Neural Machine Translation

CS4742 Natural Language Processing
Lecture 00

Jiho Noh

Department of Computer Science
Kennesaw State University

CS4742 Summer 2025 [1]

KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING

[1] This lecture is based on the slides from Dr. Hafiz Khan at KSU.

1. Encoder-Decoder for Machine Translation
2. Sequence to Sequence (Seq2Seq)
3. Beam Search
4. Attention Mechanism

Neural Machine Translation

- A single **neural network** is used to translate from *source* in one language to *target* in another language
- Typical architecture: Encoder-Decoder
  - **Encoder**: reads the source sentence and encodes it into a fixed-length *context vector*
  - **Decoder**: generates the target sentence from the *context vector*

Encoder-Decoder Model

- **Encoder-Decoder** models are essential in *sequence-to-sequence* tasks.

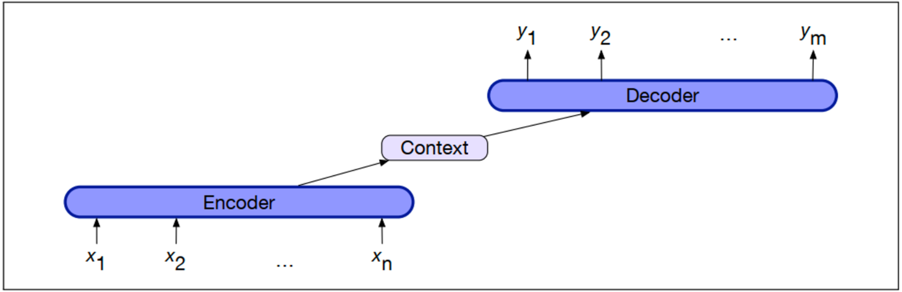
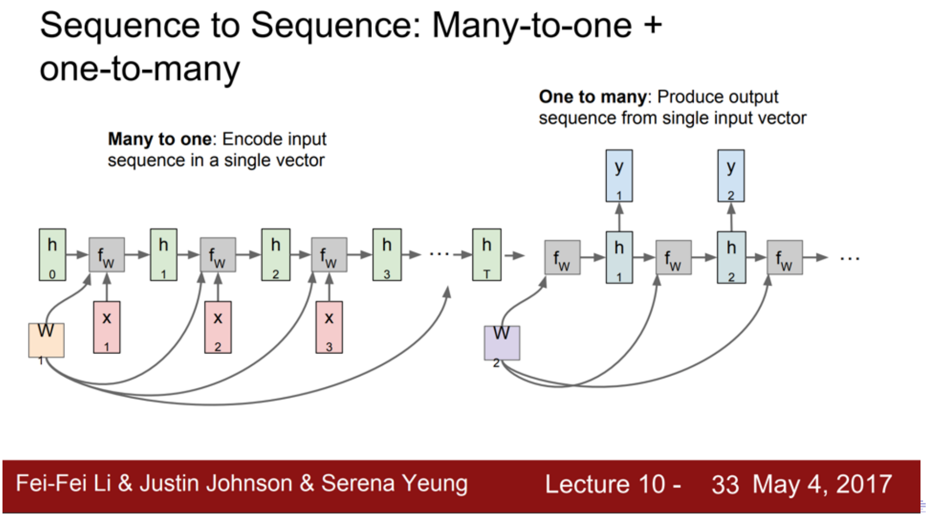


Figure 11.3 The encoder-decoder architecture. The context is a function of the hidden representations of the input, and may be used by the decoder in a variety of ways.

Recall: RNNs

- RNNs use *sequential information* and maintain a hidden state.



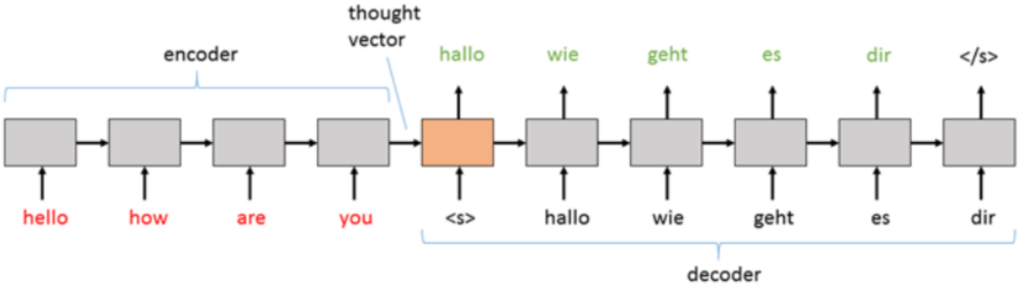
1 Encoder-Decoder for Machine Translation

2 **Sequence to Sequence (Seq2Seq)**

3 Beam Search

4 Attention Mechanism

Sequence to Sequence (Seq2Seq)

