STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

CS584 Natural Language Processing

Machine Translation, Seq2seq models

Ping Wang
Department of Computer Science
Stevens Institute of Technology

Overview

- Introduce a new task: Machine Translation

↓ is a major use-case of

- Introduce a new neural architecture: Seq2seq models

↓ is improved by

- Introduce a new neural technique: Attention

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: have a good day (English)
  ↓
- y: bonne journée (French)

Machine Translation

- Translation is easy for (bilingual) people

- Process:
  - Read the text in English
  - Understand it
  - Write it down in French

- Hard tasks for computers
  - The human process is invisible, intangible

History of Machine Translation

- 1950s: Early Machine Translation (Rule-based)
  - Russian → English (motivated by the Cold War!)
  - Systems were mostly rule-based, using a bilingual dictionary to map Russian words to their English counterparts

- 1990s-2010s: Statistical Machine Translation (SMT, Statistics-based)
  - Core idea: Learn a probabilistic model from data
  - The best systems were extremely complex

- 2014- : Neural Machine Translation (NMT)

Statistical Machine Translation

- Core idea: Learn a probabilistic model from data
- French f → English e
- We want to find best English sentence e, given French sentence f.
- Use Bayes Rule to break down to two components

Mathematically:

$$P(e|f) = \frac{P(e)\,P(f|e)}{P(f)}$$

$$T(f) = \hat{e} = \text{argmax}_e\ P(e|f) = \text{argmax}_e\ P(e)\,P(f|e)$$

Statistical Machine Translation

Components

$$T(f) = \hat{e} = \underset{e}{\operatorname{argmax}} P(e)P(f|e)$$

- Language Model
  - Takes care of fluency in the target language
  - Models how to write good English
  - Data: corpora in the target language
- Translation Model
  - Lexical correspondence between languages
  - Models how words and phrases should be translated.
  - Data: aligned corpora in source and target languages
- argmax
  - Search for the best translation

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 specially-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!

NMT: the first big success story of NLP Deep Learning

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

- **2014**: First seq2seq paper published
- **2016**: Google Translate switches from SMT to NMT – and by 2018 everyone has

- This is amazing!
  - **SMT** systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by a small group of engineers in a few months

https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture06-fancy-rnn.pdf

Neural Machine Translation

- A way to do Machine Translation with a single end-to-end neural network.
- The neural network architecture is called sequence-to-sequence (seq2seq) and it involves two RNNs.
- Both the input and output are sequences.
- **End-to-end learning**: in the context of AI and ML, it is a technique where a single, comprehensive model is used to solve a complex task without the need for intermediate steps or handcrafted features.

Seq2Seq Basics

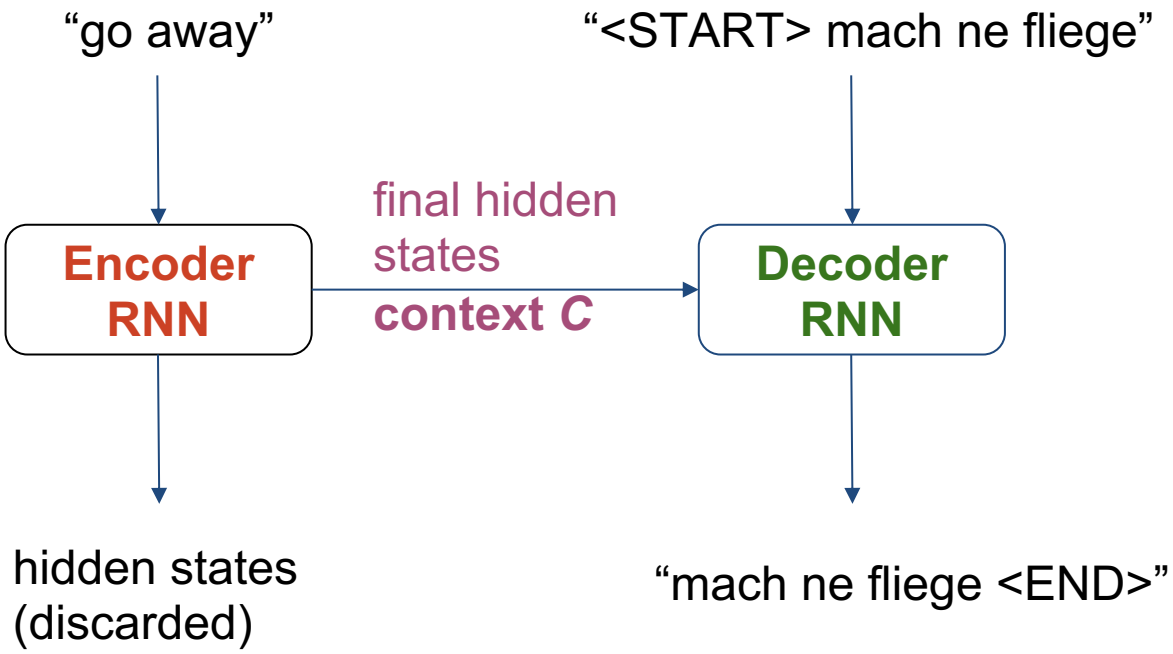
- A paradigm published in 2014 for English-French translation (Sutskever et al. 2014).

- At a high level, a seq2seq model is an end-to-end model made up of two RNNs:
  - an *encoder*, which takes the model's input sequence as input and encodes it into a fixed-size "context vector"
  - a *decoder*, which uses the context vector from above as a "seed" from which to generate an output sequence.

- Seq2Seq models are often referred to as "encoder-decoder" models.

Seq2Seq

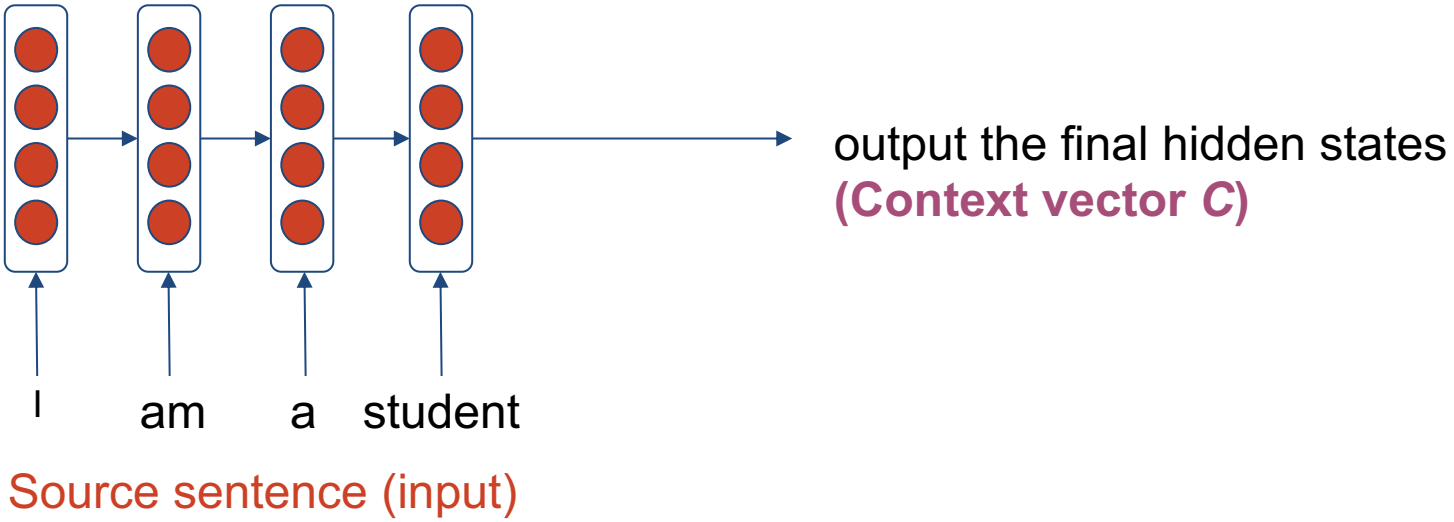
- 2 parts: encoder and decoder.



- The decoder is a text generator.
- Difference from the simple text generator: the initial states are determined by the encoder.

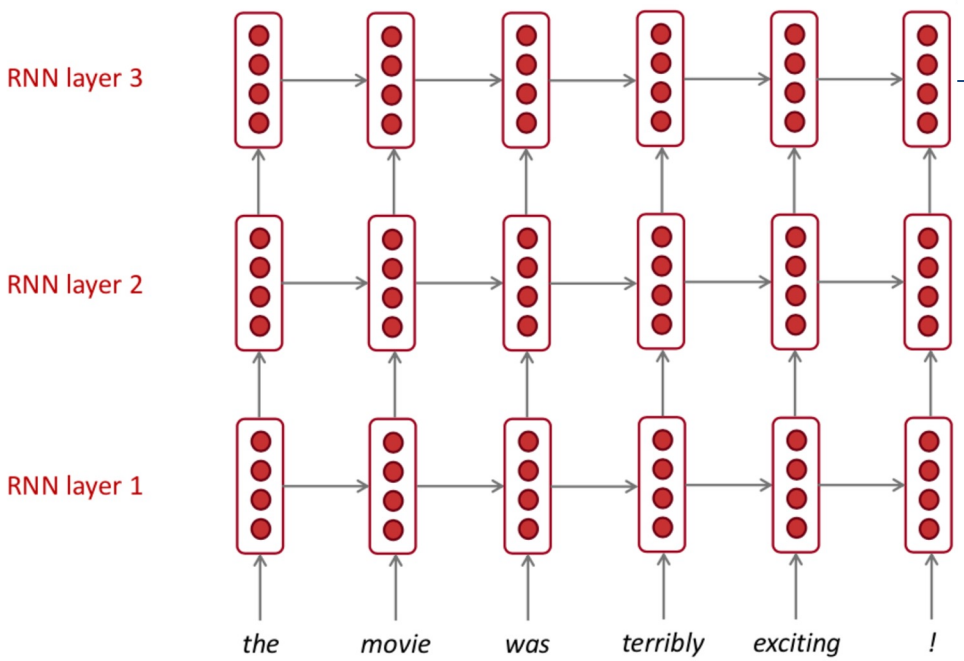
Seq2Seq - Encoder

- **Goal**: read the input sequence to our Seq2Seq model and generate a fixed-dimensional context vector *C* for the sequence.
- The encoder will use a recurrent neural network cell (usually an LSTM or GRU) to read the input tokens one at a time. The final hidden state of the cell will then become *C*.



Seq2Seq - Encoder

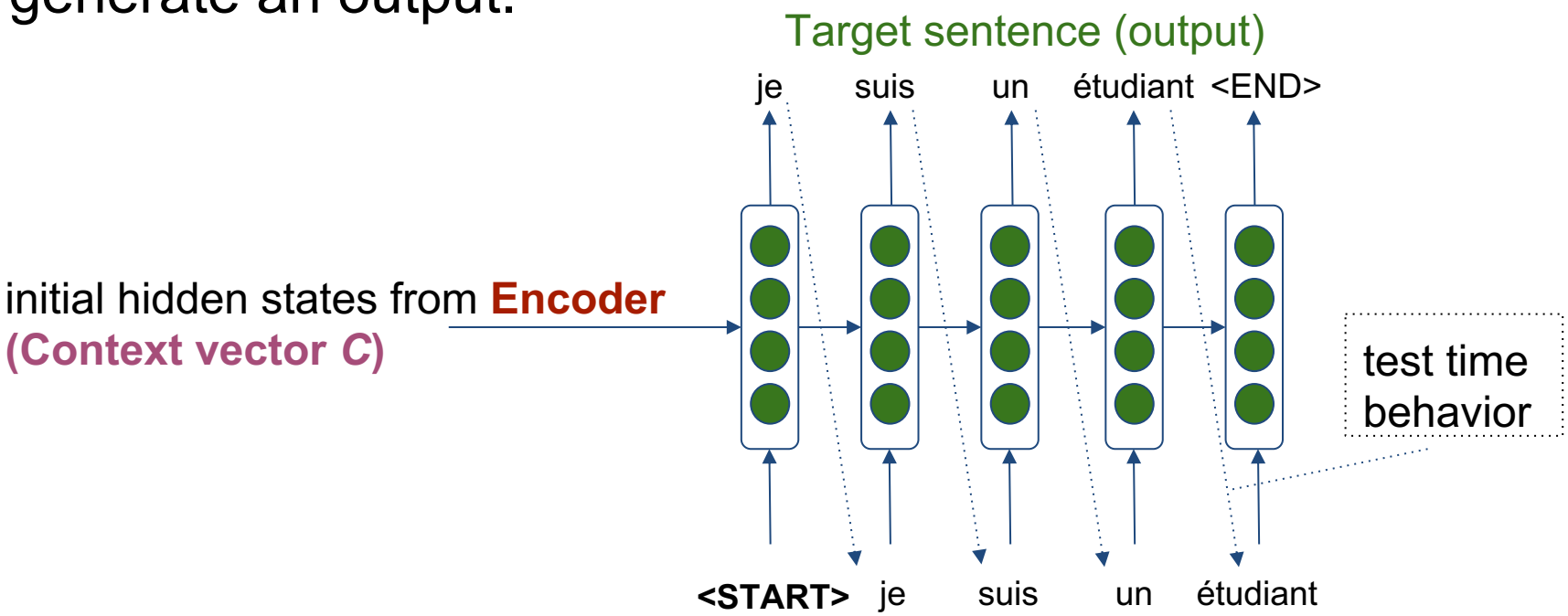
- However, it's so difficult to compress an arbitrary-length sequence into a single fixed-size vector (especially for difficult tasks like translation)
- The encoder will usually consist of stacked RNNs



