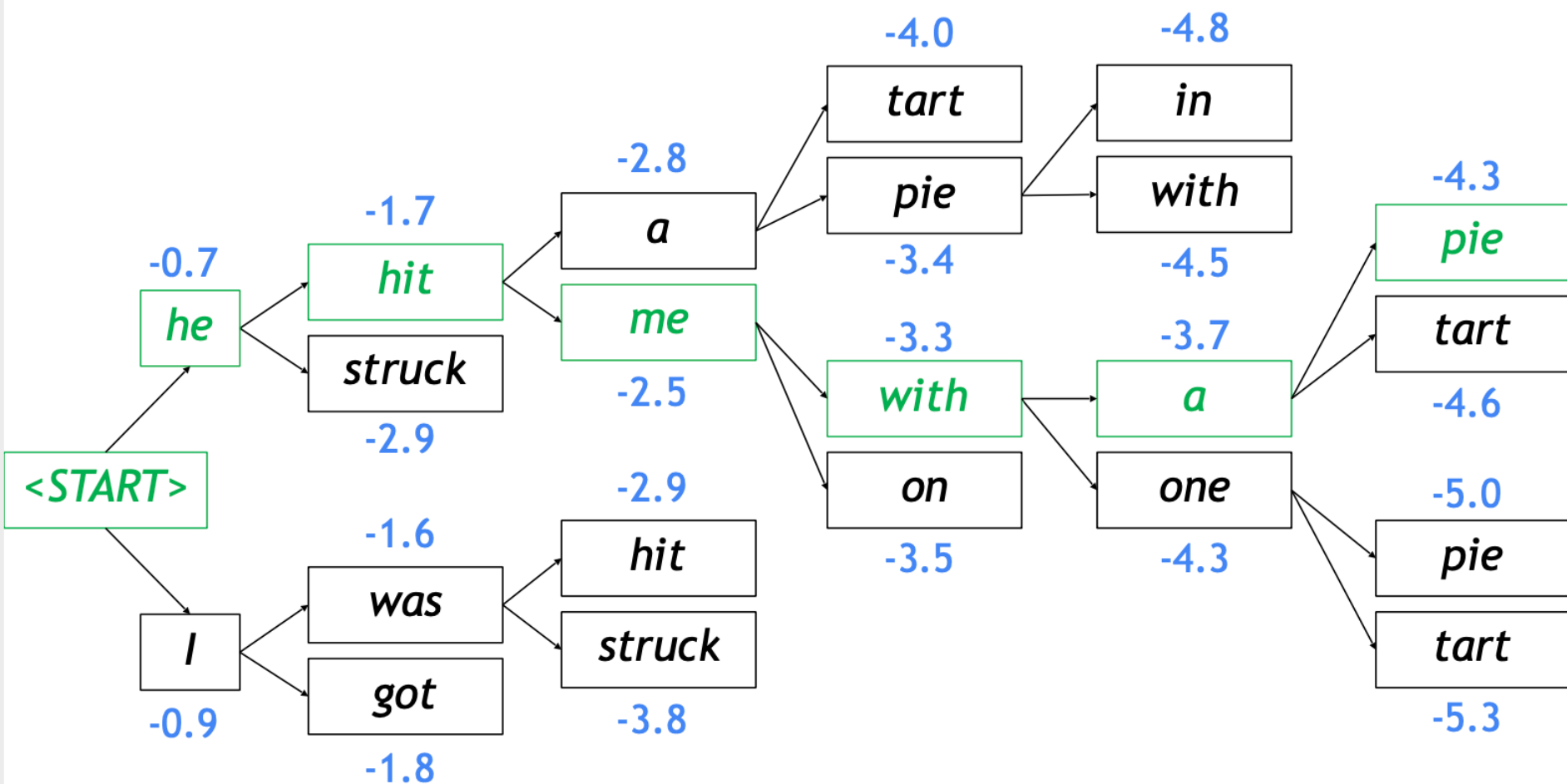


Beam search decoding – example



This is the top-scoring hypothesis!

Beam search decoding – example



Backtrack to obtain the full hypothesis

Beam search decoding – last words!

- Achieving the optimal solution is not guaranteed ...
 - ... but it is much more efficient than exhaustive search decoding
- Stop criteria:
 - Each hypothesis continues till reaching the `<eos>` token
 - Usually beam search decoding continues until:
 - We reach a cutoff timestep T (a hyperparameter), or
 - We have at least n completed hypotheses (another hyperparameter)

Agenda

- Document summarization
- Abstractive summarization with seq2seq
- **Extractive summarization with RNNs**