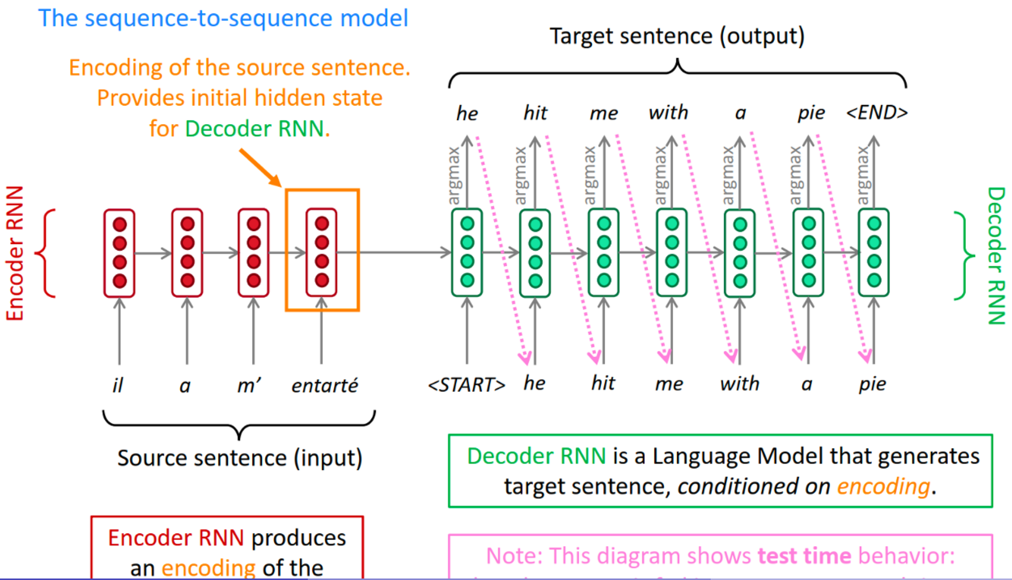


- **Encodes** the input sequence into a single vector (using an RNN)
- **Decode** one word at a time (again using an RNN; this method is called *auto-regressive* decoding)
  - **Beam search** is used to select the best inference

2

[2] Sutskever et al., 2014

Neural Machine Translation — *Information Flow*



Seq2seq Models in Applications

- **Sequence-to-sequence models** are used in many applications:
- Many *NLP tasks* can be phrased as sequence-to-sequence:
  - Summarization (long text $\rightarrow$ short text)
  - Dialogue (previous utterances $\rightarrow$ next utterance)
  - Parsing (input text $\rightarrow$ output parse as sequence)
  - Code generation (natural language $\rightarrow$ Python code)

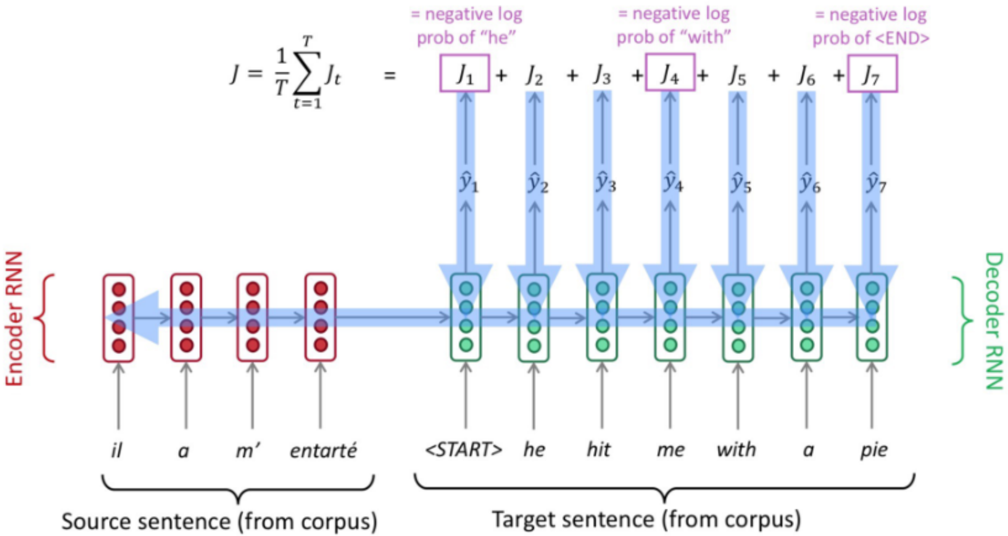
Seq2seq Training

- Similar to training a **language model**!
- Minimize *cross-entropy loss*

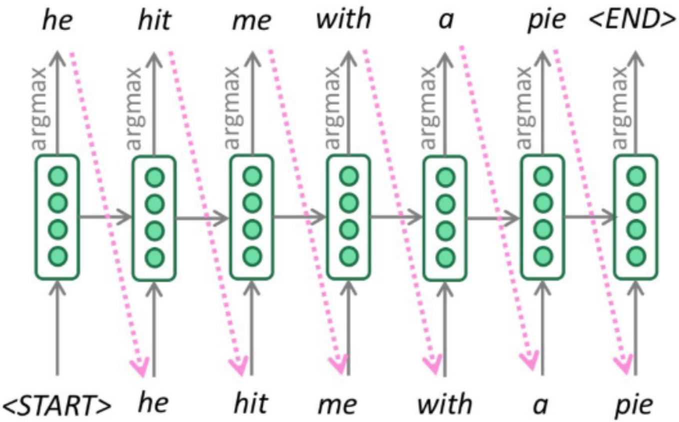
$$\sum_{t=1}^{T} - \log P\left(y_t | y_1, \ldots, y_{t-1}, x_1, \ldots, x_n\right)$$

- Need a big parallel corpus, such as Europarl (European Parliament Proceedings Parallel Corpus)
  - Europarl (French-English) contains 2,007,723 pairs of sentences

Seq2seq Training



Greedy Decoding



Takes the most probable word at each time step. **Any problems with this method?**

Exhaustive Search?

