

# Natural Language Processing with Deep Learning

## CS224N/Ling284



Tatsunori Hashimoto

Lecture 6: LSTM RNNs and Neural Machine Translation

(Slides mostly from Chris Manning's 2023 version)

### Long Short-Term Memory RNNs (LSTMs)

- On step  $t$ , there is a **hidden state**  $h^{(t)}$  and a **cell state**  $c^{(t)}$ 
  - Both are vectors length  $n$
  - The cell stores **long-term information**
  - The LSTM can **read**, **erase**, and **write** information from the cell
    - The cell becomes conceptually rather like RAM in a computer
- The selection of which information is erased/written/read is controlled by three corresponding **gates**
  - The gates are also vectors of length  $n$
  - On each timestep, each element of the gates can be **open** (1), **closed** (0), or somewhere in-between
  - The gates are **dynamic**: their value is computed based on the current context

19

### Long Short-Term Memory RNNs (LSTMs)

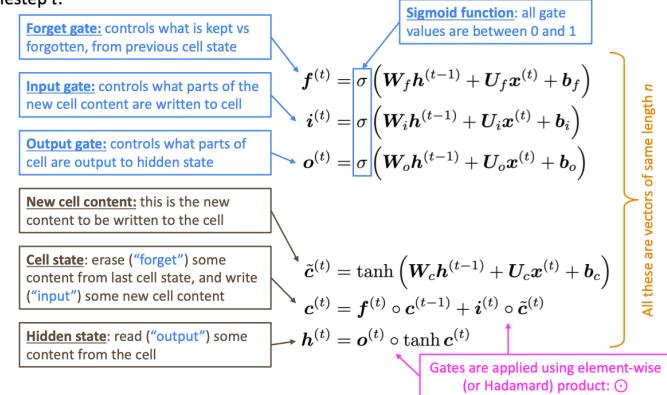
- A type of RNN proposed by Hochreiter and Schmidhuber in 1997 as a solution to the problem of vanishing gradients
  - Everyone cites that paper but really a crucial part of the modern LSTM is from Gers et al. (2000) ❤️
- Only started to be recognized as promising through the work of S's student Alex Graves c. 2006
  - Work in which he also invented CTC (connectionist temporal classification) for speech recognition
- But only really became well-known after Hinton brought it to Google in 2013
  - Following Graves having been a postdoc with Hinton

Hochreiter and Schmidhuber, 1997. Long short-term memory. <https://www.bioinf.jku.at/publications/older/2604.pdf>  
 Gers, Schmidhuber, and Cummins, 2000. Learning to Forget: Continual Prediction with LSTM. <https://dl.acm.org/doi/10.1162/08997660030015015>  
 Graves, Fernandez, Gomez, and Schmidhuber, 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural nets. [https://www.cs.toronto.edu/~graves/cml\\_2006.pdf](https://www.cs.toronto.edu/~graves/cml_2006.pdf)

18

### Long Short-Term Memory (LSTM)

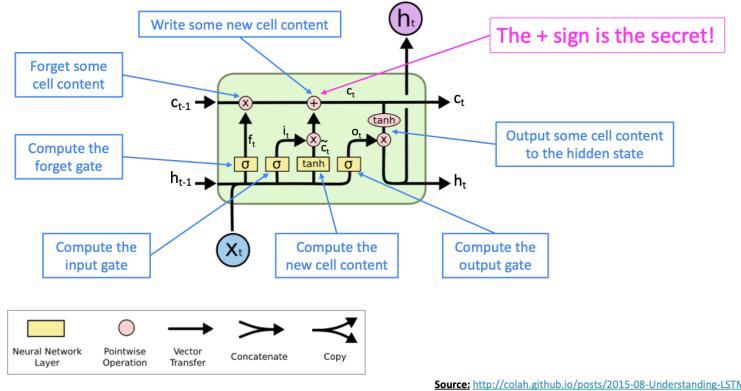
We have a sequence of inputs  $x^{(t)}$ , and we will compute a sequence of hidden states  $h^{(t)}$  and cell states  $c^{(t)}$ . On timestep  $t$ :



20

## Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:



22

## LSTMs: real-world success

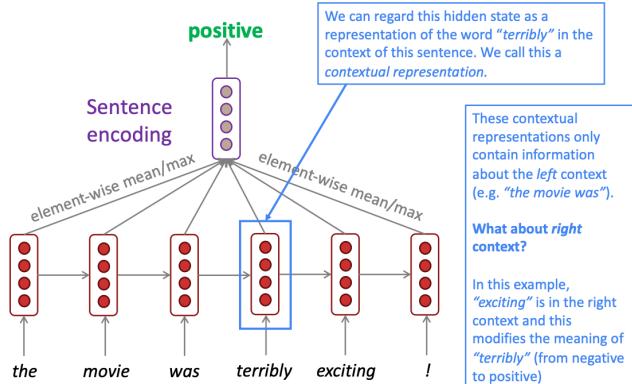
- In 2013–2015, LSTMs started achieving state-of-the-art results
  - Successful tasks include handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language models
  - LSTMs became the dominant approach for most NLP tasks
- Now (2019–2024), Transformers have become dominant for all tasks
  - For example, in WMT (a Machine Translation conference + competition):
    - In WMT 2014, there were 0 neural machine translation systems (!)
    - In WMT 2016, the summary report contains “RNN” 44 times (and these systems won)
    - In WMT 2019: “RNN” 7 times, “Transformer” 105 times

Source: "Findings of the 2016 Conference on Machine Translation (WMT16)", Bojar et al. 2016, <http://www.statmt.org/wmt16/pdf/W16-2301.pdf>  
 Source: "Findings of the 2018 Conference on Machine Translation (WMT18)", Bojar et al. 2018, <http://www.statmt.org/wmt18/pdf/WMT028.pdf>  
 Source: "Findings of the 2019 Conference on Machine Translation (WMT19)", Barraut et al. 2019, <http://www.statmt.org/wmt19/pdf/WMT028.pdf>

26

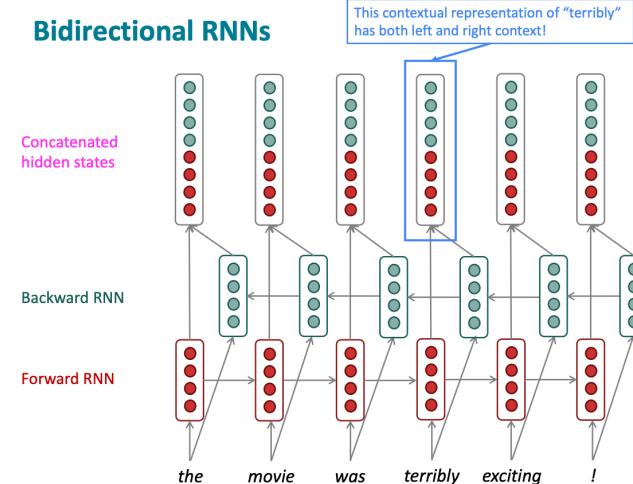
## 4. Bidirectional and Multi-layer RNNs: motivation

Task: Sentiment Classification



32

## Bidirectional RNNs



33

## Bidirectional RNNs

On timestep  $t$ :

This is a general notation to mean “compute one forward step of the RNN” – it could be a simple RNN or LSTM computation.

$$\text{Forward RNN } \vec{h}^{(t)} = \text{RNN}_{\text{FW}}(\vec{h}^{(t-1)}, \mathbf{x}^{(t)})$$

$$\text{Backward RNN } \overleftarrow{h}^{(t)} = \text{RNN}_{\text{BW}}(\overleftarrow{h}^{(t+1)}, \mathbf{x}^{(t)})$$

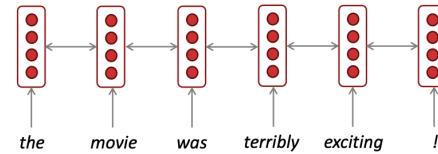
$$\text{Concatenated hidden states } \vec{h}^{(t)} = [\vec{h}^{(t)}; \overleftarrow{h}^{(t)}]$$

Generally, these two RNNs have separate weights

We regard this as “the hidden state” of a bidirectional RNN. This is what we pass on to the next parts of the network.

34

## Bidirectional RNNs: simplified diagram



The two-way arrows indicate bidirectionality and the depicted hidden states are assumed to be the concatenated forwards+backwards states

35

## Bidirectional RNNs

- Note: bidirectional RNNs are only applicable if you have access to the **entire input sequence**
  - They are **not** applicable to Language Modeling, because in LM you *only* have left context available.
- If you do have entire input sequence (e.g., any kind of encoding), **bidirectionality is powerful** (you should use it by default).
- For example, **BERT** (**Bidirectional Encoder Representations from Transformers**) is a **powerful** pretrained contextual representation system **built on bidirectionality**.
  - You will learn more about **transformers**, including BERT, in a couple of weeks!