output the final layer's hidden states **(Context vector *C*)**

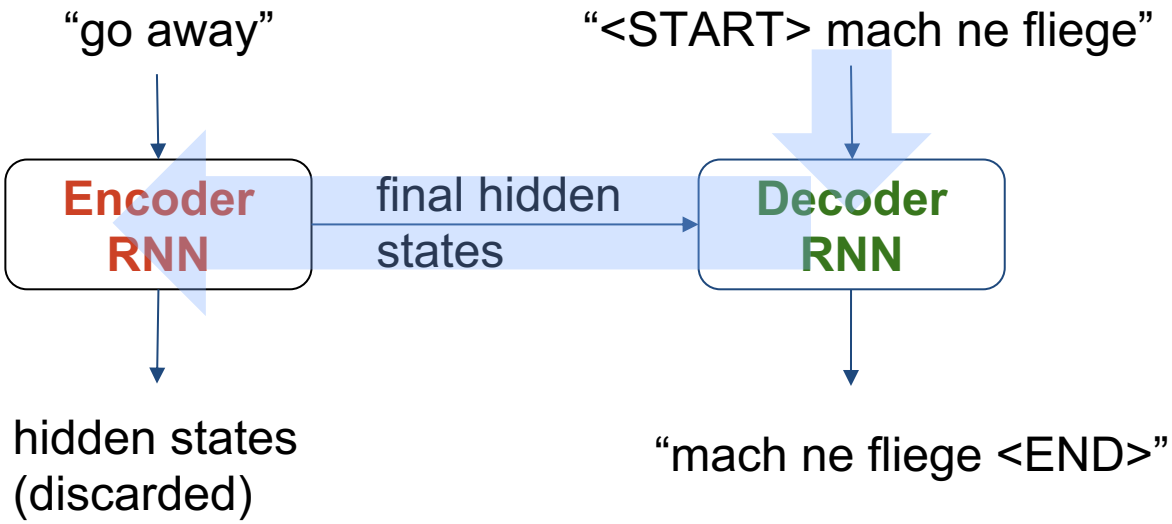
Seq2Seq - Decoder

- Initialize the hidden state with the context vector
- A stack of several recurrent units (one or more layers) where each predicts an output y_t at a time step t.
- The decoder will literally use the context of the input to generate an output.

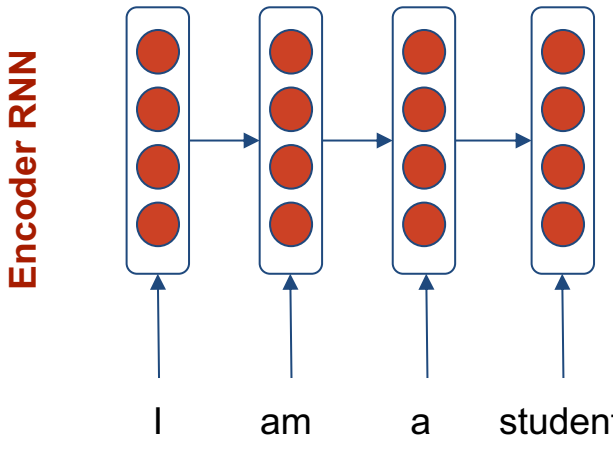


Seq2Seq - Training

- Define a **loss**, the cross entropy on the prediction sequence
- Minimize it with a gradient descent algorithm and **back-propagation**.
- Both the encoder and decoder are trained at the same time, so that they both learn the same context vector representation.



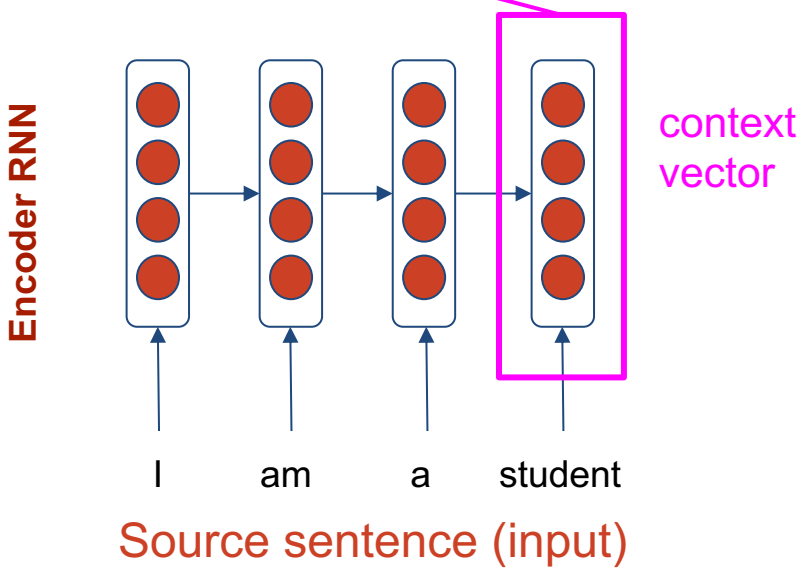
Neural Machine Translation - Seq2Seq



Encoder RNN produces an encoding of the source sentence.

Neural Machine Translation - Seq2Seq

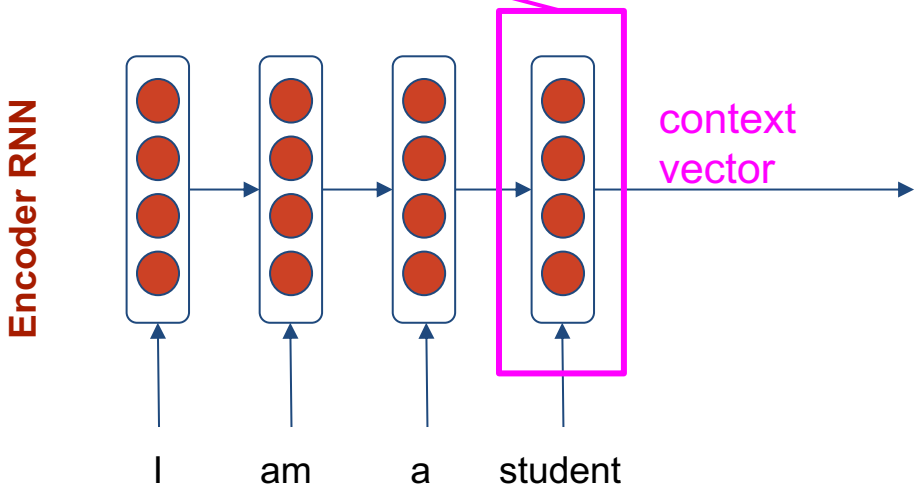
Encoding of the source sentence. Provides initial hidden state for Decoder RNN.



Encoder RNN produces an encoding of the source sentence.

Neural Machine Translation - Seq2Seq

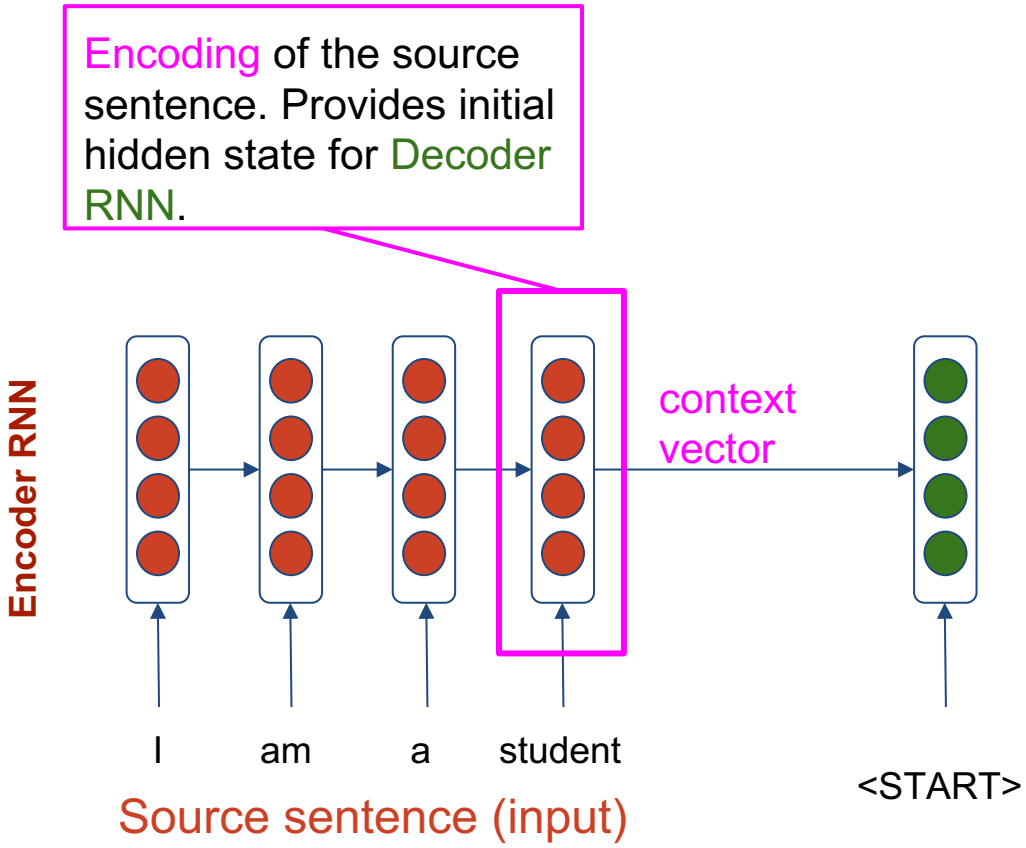
Encoding of the source sentence. Provides initial hidden state for Decoder RNN.



Encoder RNN produces an encoding of the source sentence.

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

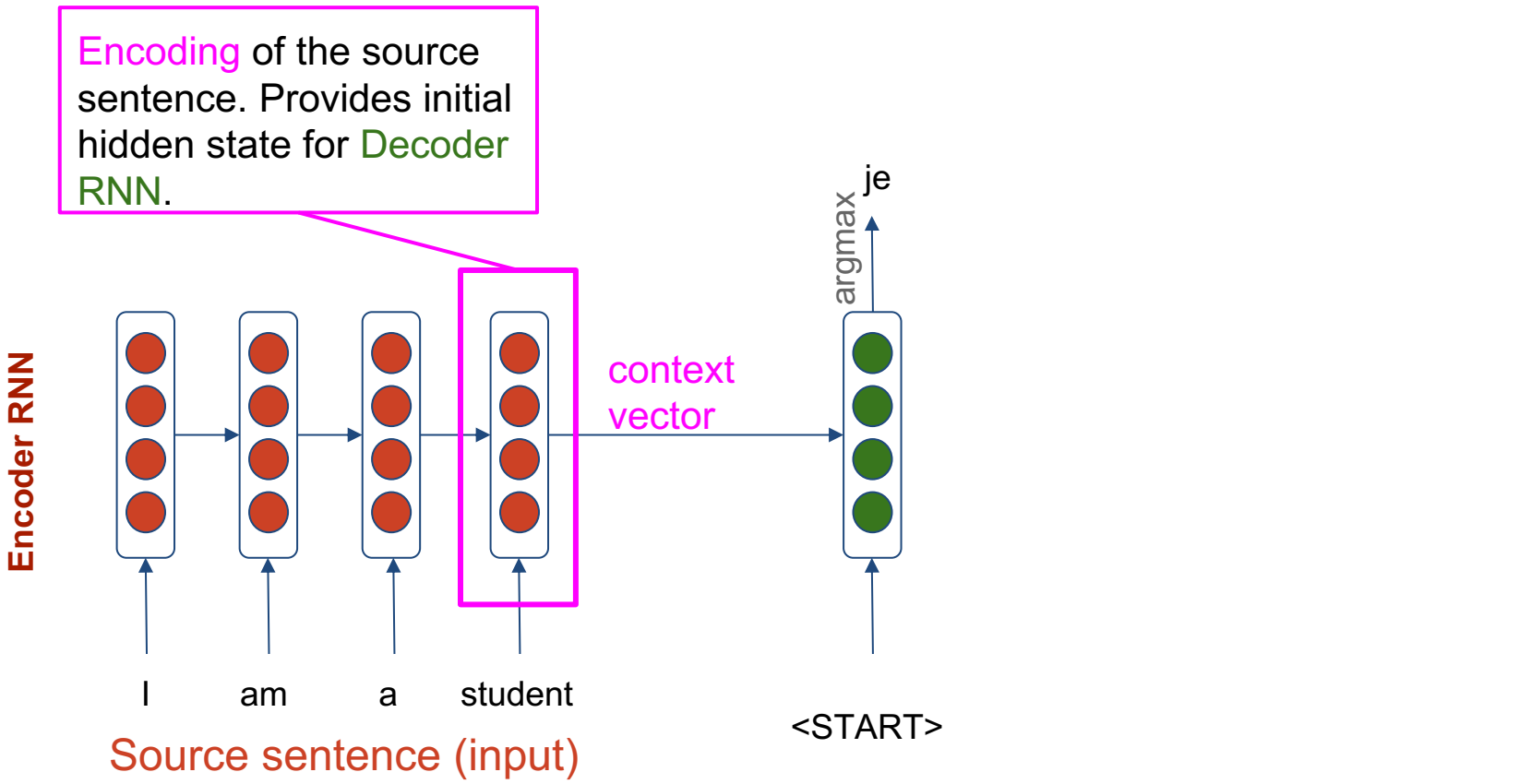
Neural Machine Translation - Seq2Seq



Encoder RNN produces an encoding of the source sentence.