- Find $\arg\max_{y_1,\ldots,y_T} P\left(y_1,\ldots,y_T \mid x_1,\ldots,x_n\right)$
- We could *try* computing all possible sequences
  - $O(V^T)$ is expensive, where V is the vocabulary size and T is the length of the output sequence

1 Encoder-Decoder for Machine Translation

2 Sequence to Sequence (Seq2Seq)

3 Beam Search

4 Attention Mechanism

Beam Search

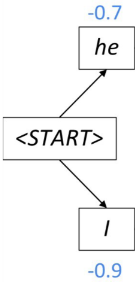
- **Core idea**: On each step of decoder, keep track of the *k most probable partial translations* (which we call hypotheses).
- A hypothesis $Y_1, \ldots, Y_t$ has a **score** which is its *log probability*:

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

- *Not guaranteed to be optimal*, but **more efficient** than exhaustive search.

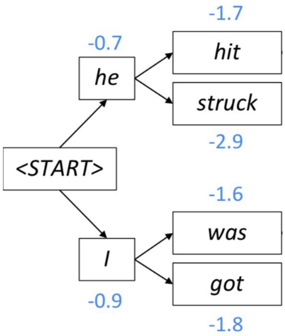
Beam Decoding

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



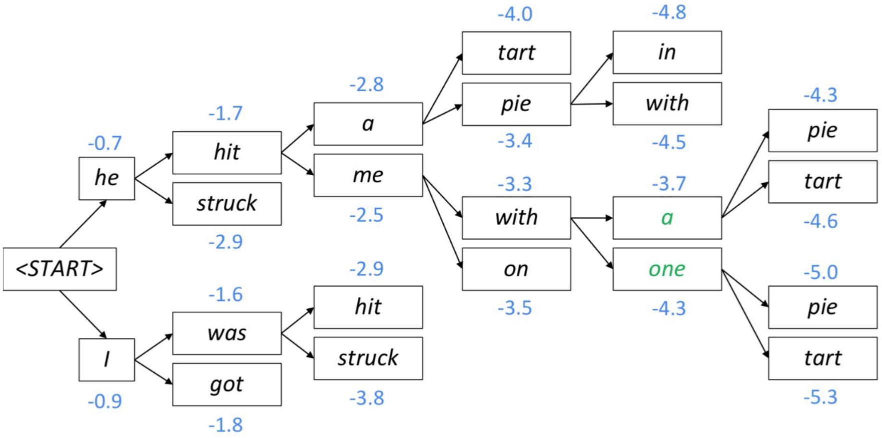
Beam Decoding

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



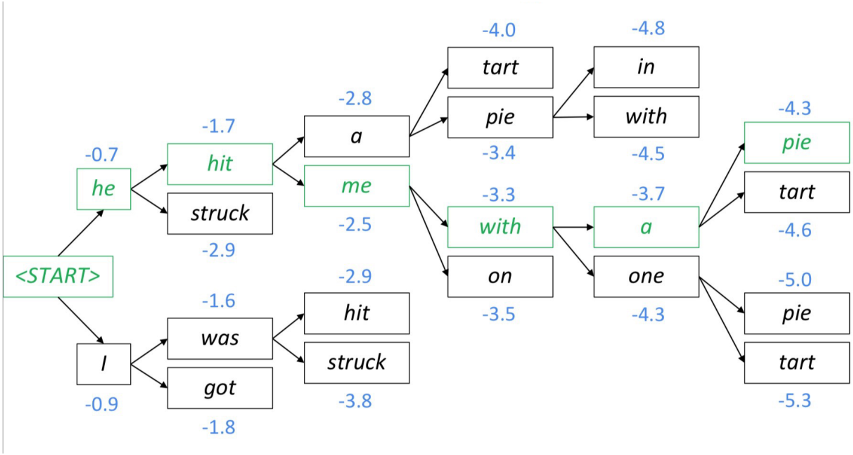
Beam Decoding

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



Beam Decoding

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



Decode until When?

- Different hypotheses may produce `⟨end⟩` token at different time steps.
- When a hypothesis produces `⟨end⟩`, stop expanding it and place it aside.
- Continue beam search until:
  - All k hypotheses produce `⟨end⟩` **OR**
  - Hit max decoding limit T
- Select top hypotheses using the normalized likelihood score

$$\frac{1}{T}\sum_{t=1}^{T}\log P\left(y_t \mid y_1,\ldots,y_{t-1},x_1,\ldots,x_n\right)$$

- Various normalization methods may consider the *length* of the hypothesis.

Beam Search Normalization — Wu et al. (2016)

- Regular beam search favor shorter results since a negative values of log-probability is added at each step.
- We need some form of length normalization to compare different length of hypotheses.
- In Wu et al. (2016),

$$\mathrm{score}(\mathrm{Y}, \mathrm{X}) = \frac{\log \mathrm{P}(\mathrm{Y} \mid \mathrm{X})}{\mathrm{lp}(\mathrm{Y})} + \mathrm{cp}(\mathrm{X}, \mathrm{Y})$$

- The **length penalty** is defined as below:

$$\mathrm{lp}(Y) = \frac{(5 + |Y|)^{\alpha}}{(5+1)^{\alpha}}$$

  - where $|Y|$ is the current target length and α is the *length penalty hyperparameter*.