36

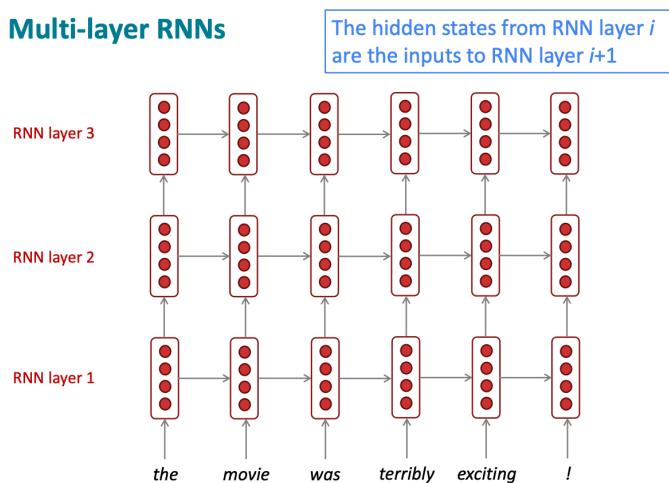
## Multi-layer RNNs

- RNNs are already “deep” on one dimension (they unroll over many timesteps)
- We can also make them “deep” in another dimension by **applying multiple RNNs** – this is a multi-layer RNN.
- This allows the network to compute **more complex representations**
  - The **lower RNNs** should **compute lower-level features** and the **higher RNNs** should **compute higher-level features**.
- Multi-layer RNNs are also called **stacked RNNs**.



37

## Multi-layer RNNs



38

## Multi-layer RNNs in practice

- Multi-layer or stacked RNNs allow a network to compute **more complex representations**
  - they work better than just have one layer of high-dimensional encodings!
  - The **lower RNNs** should **compute lower-level features** and the **higher RNNs** should **compute higher-level features**.
- **High-performing RNNs are usually multi-layer** (but aren't as deep as convolutional or feed-forward networks)
- For example: In a 2017 paper, Britz et al. find that for Neural Machine Translation, **2 to 4 layers** is best for the encoder RNN, and **4 layers** is best for the decoder RNN
  - Often 2 layers is a lot better than 1, and 3 might be a little better than 2
  - Usually, **skip-connections/dense-connections** are needed to train deeper RNNs (e.g., **8 layers**)
- **Transformer-based networks** (e.g., BERT) are usually deeper, like **12 or 24 layers**.
  - You will learn about Transformers later; they have a lot of skipping-like connections

"Massive Exploration of Neural Machine Translation Architectures", Britz et al, 2017. <https://arxiv.org/pdf/1703.03906.pdf>

39

## 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*

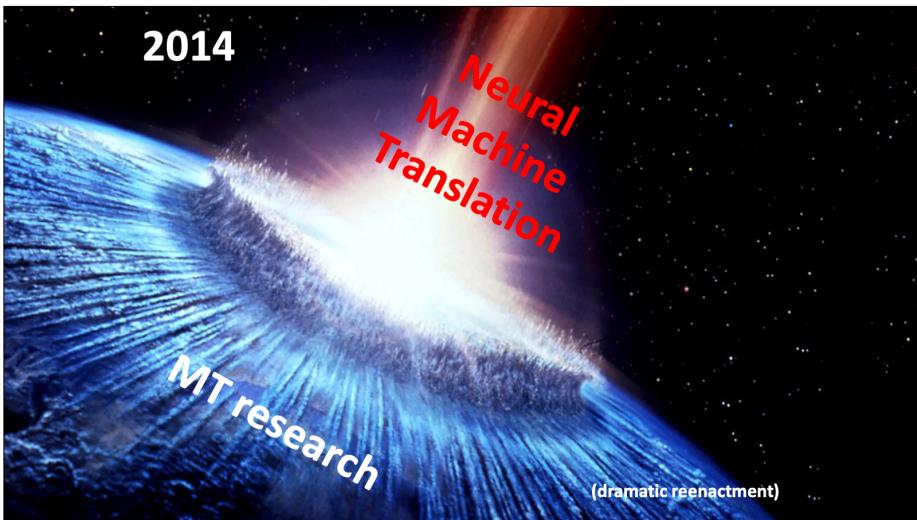
– Rousseau

40

## 1990s–2010s: Statistical Machine Translation

- SMT was a **huge research field**
- The best systems were **extremely complex**
  - Hundreds of important details
- Systems had many **separately-designed subcomponents**
  - Lots of **feature engineering**
    - Need to design features to capture particular language phenomena
  - Required compiling and maintaining **extra resources**
    - Like tables of equivalent phrases
  - Lots of **human effort** to maintain
    - Repeated effort for each language pair!