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

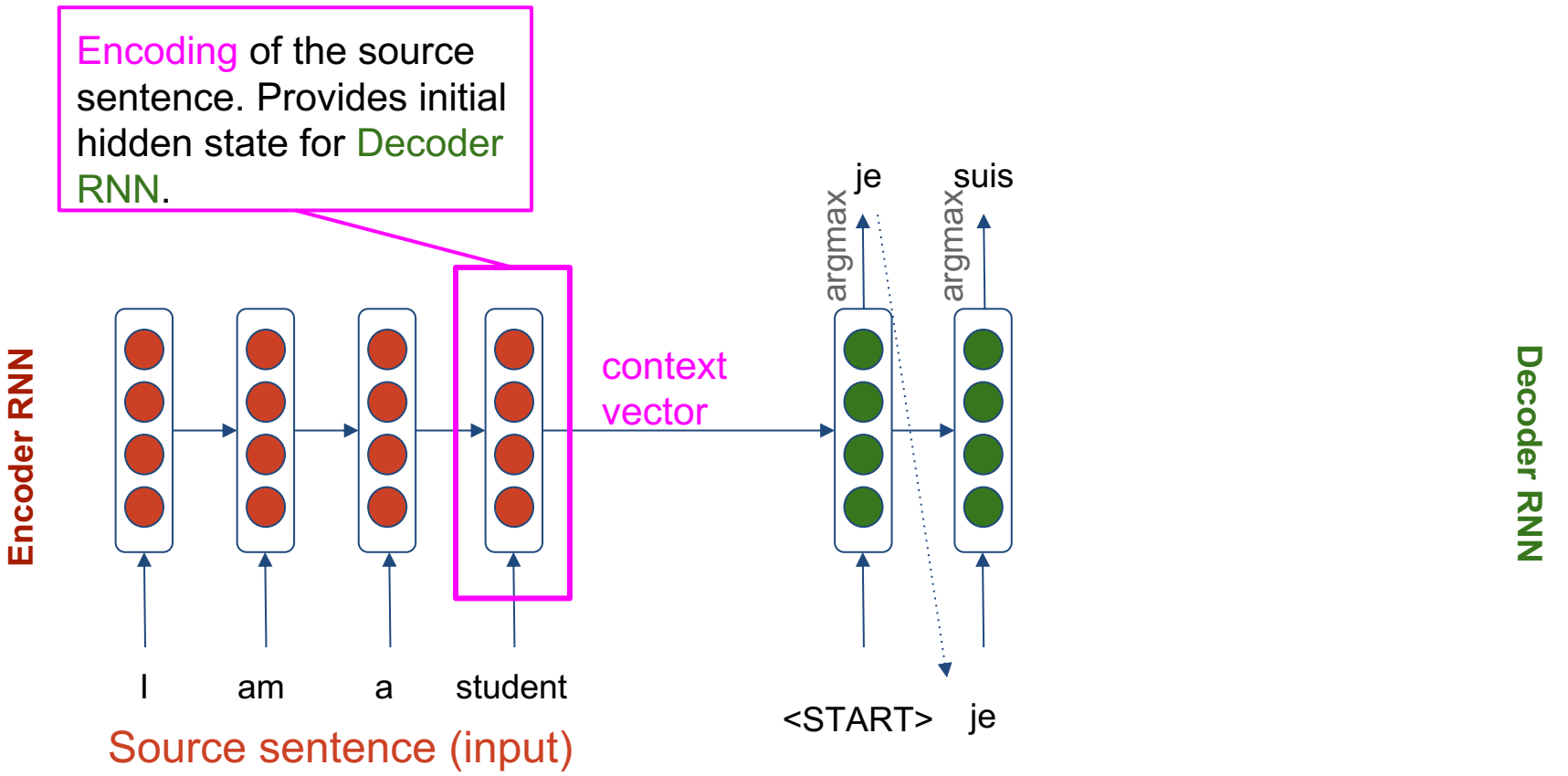
Neural Machine Translation - Seq2Seq



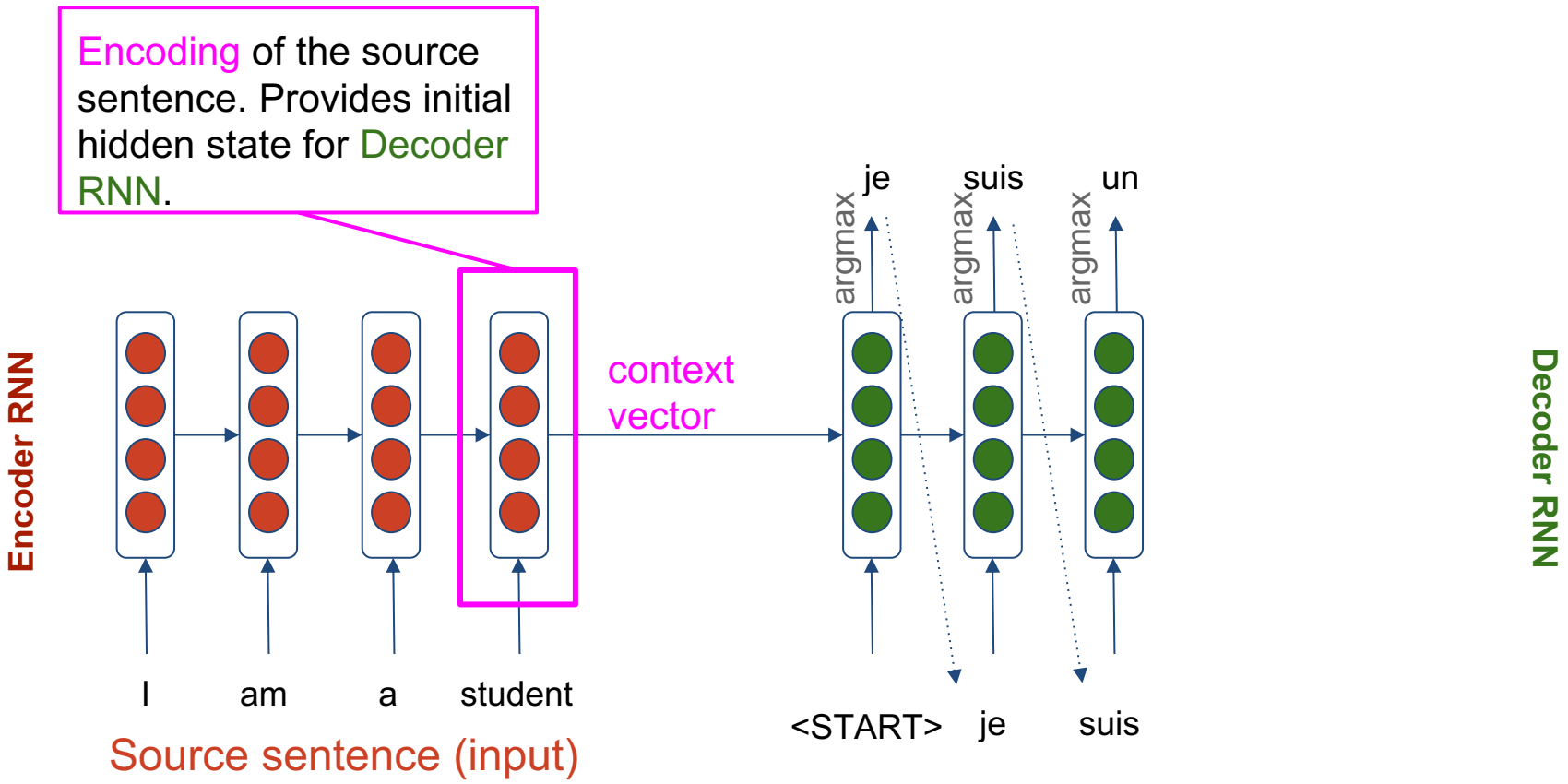
Encoder RNN produces an encoding of the source sentence.