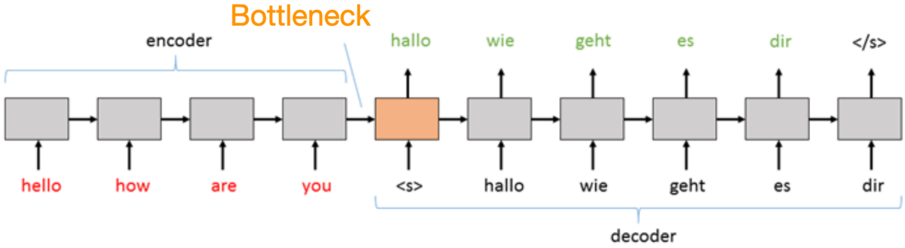
Beam Search Normalization — Wu et al. (2016)

- Scores are also penalized by the following formula:

$$\mathrm{cp}(X, Y) = \beta \sum_{i=1}^{|X|} \log \left(\min \left(\sum_{j=1}^{|Y|} p_{i,j}, 1.0 \right) \right)$$

- where $p_{i,j}$ is the attention probability of the j-th target word y_j on the i-th source word x_i
- $|X|$ is the source length, $|Y|$ is the current target length
- β is the **coverage normalization coefficient**
- The idea is to **encourage** the model to translate *all* of the provided input.

Issues with Vanilla Seq2Seq (or RNN)



- **A single encoding vector**, h^{enc}, *needs to capture all the information about the source sentence*
- Longer sequences can lead to **vanishing gradients**
- **Overfitting**

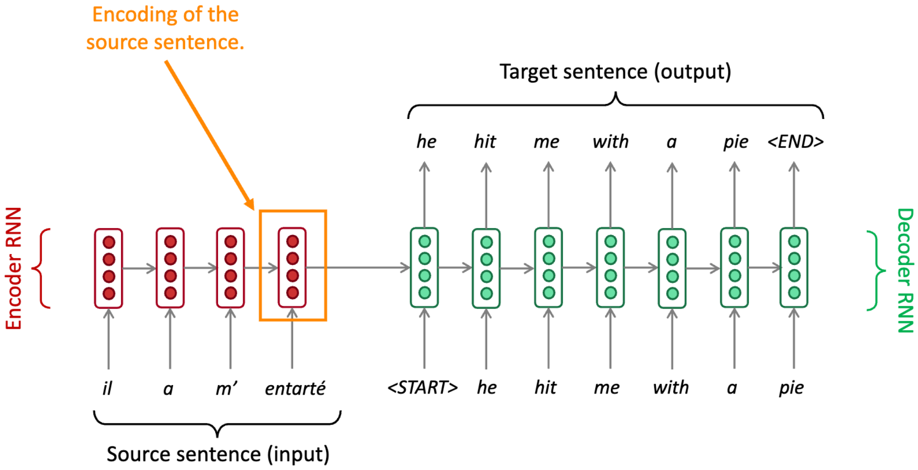
1 Encoder-Decoder for Machine Translation