45



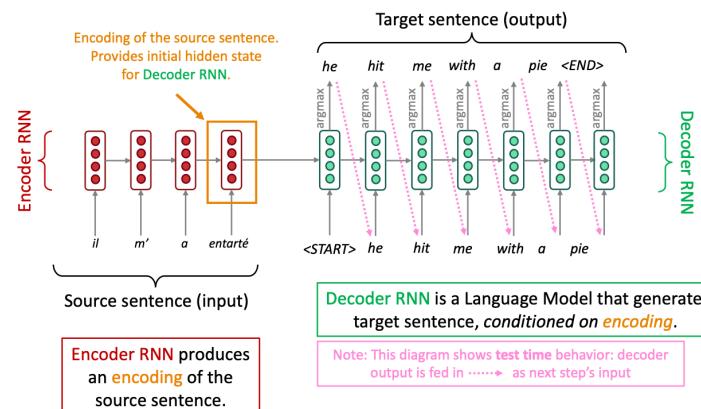
## What is Neural Machine Translation?

- Neural Machine Translation (NMT) is a way to do Machine Translation with a *single end-to-end neural network*
- The neural network architecture is called a *sequence-to-sequence* model (aka *seq2seq*) and it involves *two RNNs*

47

## Neural Machine Translation (NMT)

The sequence-to-sequence model



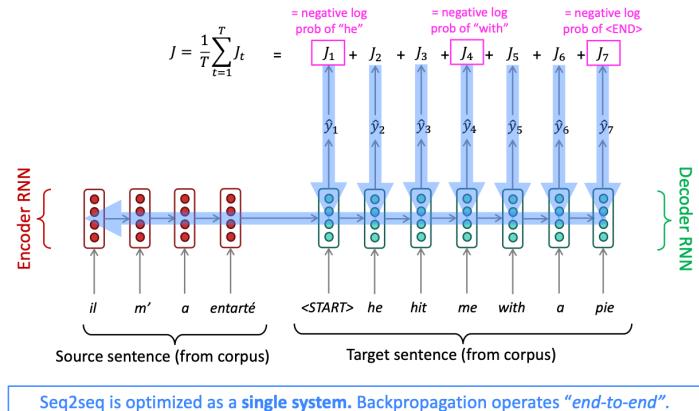
48

## Sequence-to-sequence is versatile!

- The general notion here is an *encoder-decoder* model
  - One neural network takes input and produces a neural representation
  - Another network produces output based on that neural representation
  - If the input and output are sequences, we call it a seq2seq model
- 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)

49

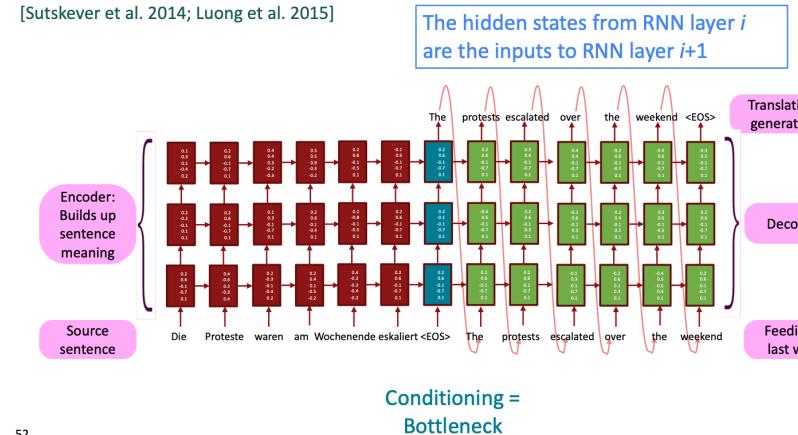
## Training a Neural Machine Translation system



51

## Multi-layer deep encoder-decoder machine translation net

[Sutskever et al. 2014; Luong et al. 2015]



52

## NMT: the first big success story of NLP Deep Learning

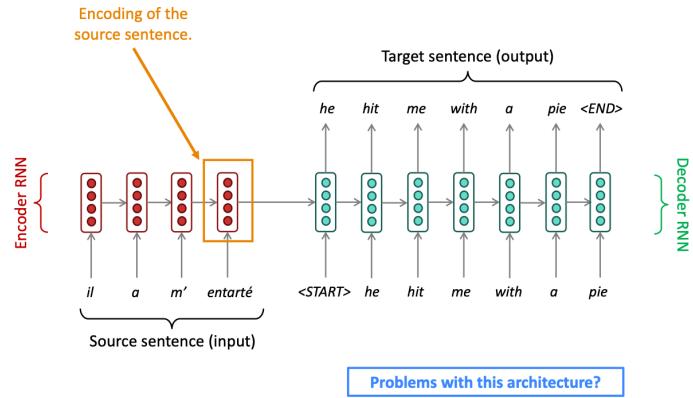
Neural Machine Translation went from a fringe research attempt in 2014 to the leading standard method in 2016