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

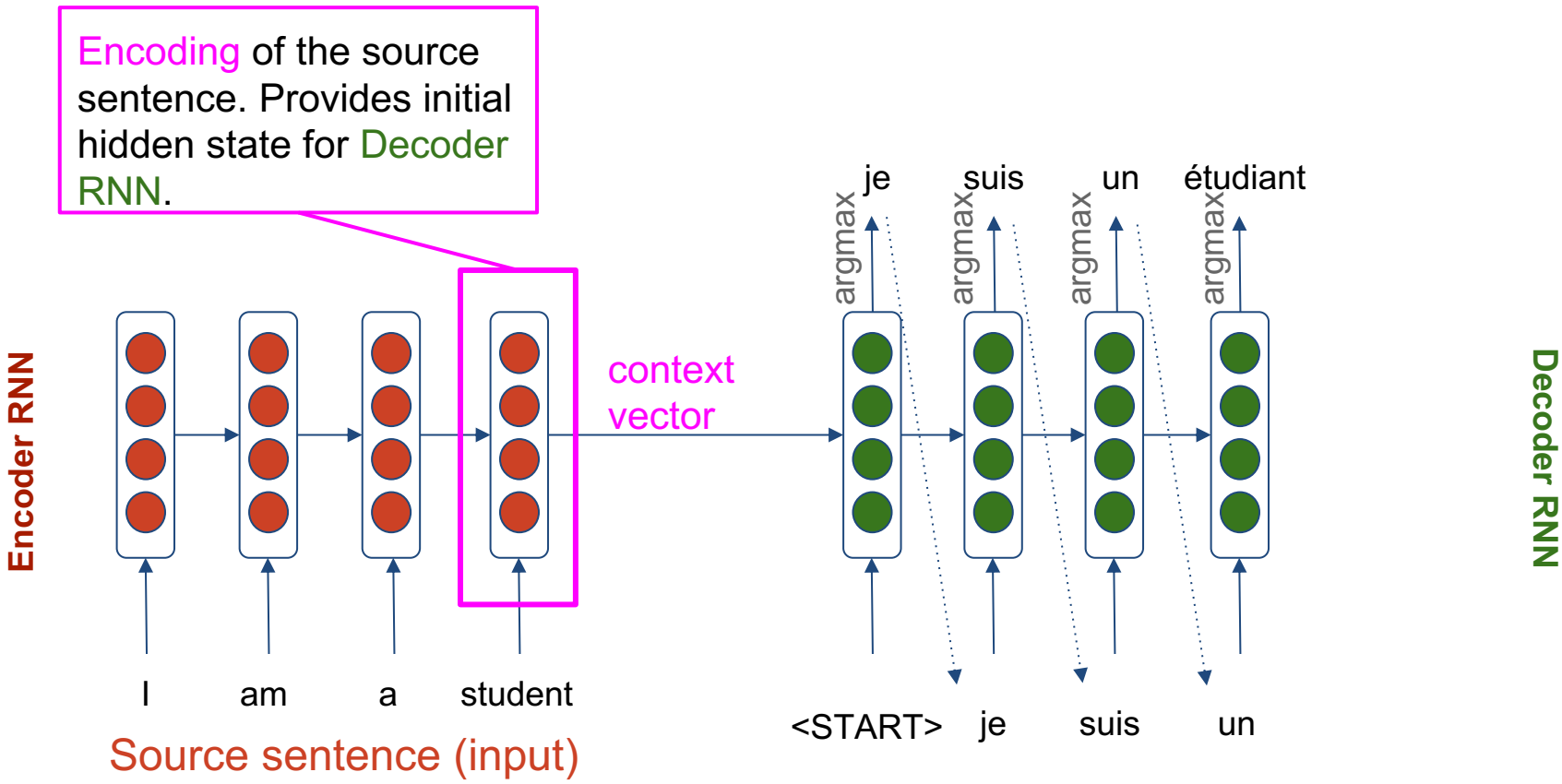
Neural Machine Translation - Seq2Seq



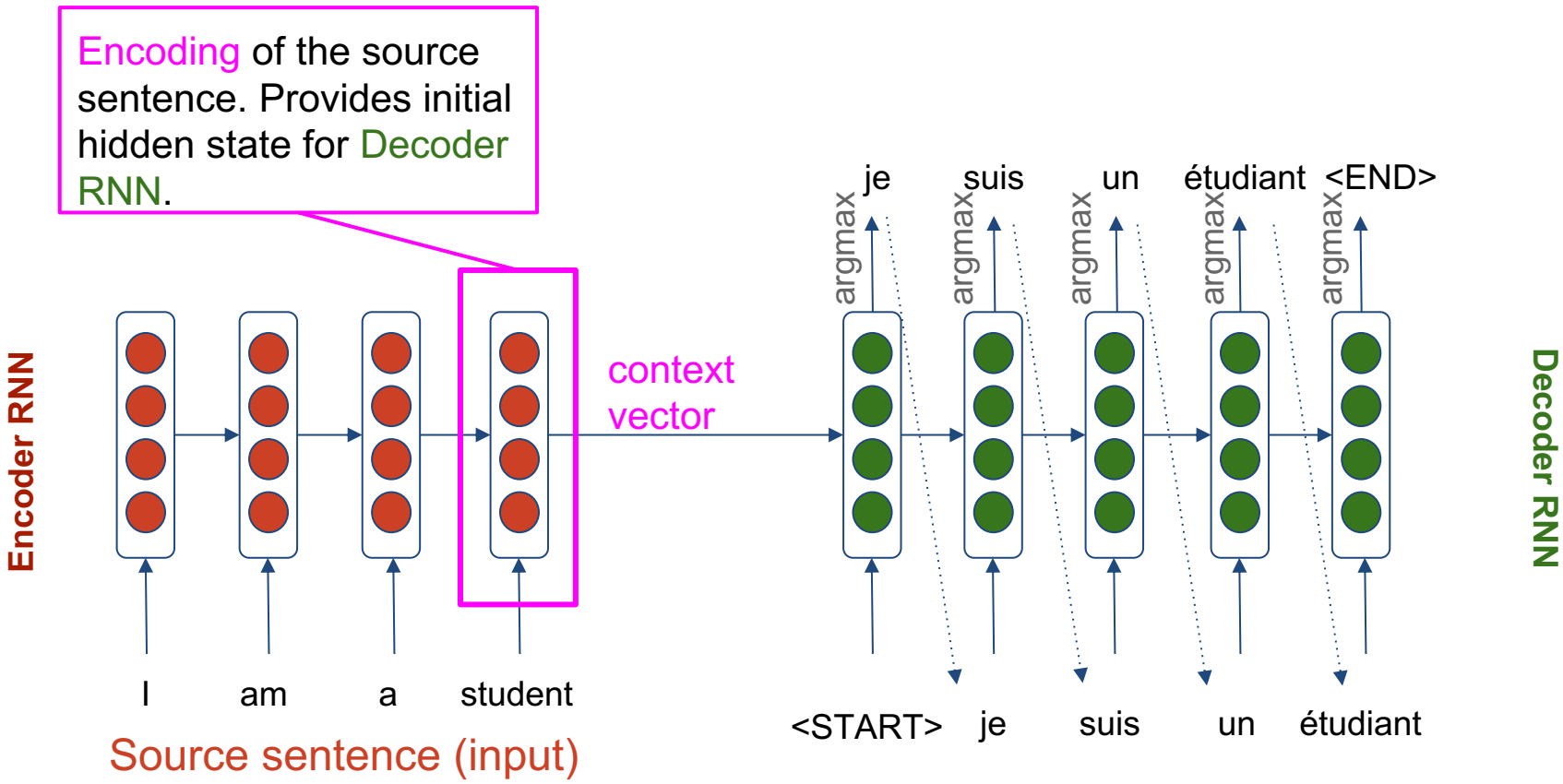
Neural Machine Translation - Seq2Seq



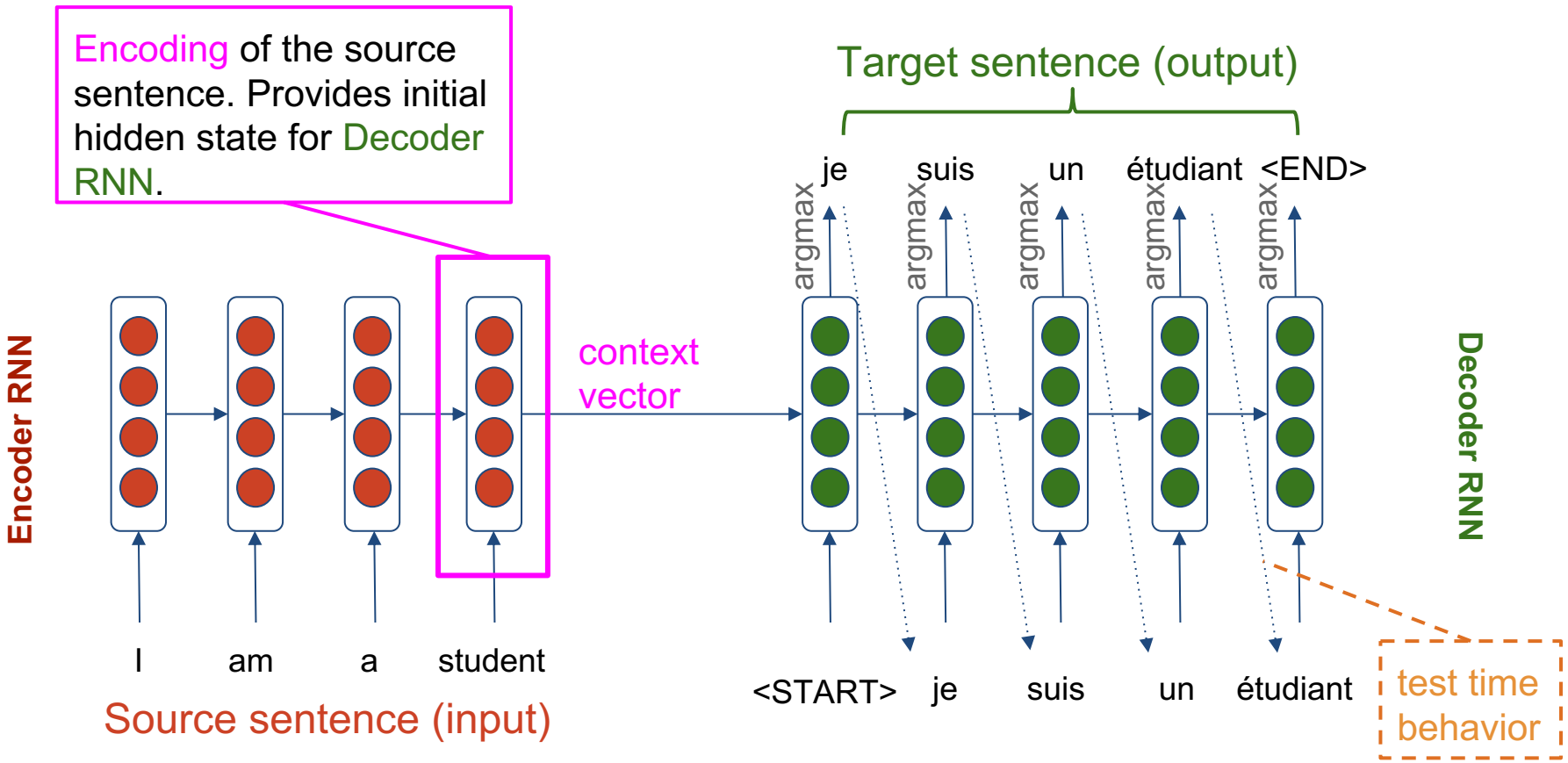
Neural Machine Translation - Seq2Seq



Neural Machine Translation - Seq2Seq



Neural Machine Translation - Seq2Seq



Seq2Seq is versatile!

- Seq2seq 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)
  - Code generation (natural language → Python code)

Neural Machine Translation (NMT)

- The seq2seq 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):

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

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

- How to train a NMT system?
  - Easy answer: Get a big parallel corpus…
  - Now: there is some advanced and exciting work.

Training a Neural Machine Translation system

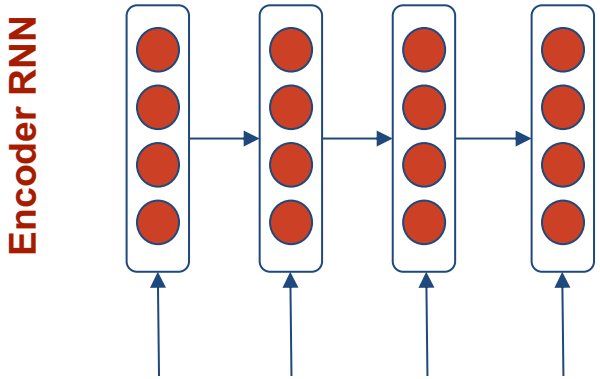
I am a student

Source sentence (from corpus)

<START> je suis un étudiant

Translated sentence (from corpus)

Training a Neural Machine Translation system

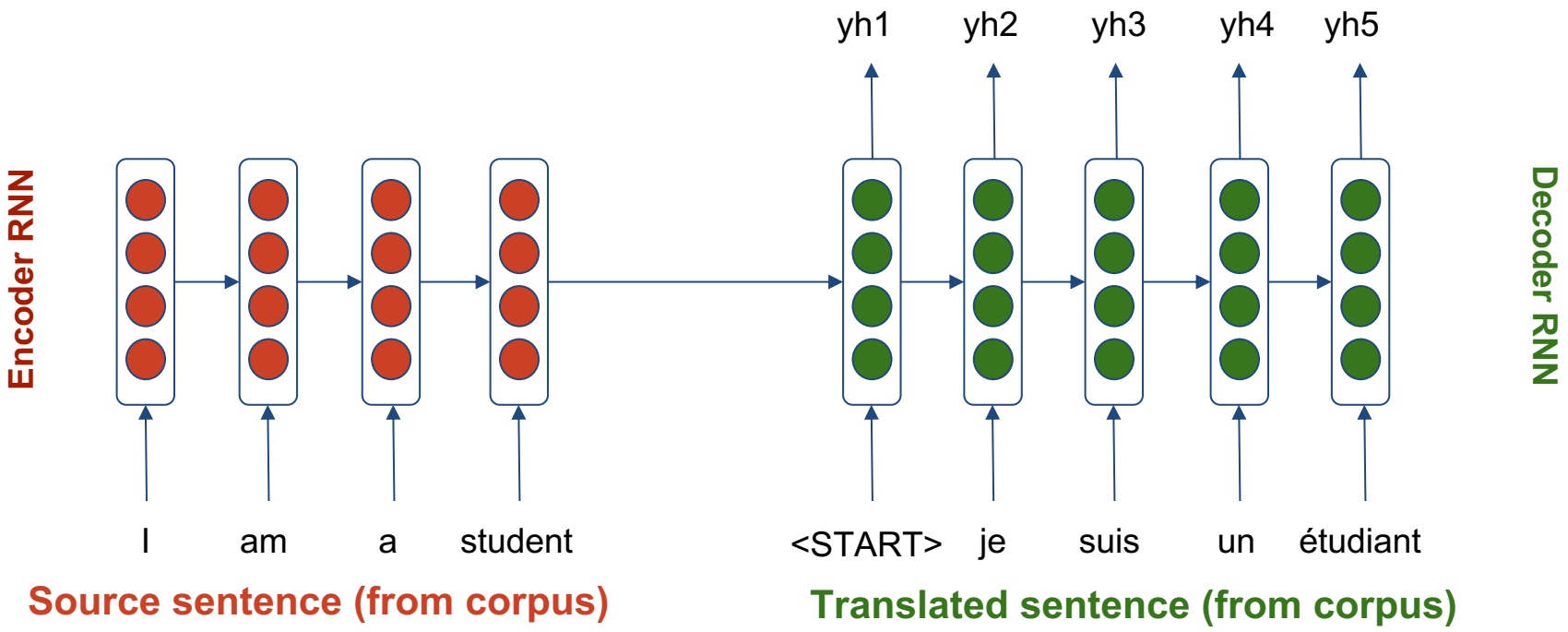


Source sentence (from corpus)