2 Sequence to Sequence (Seq2Seq)

3 Beam Search

4 Attention Mechanism

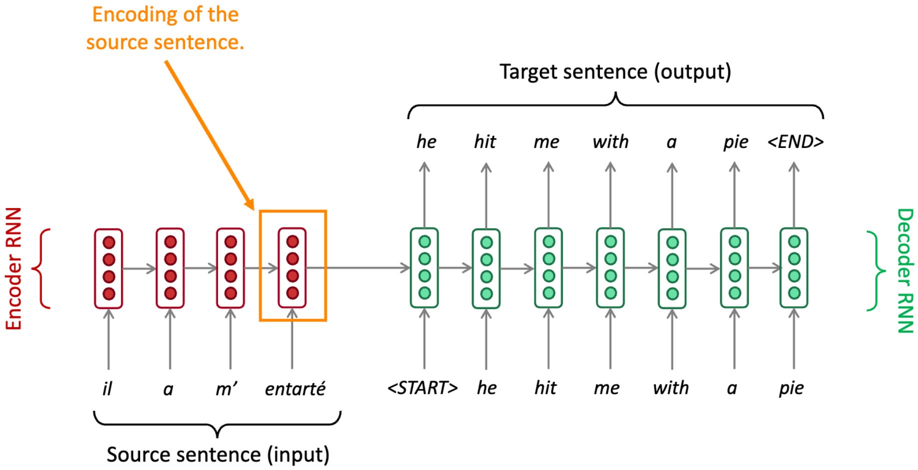
Attention Revisited



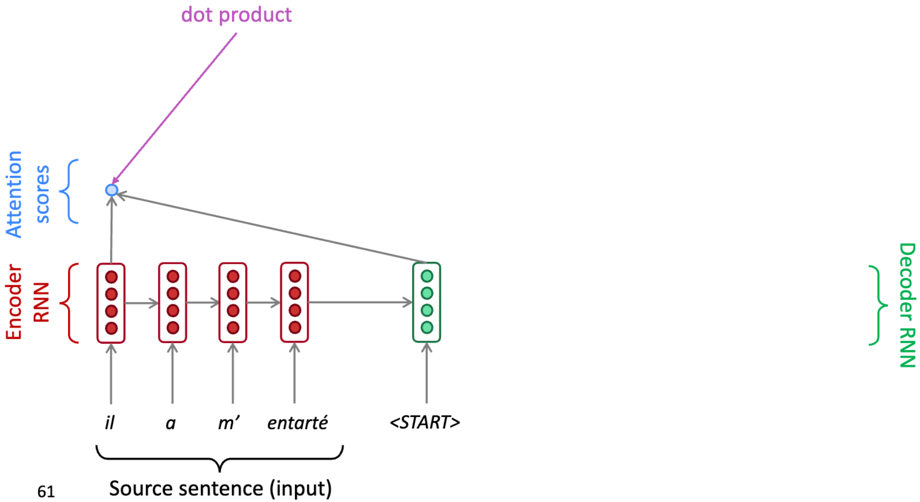
Problems with this architecture?

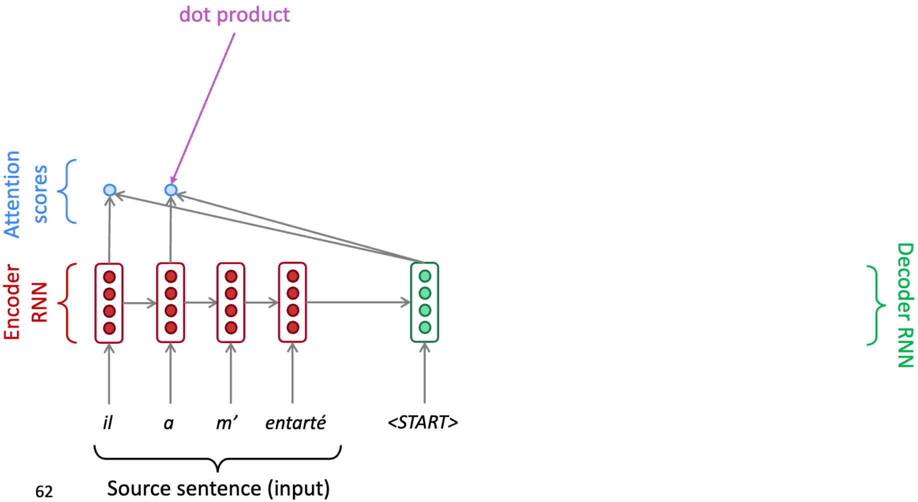
4

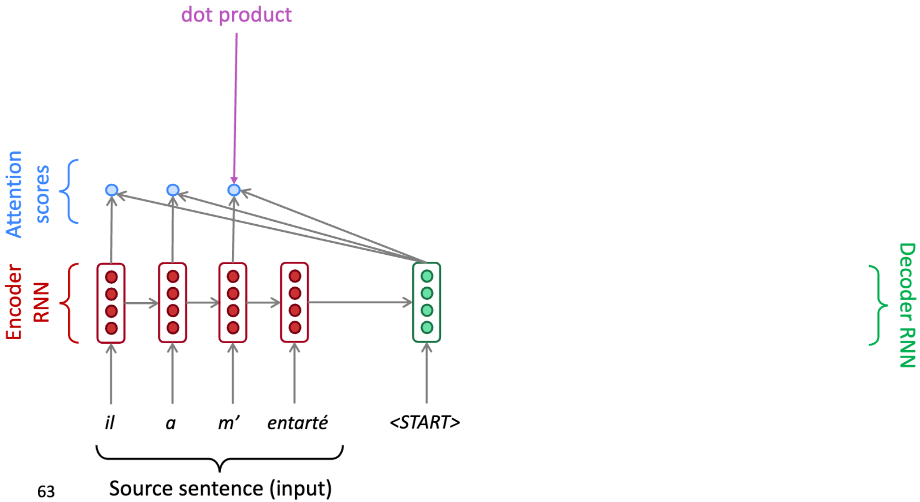
[4]Slides credit: Abigail See and https://www.site.uottawa.ca/~diana/csi5386/NMT.pdf



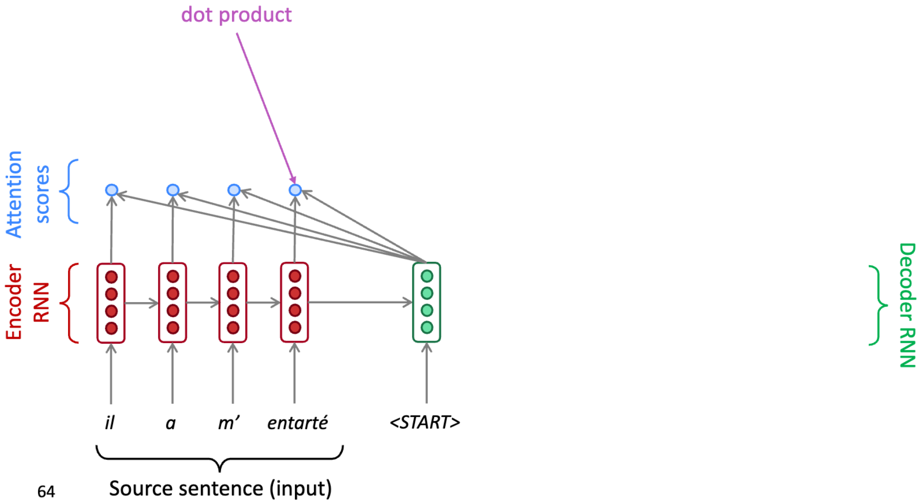
Problems with this architecture?