- 2014: First seq2seq paper published [Sutskever et al. 2014]
- 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 small groups of engineers in a few months

4

## Attention Is All You Need

## The final piece: the bottleneck problem in RNNs



5

## Attention

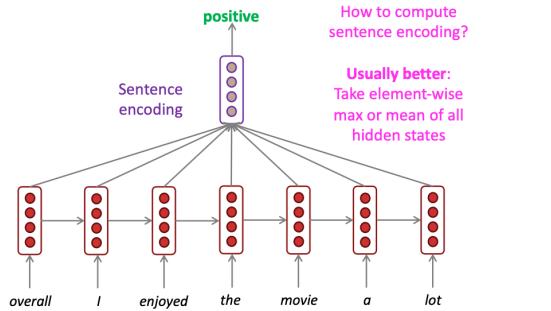
- Attention provides a solution to the bottleneck problem.
- **Core idea:** on each step of the decoder, use *direct connection to the encoder* to focus on a particular part of the source sequence



- First, we will show via diagram (no equations), then we will show with equations

9

## The starting point: mean-pooling for RNNs

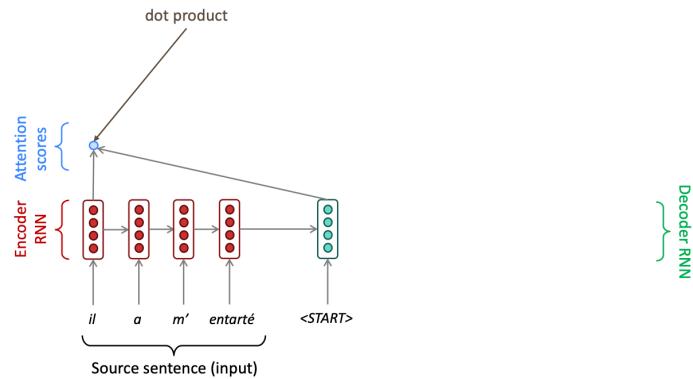


- Starting point: a very basic way of 'passing information from the encoder' is to average

10

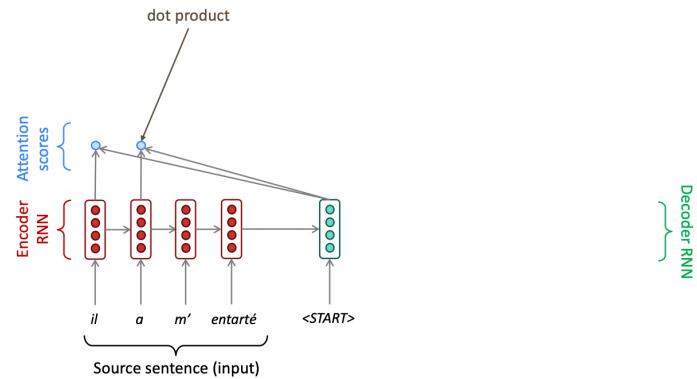
## Sequence-to-sequence with attention

**Core idea:** on each step of the decoder, use *direct connection to the encoder* to focus on a particular part of the source sequence