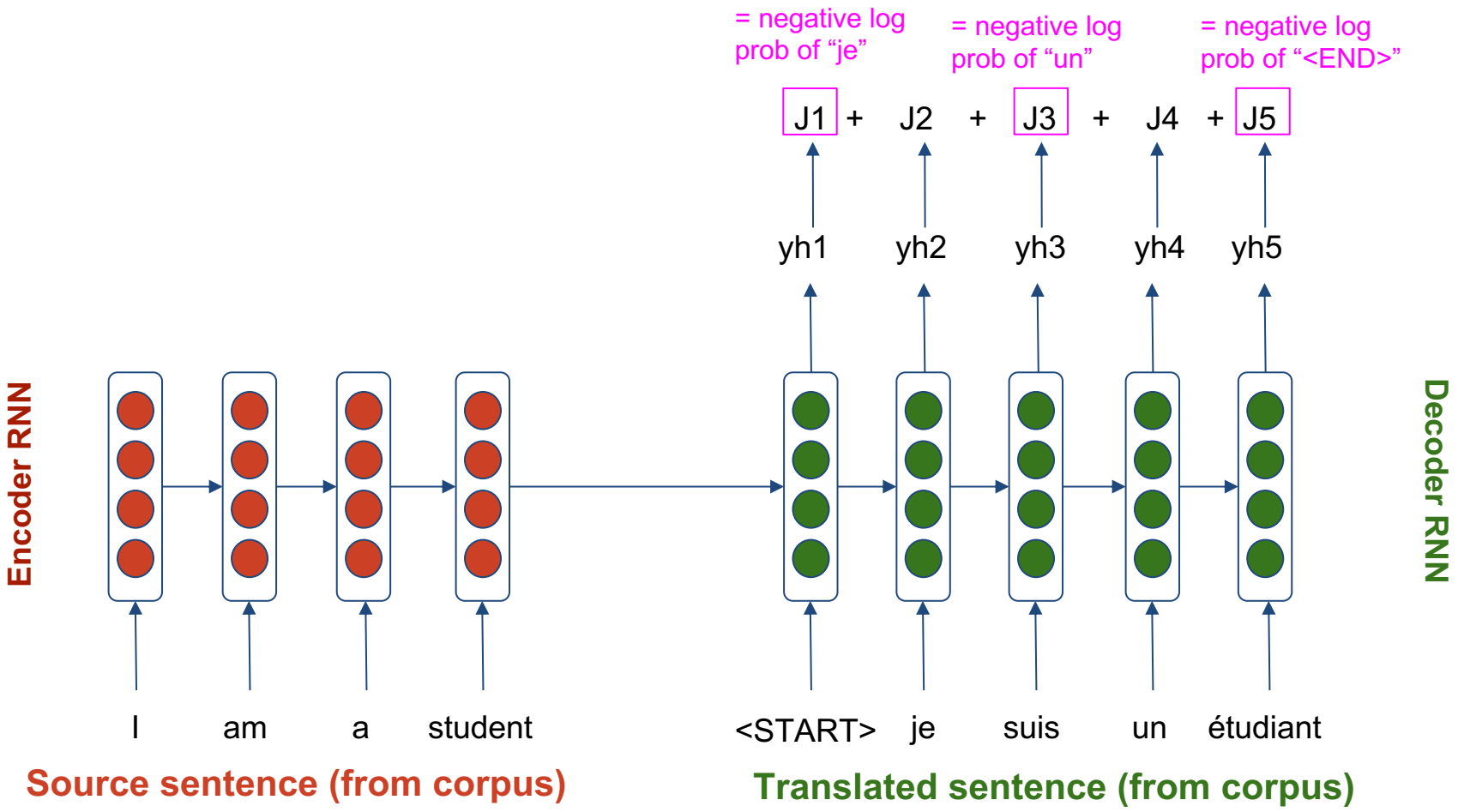
<START> je suis un étudiant

Translated sentence (from corpus)

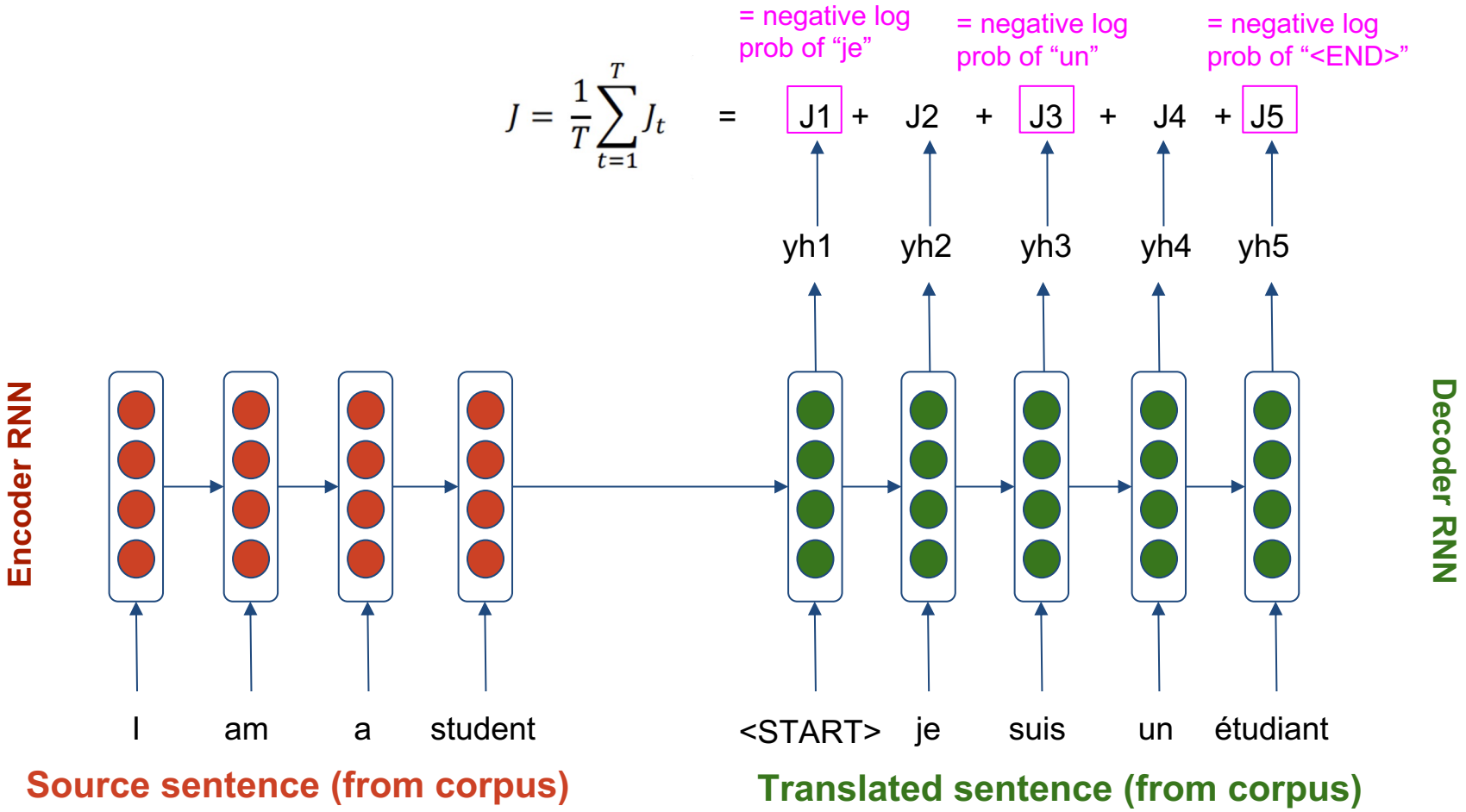
Training a Neural Machine Translation system



Training a Neural Machine Translation system



Training a Neural Machine Translation system



Training a Neural Machine Translation system

Seq2seq is optimized as a single system.

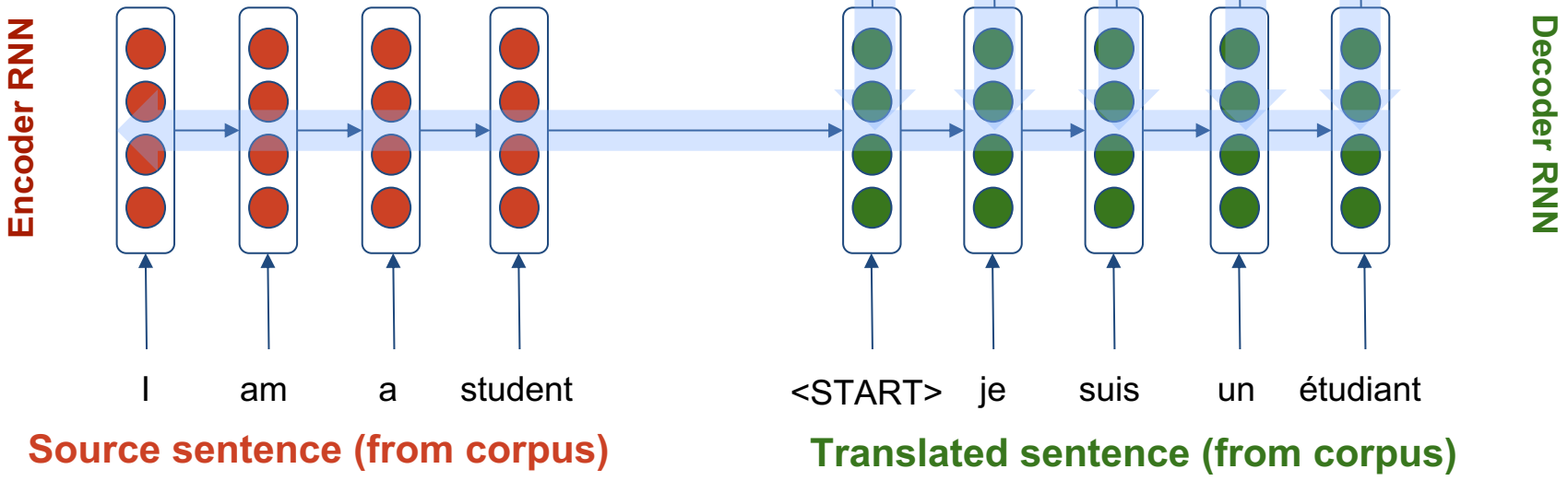
Backpropagation operates "end-to-end".

$$J = \frac{1}{T}\sum_{t=1}^{T} J_t = J1 + J2 + J3 + J4 + J5$$

= negative log prob of "je"

= negative log prob of "un"

= negative log prob of "<END>"

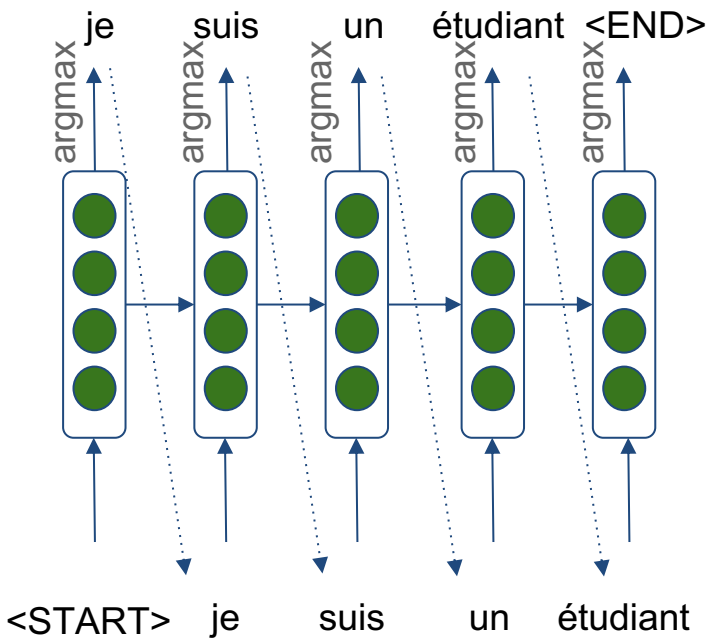


Greedy decoding

- We saw how to generate (or “decode”) the target sentence by taking argmax on each step of the decoder.
- Greedy decoding takes most probable word on each step

- Problems?
  - Greedy decoding has no way to undo decisions!