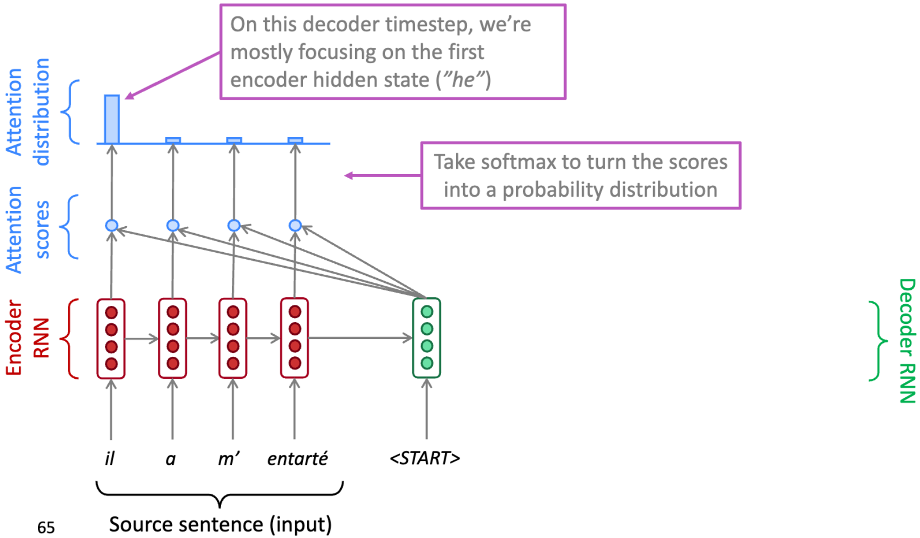

dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)
61

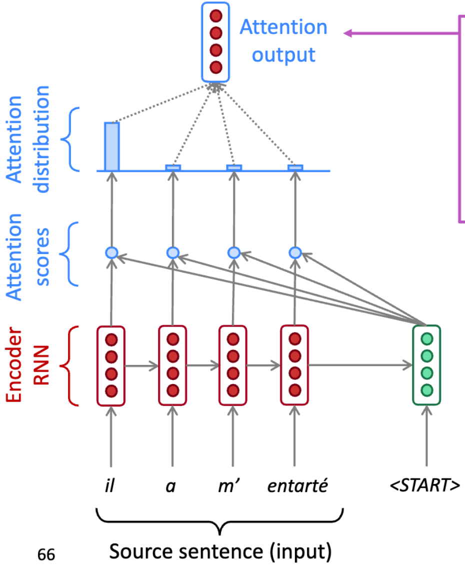

dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)
62



dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)
63


dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)
64

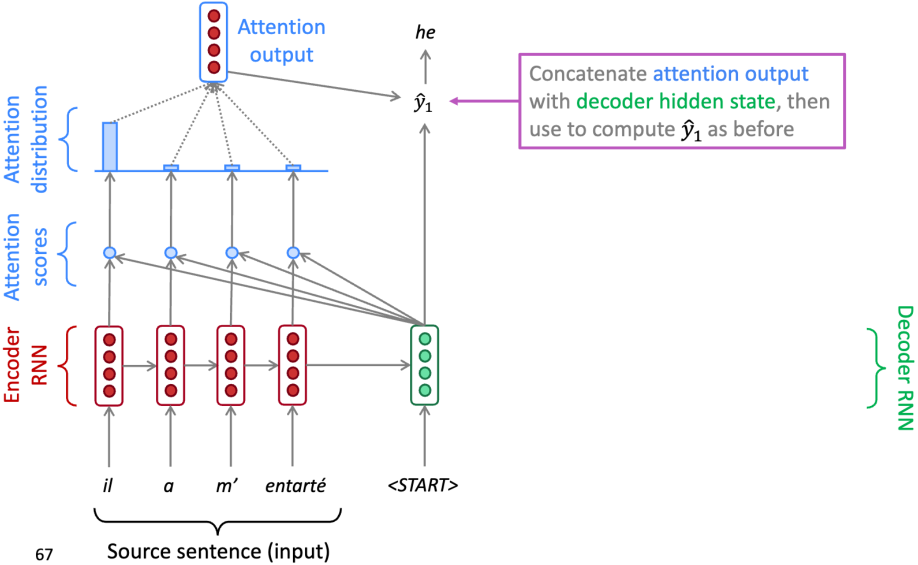

On this decoder timestep, we're mostly focusing on the first encoder hidden state ("*he*")
Take softmax to turn the scores into a probability distribution
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)
65



Use the attention distribution to take a **weighted sum** of the encoder hidden states.