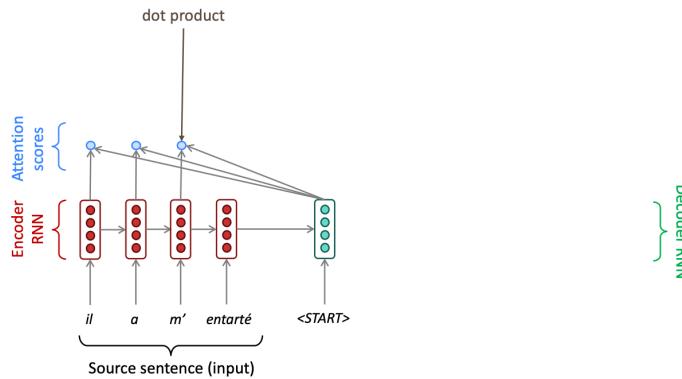
12

## Sequence-to-sequence with attention



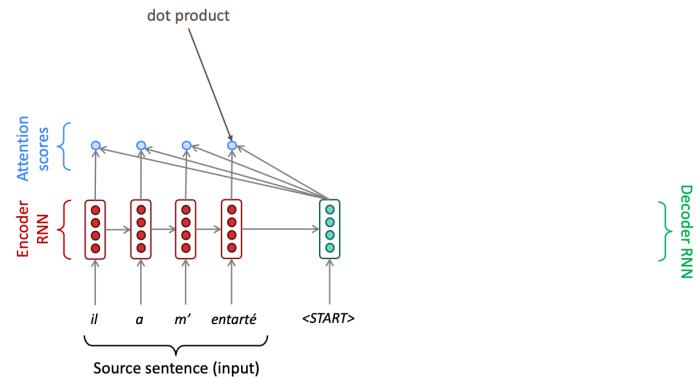
13

## Sequence-to-sequence with attention



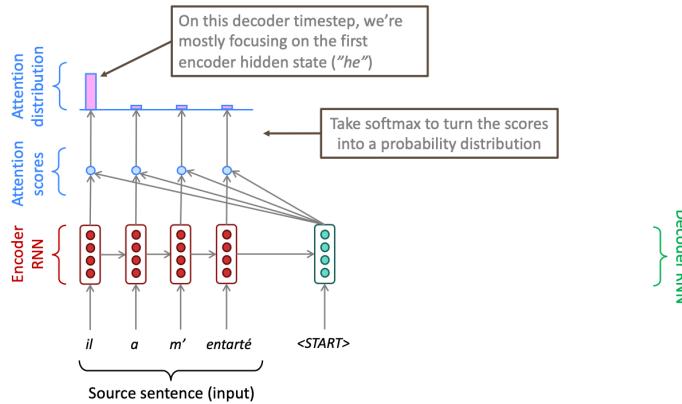
14

## Sequence-to-sequence with attention



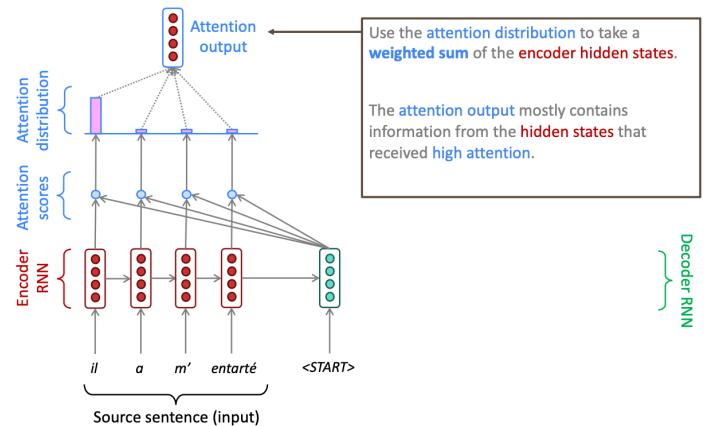
15

## Sequence-to-sequence with attention



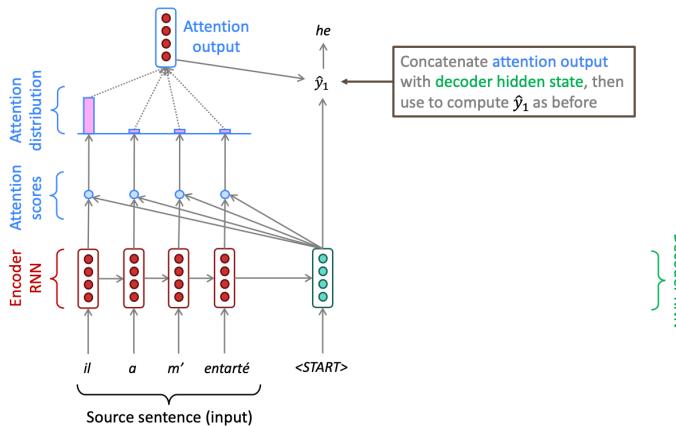
16

## Sequence-to-sequence with attention



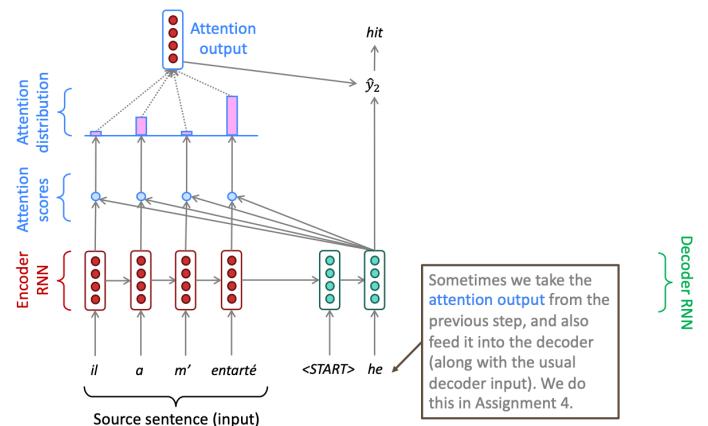
17

## Sequence-to-sequence with attention



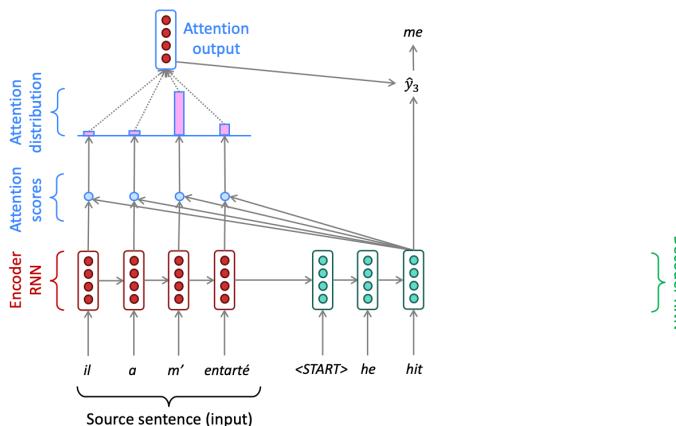
18

## Sequence-to-sequence with attention



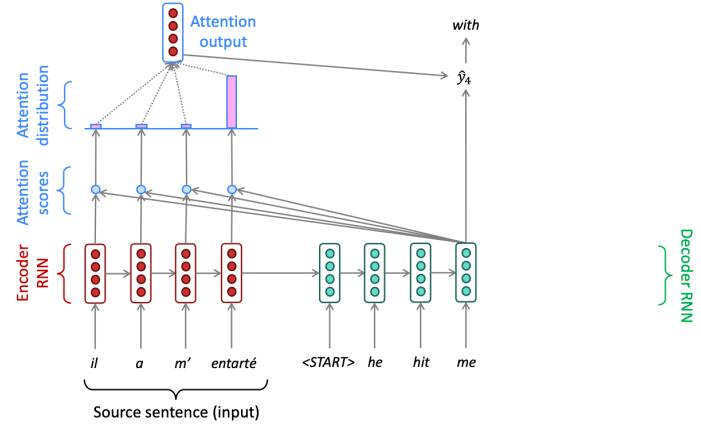
19

## Sequence-to-sequence with attention



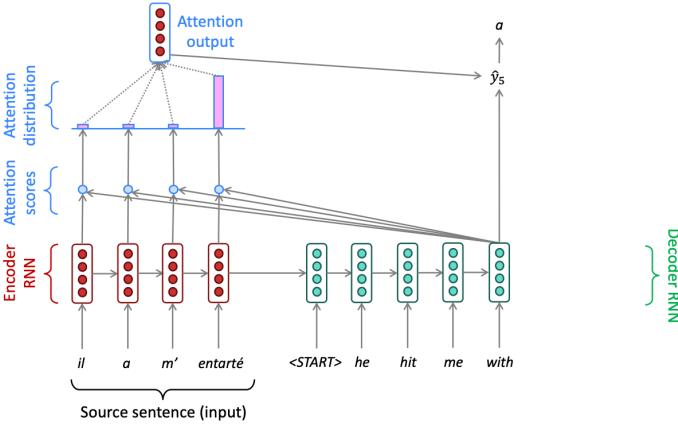
20

## Sequence-to-sequence with attention



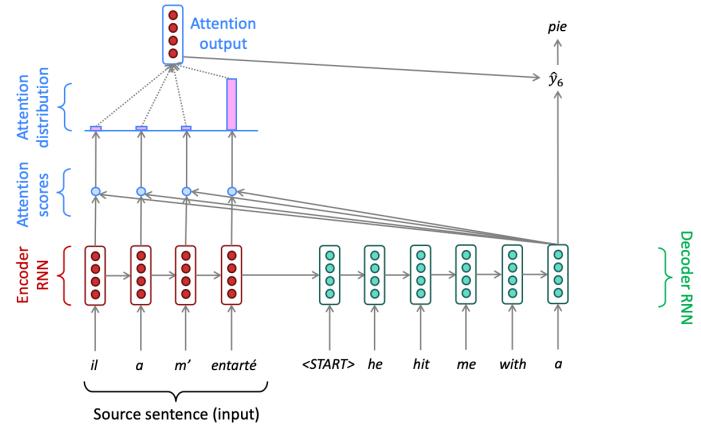
21

## Sequence-to-sequence with attention



22

## Sequence-to-sequence with attention



23

## Attention: in equations

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

$$e^t = [s_t^T h_1, \dots, s_t^T h_N] \in \mathbb{R}^N$$
- We take softmax to get the attention distribution  $\alpha^t$  for this step (this is a probability distribution and sums to 1)  

$$\alpha^t = \text{softmax}(e^t) \in \mathbb{R}^N$$
- We use  $\alpha^t$  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  

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

24

## Attention is great!

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention provides a more “human-like” model of the MT process
  - You can look back at the source sentence while translating, rather than needing to remember it all