- How to fix this?
  - Exhaustive search decoding
  - Beam search decoding



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
  - 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 $y_1, \ldots, y_t$ 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
  - 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

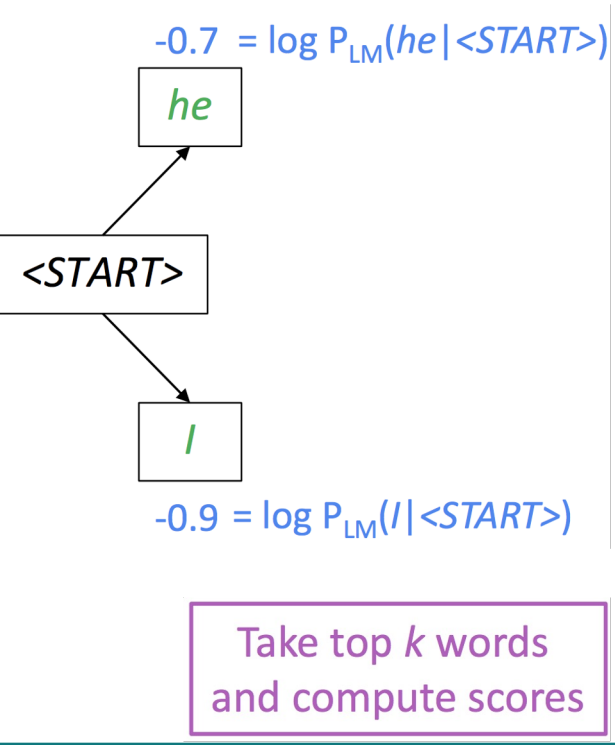
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

<START>

Calculate prob dist of next word

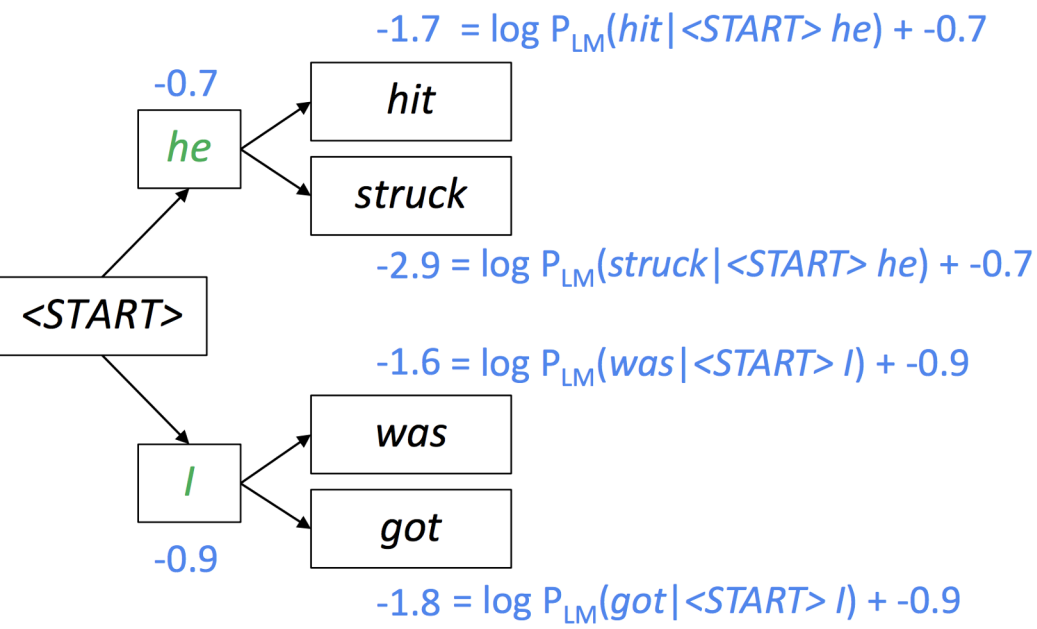
Beam Search Decoding: example

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



Beam Search Decoding: example

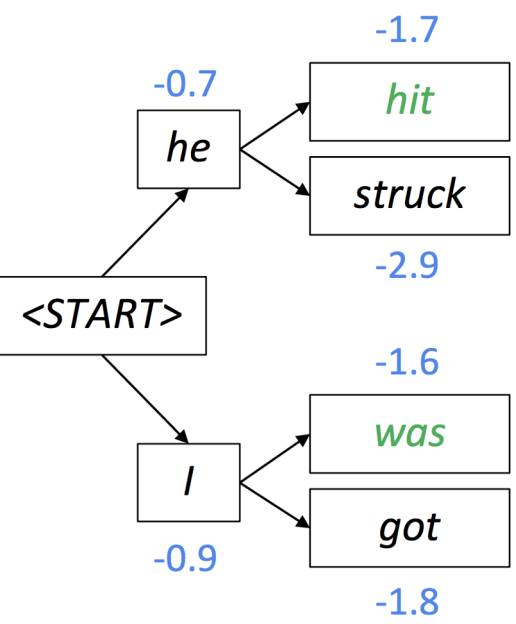
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam Search Decoding: example

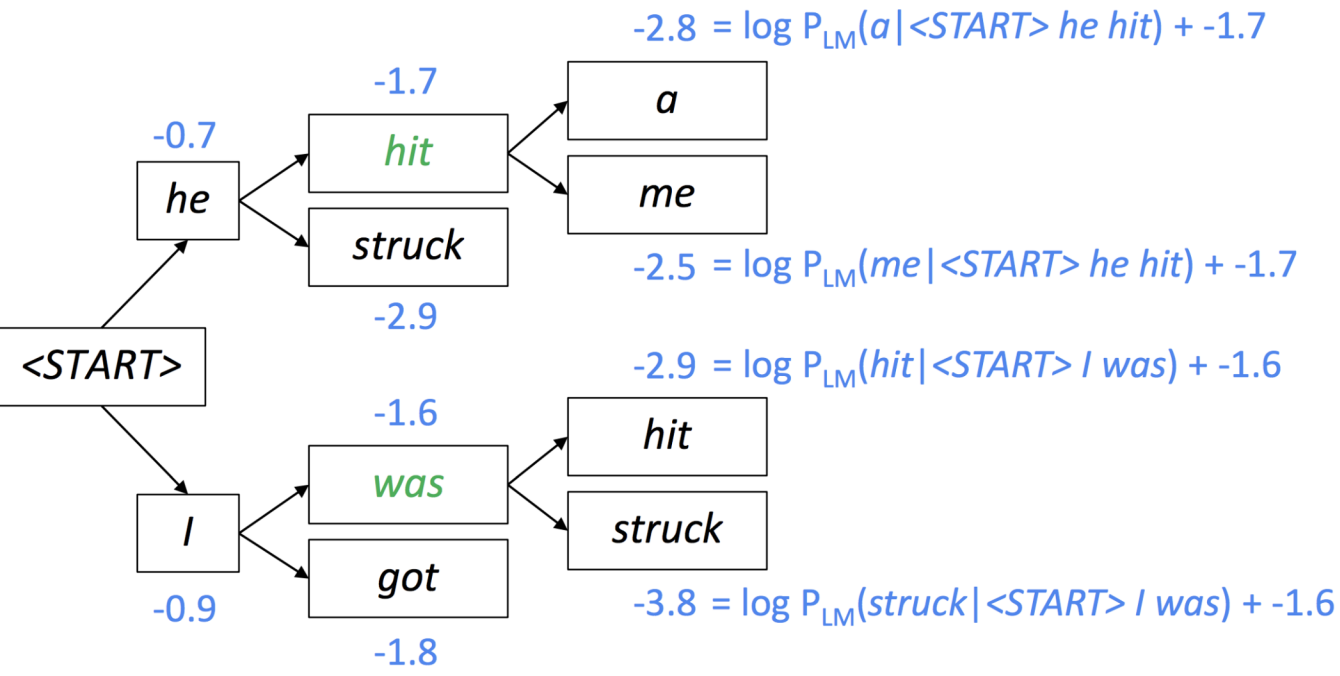
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam Search Decoding: example

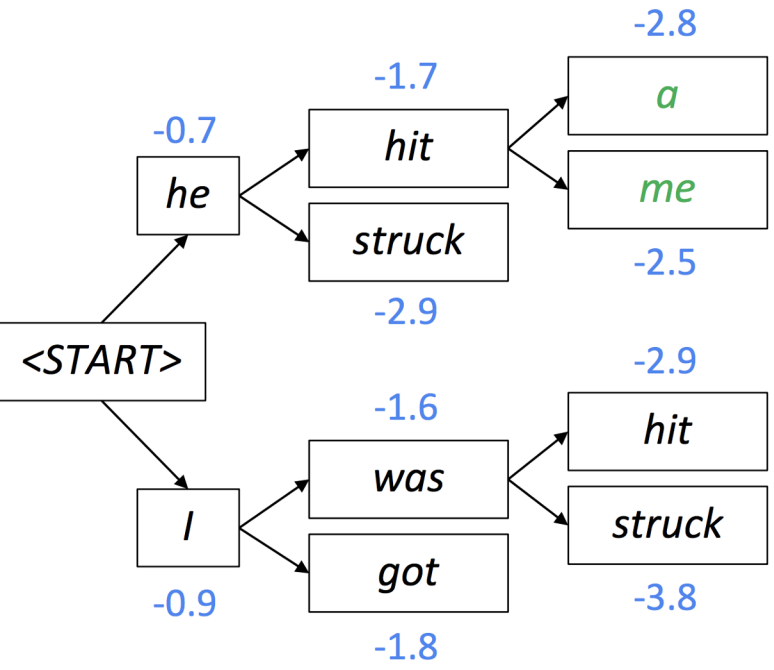
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam Search Decoding: example

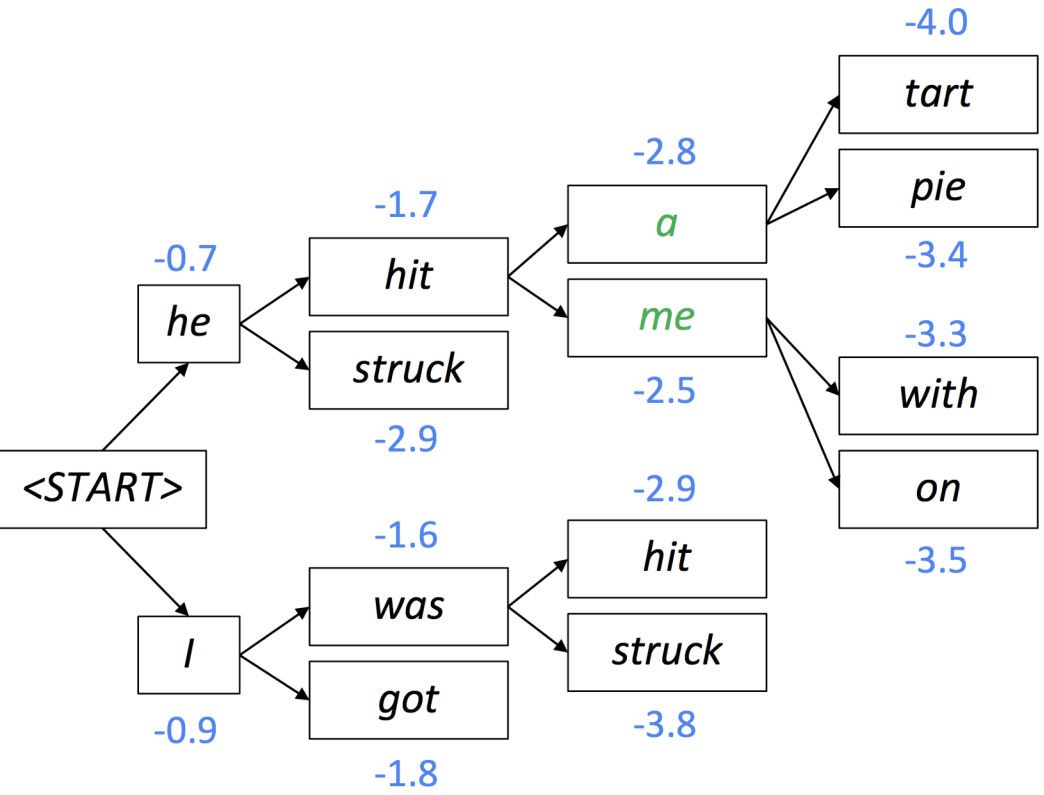
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam Search Decoding: example