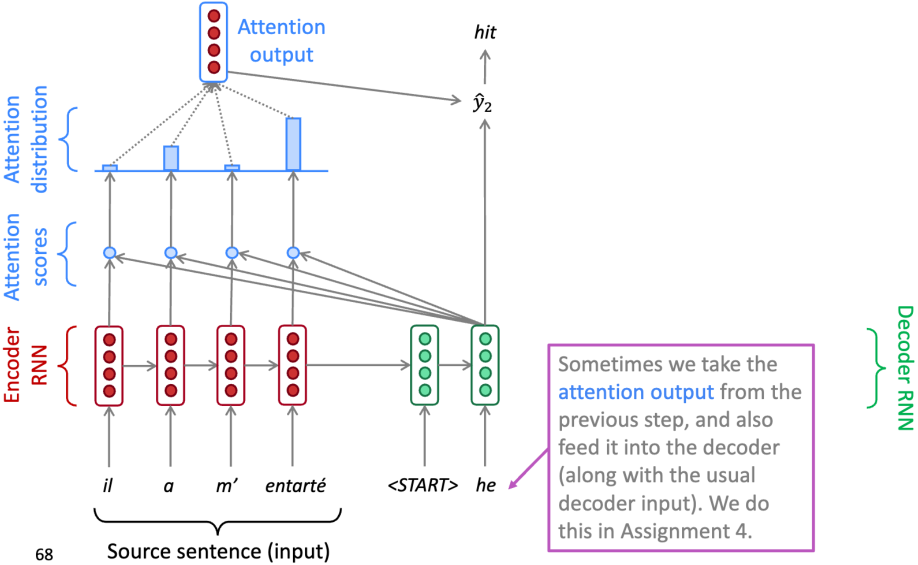
The attention output mostly contains information from the hidden states that received high attention.