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with the vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we 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

26



il	e	z	me	with	a	pe
a						
m'						
entarté						

There are multiple ways to do this

## There are several attention variants

- We have some values  $h_1, \dots, h_N \in \mathbb{R}^{d_1}$  and a query  $s \in \mathbb{R}^{d_2}$
  - Attention always involves:
    1. Computing the **attention scores**  $e \in \mathbb{R}^N$
    2. Taking softmax to get **attention distribution**  $\alpha$ :
$$\alpha = \text{softmax}(e) \in \mathbb{R}^N$$
  - 3. Using attention distribution to take weighted sum of values:
- $$a = \sum_{i=1}^N \alpha_i h_i \in \mathbb{R}^{d_1}$$
- thus obtaining the **attention output**  $a$  (sometimes called the **context vector**)

27

## Attention variants

You'll think about the relative advantages/disadvantages of these in Assignment 4!

There are several ways you can compute  $e \in \mathbb{R}^N$  from  $h_1, \dots, h_N \in \mathbb{R}^{d_1}$  and  $s \in \mathbb{R}^{d_2}$ :

- Basic dot-product attention:  $e_i = s^T h_i \in \mathbb{R}$ 
  - Note: this assumes  $d_1 = d_2$ . This is the version we saw earlier.
- Multiplicative attention:  $e_i = s^T W h_i \in \mathbb{R}$  [Luong, Pham, and Manning 2015]
  - Where  $W \in \mathbb{R}^{d_2 \times d_1}$  is a weight matrix. Perhaps better called “bilinear attention”
- Reduced-rank multiplicative attention:  $e_i = s^T (\mathbf{U}^T \mathbf{V}) h_i = (\mathbf{U}s)^T (\mathbf{V} h_i)$ 
  - For low rank matrices  $\mathbf{U} \in \mathbb{R}^{k \times d_2}, \mathbf{V} \in \mathbb{R}^{k \times d_1}, k \ll d_1, d_2$