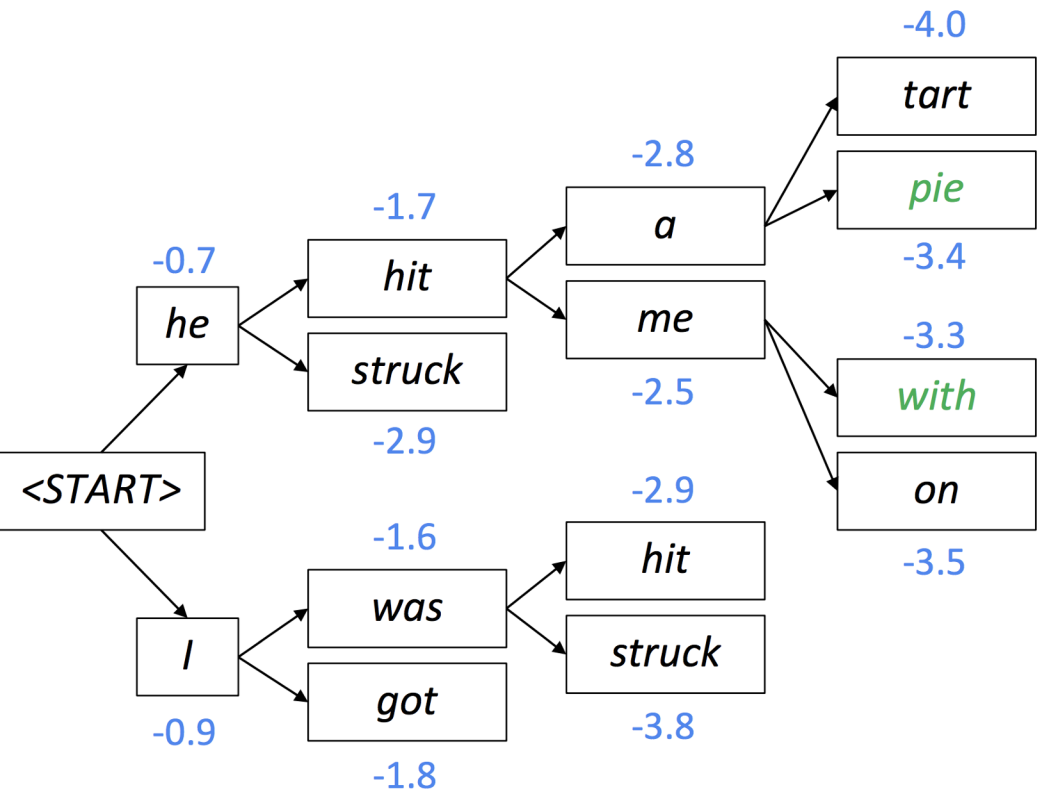
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam Search Decoding: example

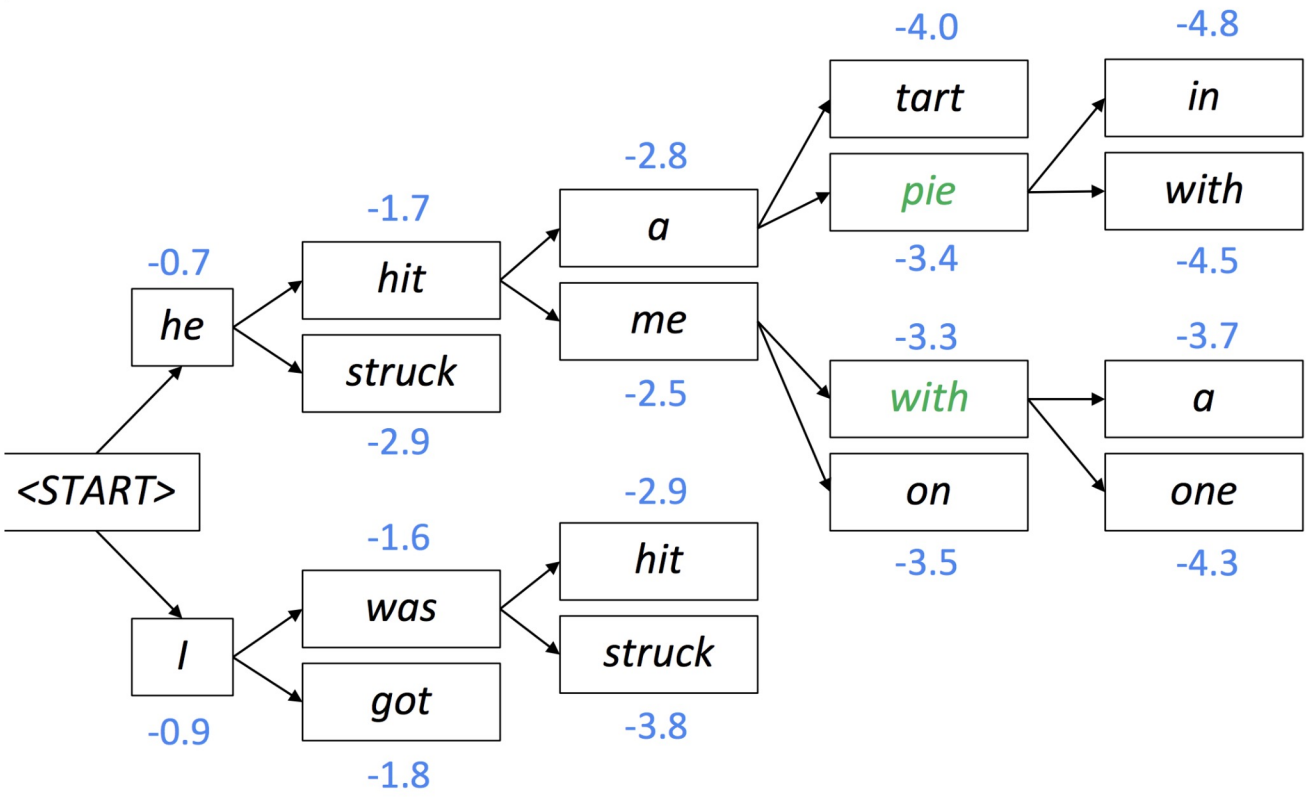
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam Search Decoding: example

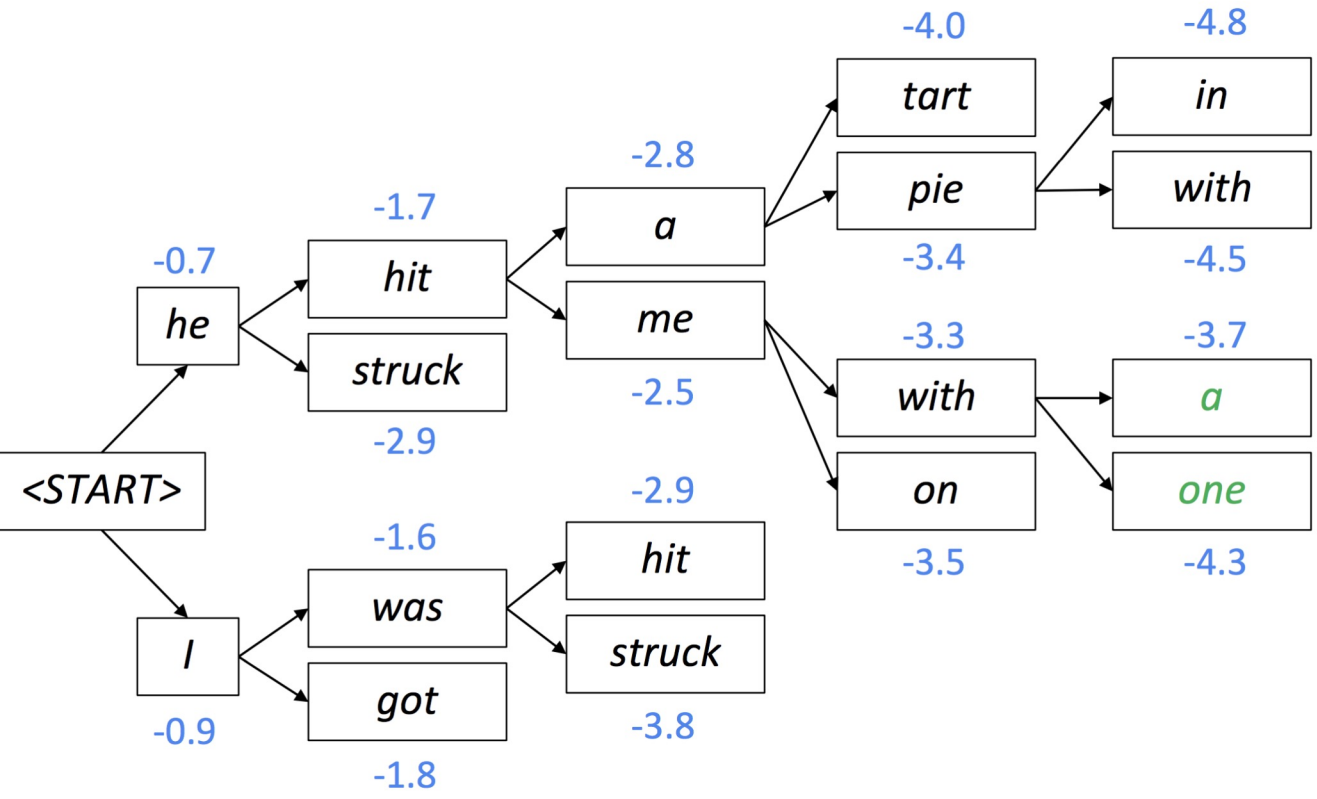
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam Search Decoding: example

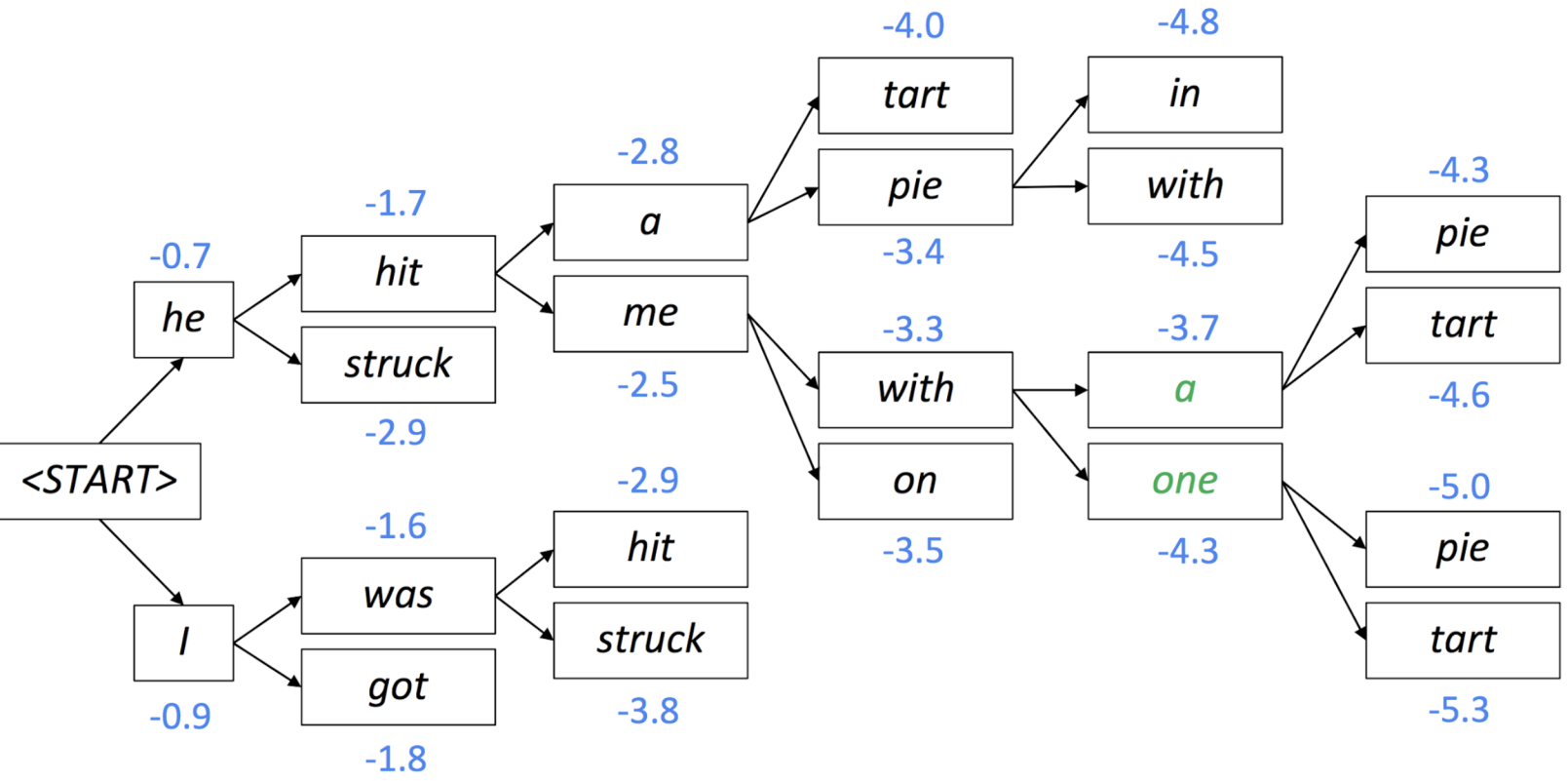
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam Search Decoding: example