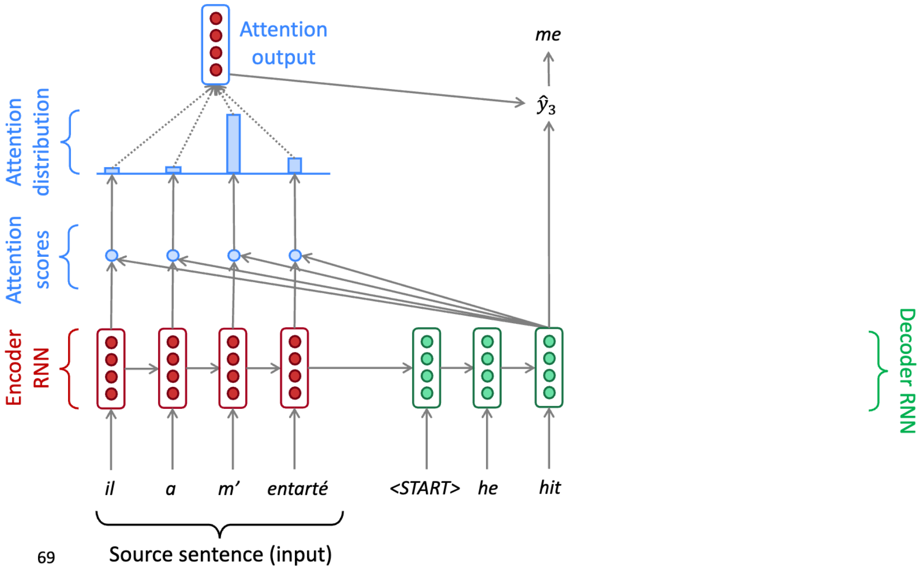
66

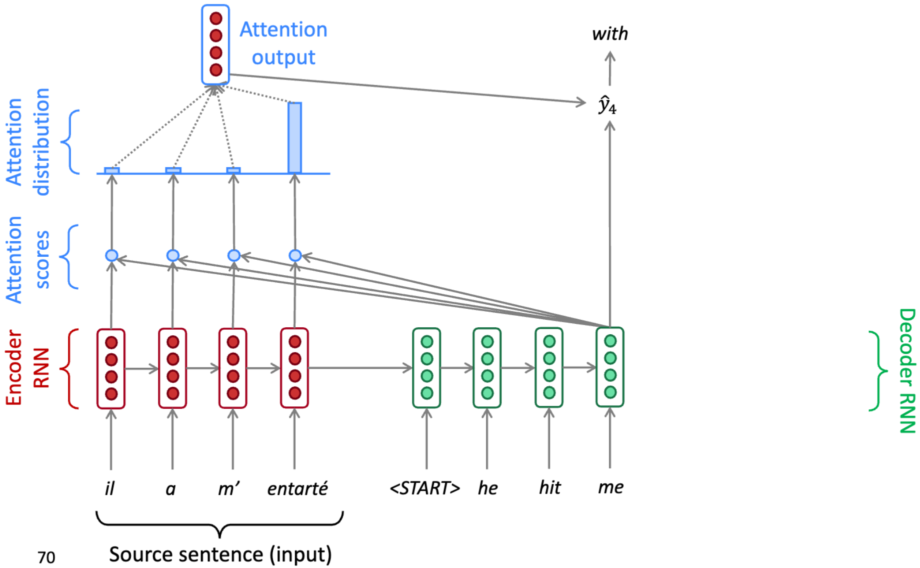


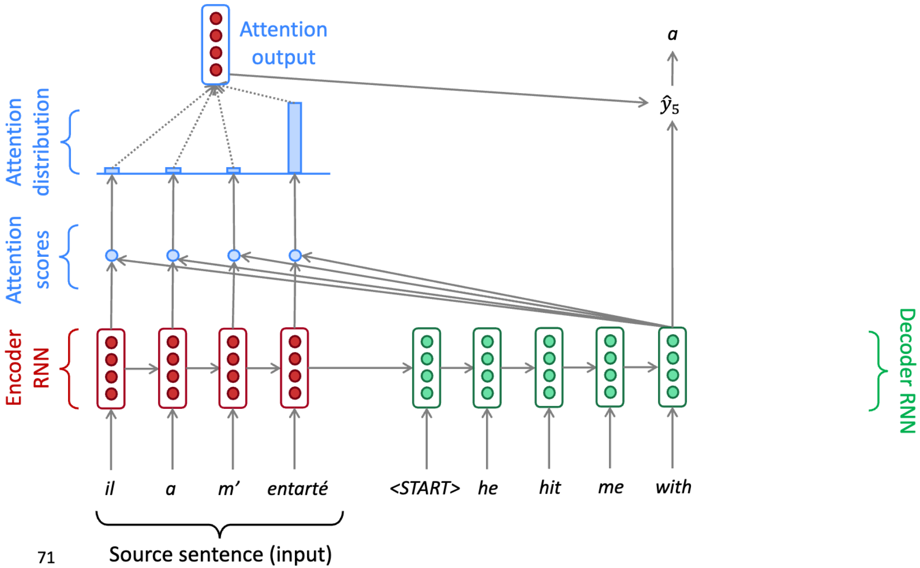
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

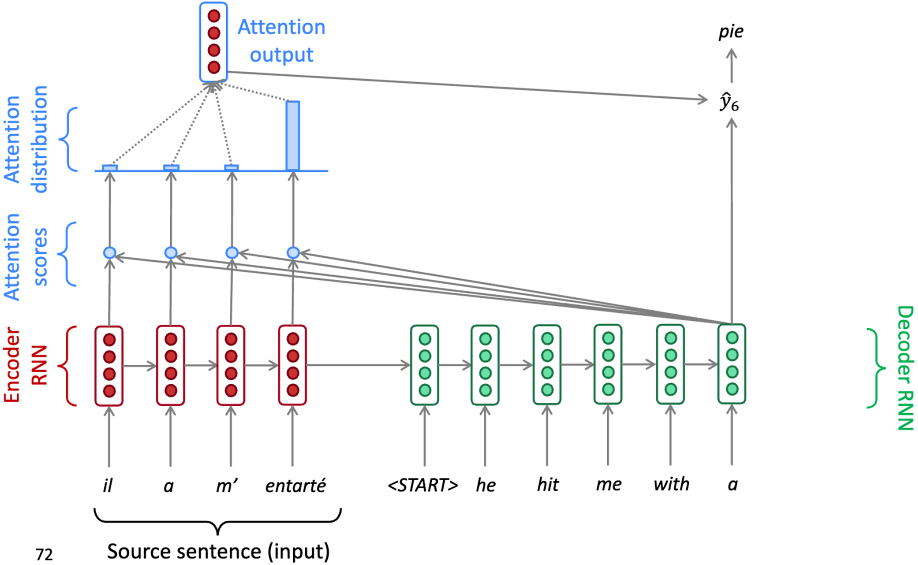
67


Attention output
hit
$\hat{y}_2$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m’
entarté
<START>
he
Source sentence (input)
68
Sometimes we take the attention output from the previous step, and also feed it into the decoder (along with the usual decoder input). We do this in Assignment 4.


Attention output
me
$\hat{y}_3$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
Source sentence (input)
69


Attention output
with
$\hat{y}_4$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
70
Source sentence (input)


Attention output
a
$\hat{y}_5$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
with
Source sentence (input)
71


Attention output
pie
$\hat{y}_6$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
with
a
Source sentence (input)

72

Attention: in equations (Part 1)

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep t, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores e^t for this step:

$$e^t = \left[s_t^T h_1, \ldots, s_t^T h_N \right] \in \mathbb{R}^N$$

- We take *softmax* to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}\left(e^t\right) \in \mathbb{R}^N$$

Attention: in equations (Part 2)

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output a_t

$$a_t = \sum_{i=1}^{N} \alpha_i^t h_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output a_t with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[a_t; s_t] \in \mathbb{R}^{2h}$$

Attention in Neural Machine Translation (NMT)

- **Attention** significantly improves **NMT** performance
  - It's very useful to allow decoder to focus on certain parts of the source
- **Attention** solves the *bottleneck problem*
  - **Attention** allows decoder to look directly at source; bypass bottleneck
- **Attention** helps with *vanishing gradient problem*
  - Provides shortcut to faraway states
- **Attention** provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get (soft) alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself

Summary

- **Neural Machine Translation (NMT)** is a powerful approach to translating text using neural networks.
- The *Encoder-Decoder* architecture is a key component of NMT, where the encoder processes the source sentence and the decoder generates the target sentence.
- **Seq2Seq** is the architecture used in NMT, which encodes the input sequence into a single vector and decodes it one word at a time.
- Attention mechanisms enhance the Seq2Seq model by allowing the decoder to focus on different parts of the input sequence, improving performance and interpretability.