The content of this section will not be a part of the final evaluation

Extractive Summarization – paper walkthrough

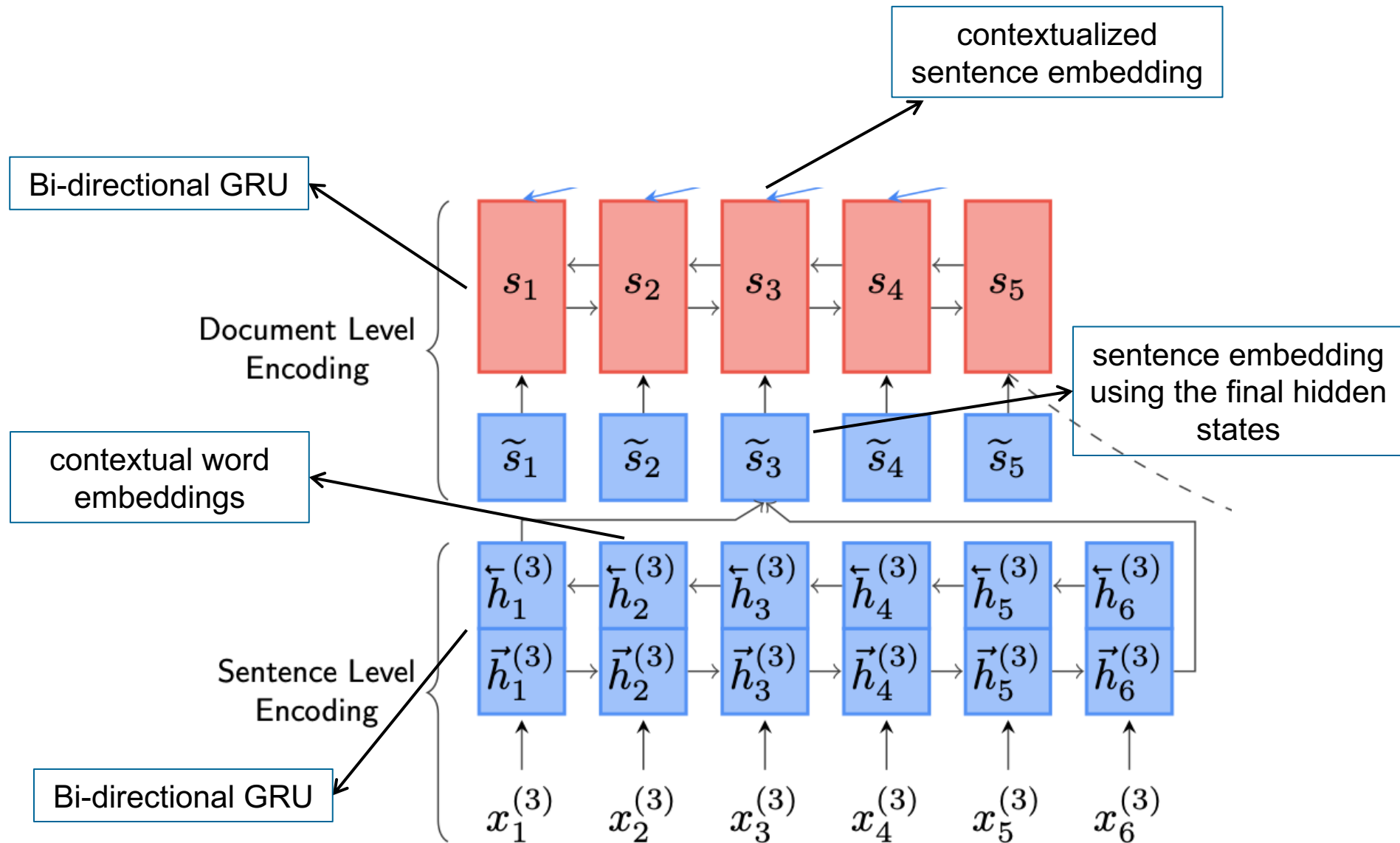
Problem definition

- Document D contains L sentences: $D = \{s_1, s_2, \dots, s_L\}$
- Each sentence s_i contains a set words, each annotated with $x^{(i)}$
- Extractive summarization objective: **select l sentences** of the document such that they provide the “*best*” summary (concise and comprehensive)

Core Ideas – NeuSum model

- At each time step, the model wants to decide which sentence to include in the summary (**sentence selection**)
- To do sentence selection, at each time step, the model assigns scores to sentences that are not included in summary (**sentence scoring**), and selects the one with the highest score
- The sentence scoring is based on the **representation of each sentence**, but also the content of the **previously selected sentences**
 - *Why on previously selected sentences?* Intuitively, if some contents are already included in the summary, the model should avoid selecting the sentences with similar contents

Sentence encoding



Sentence scoring and selection

- Sentence scoring learns a function δ that assigns a score to each sentence s_i at time step t . It uses:
 1. sentence embedding s_i
 2. information of previously selected sentences, embedded in the vector $h_t \rightarrow$ **current state of summary**

$$\delta(s_i) = \text{function}(h_t, s_i)$$

$$\delta(s_i) = \mathbf{w}_s \tanh(\mathbf{h}_t \mathbf{W}_q + \mathbf{s}_i \mathbf{W}_d + \mathbf{b}_i)$$

- $\delta(s_i)$ is calculated for the sentences that are not yet included in the summary
- The sentence with highest δ is added to summary

Sentence scoring

- How is **current state of summary** \mathbf{h}_t calculated?
 - Using another GRU
- A GRU model outputs the new state of the summary using the **previous state of summary** and the **last selected sentence**

$$\mathbf{h}_t = \text{GRU}(\mathbf{h}_{t-1}, \mathbf{s}_{t-1})$$

You can find such more details by looking into the paper:

- \mathbf{h}_0 is created based on the document embedding
- The model is optimized using the Kullback-Leibler divergence between the distribution of output scores and the distribution of ROUGE-2 F1 gains of sentences