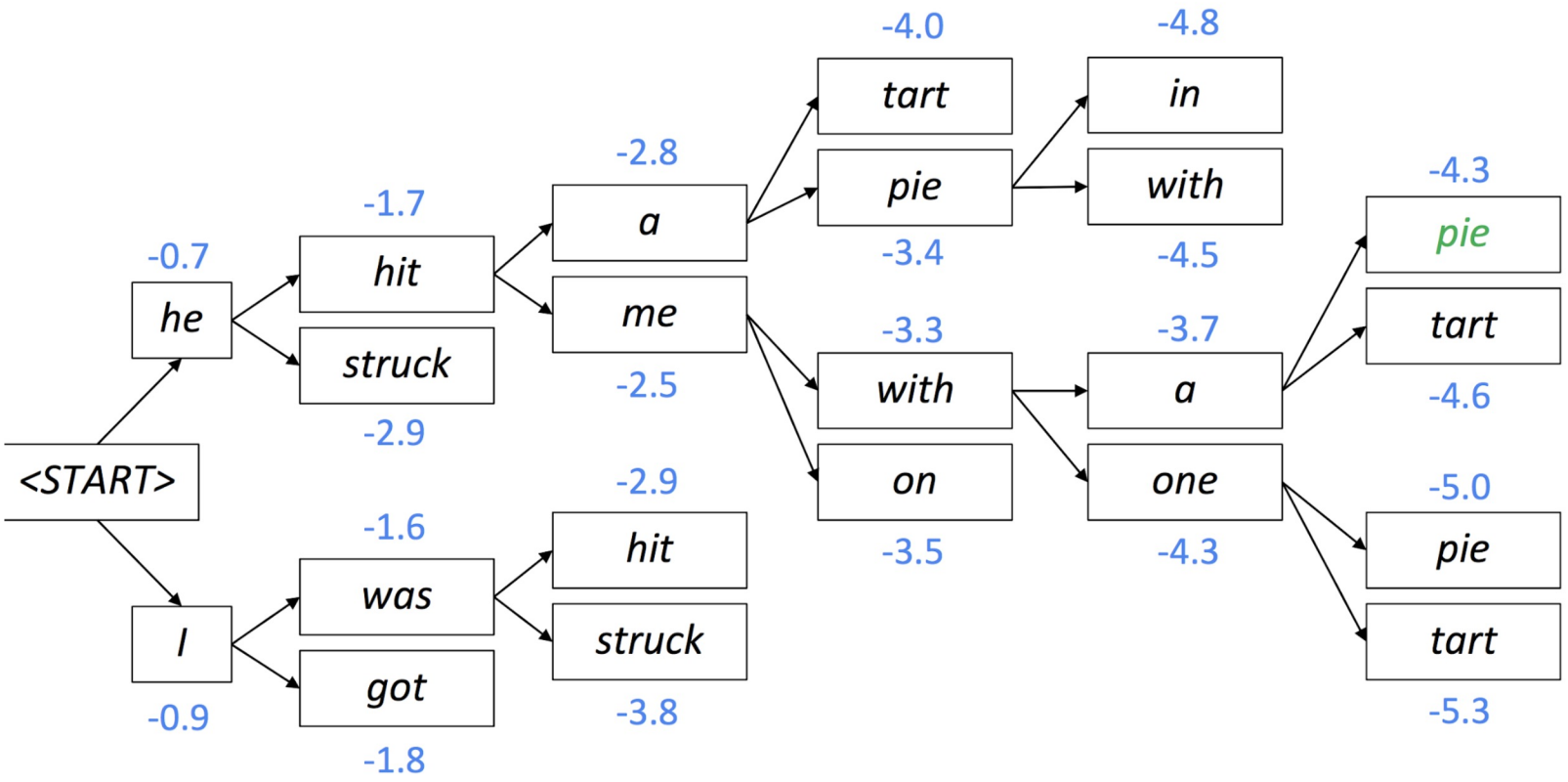
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam Search Decoding: example

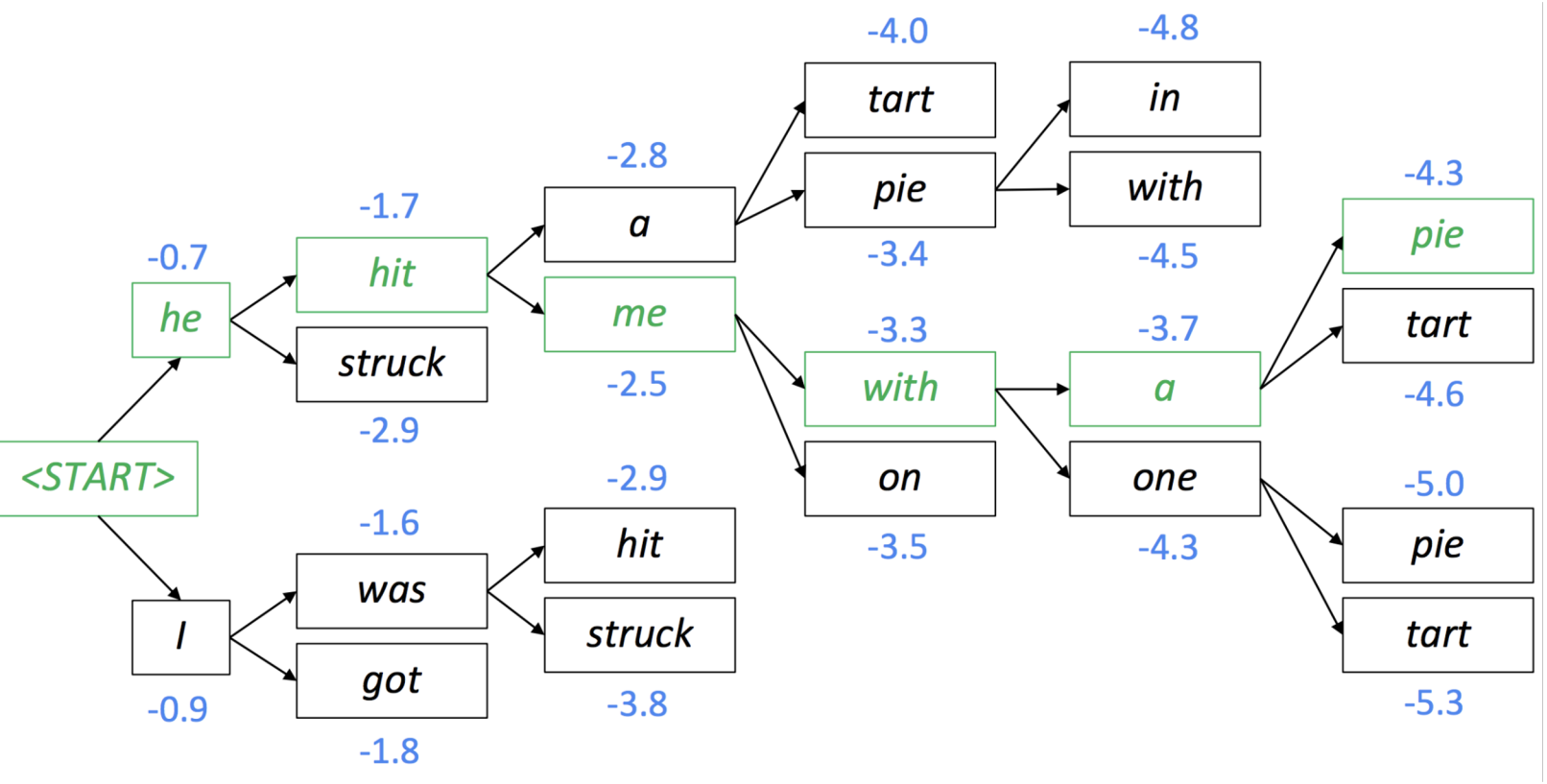
Beam size k = 2. Blue numbers $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



This is the top-scoring hypothesis!

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

Beam Search Decoding: stopping criterion

- In greedy decoding, usually we decode until the model produces a <END> token
  - e.g. <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 $y_1, \ldots, y_t$ 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: longer hypotheses have lower scores

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

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

Beam Search Decoding

Beam search decoding helps in producing more diverse, fluent, and accurate results compared to simple greedy decoding.

- **Diverse Output Generation**: helps in exploring multiple options and selecting the most probable one, which can lead to better overall sequences.
- **Global Context**: considers the cumulative probability of the entire sequence, not just the probability of the next token. This helps in generating coherent and meaningful outputs.
- **Handling Variable Length Outputs**: allows the model to generate until a stopping condition is met.

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

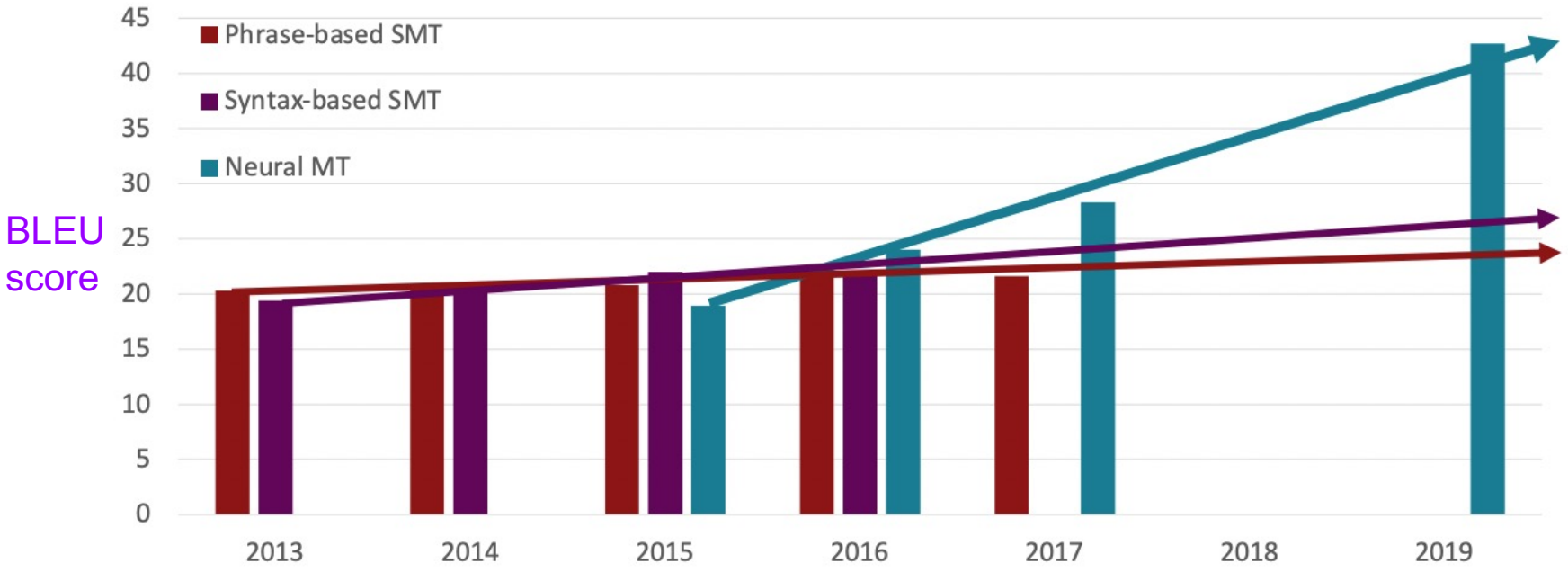
How do we evaluate Machine Translation?

- BLEU (**B**ilingual **E**valuation **U**nderstudy) compares the machine-written translation to one or several human-written reference translation(s), and computes a similarity score based on:
  - n-gram precision (usually for 1, 2, 3 and 4-grams): measures the proportion of correctly generated n-grams in the machine translation.
  - Plus a penalty for too-short system translations: prevents from producing very short, meaningless translations that might have high precision by chance.
  - Higher BLEU scores indicate better translation quality.

- 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.
  - Can be used combined with human evaluation.

MT progress over time

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal; NMT 2019 FAIR on newstest2019]



source: https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture06-fancy-rnn.pdf

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!"

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

  - We will discuss in our next lecture

Summary

- History of Machine Translation (MT)
  - Statistical Machine Translation (SMT)

- Seq2Seq is the architecture for NMT (uses 2 RNNs)

- Next: Attention is a way to focus on particular parts of the input
  - Improves seq2seq a lot!

Readings

- **CH 10 DL; CH 17 NNLP**
- Sutskever et al. Sequence to Sequence Learning with Neural Networks. 2014 (original seq2seq NMT paper)
- Papineni et al. BLEU: a Method for Automatic Evaluation of Machine Translation. 2002.
- Alex Graves. Sequence Transduction with Recurrent Neural Networks (early seq2seq speech recognition paper)

STEVENS

INSTITUTE *of* TECHNOLOGY

THE INNOVATION UNIVERSITY®

stevens.edu

Thank You