- Additive attention:  $e_i = v^T \tanh(\mathbf{W}_1 h_i + \mathbf{W}_2 s) \in \mathbb{R}$  [Bahdanau, Cho, and Bengio 2014]
  - Where  $\mathbf{W}_1 \in \mathbb{R}^{d_3 \times d_1}, \mathbf{W}_2 \in \mathbb{R}^{d_3 \times d_2}$  are weight matrices and  $v \in \mathbb{R}^{d_3}$  is a weight vector.
  - $d_3$  (the attention dimensionality) is a hyperparameter
  - “Additive” is a weird/bad name. It’s really using a feed-forward neural net layer.

28

More information: “Deep Learning for NLP Best Practices”, Ruder, 2017, <http://ruder.io/deep-learning-nlp-best-practices/index.html#attention>  
“Massive Exploration of Neural Machine Translation Architectures”, Britz et al, 2017, <https://arxiv.org/pdf/1703.03906.pdf>

## Attention is a general Deep Learning technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)
- More general definition of attention:
  - Given a set of vector **values**, and a vector **query**, **attention** is a technique to compute a weighted sum of the values, dependent on the query.
- We sometimes say that the **query attends to the values**.
- For example, in the seq2seq + attention model, each decoder hidden state (query) attends to all the encoder hidden states (values).

29

## Attention is a **general** Deep Learning technique

- **More general definition of attention:**

- Given a set of vector *values*, and a vector *query*, **attention** is a technique to compute a weighted sum of the values, dependent on the query.

### Intuition:

- The weighted sum is a *selective summary* of the information contained in the values, where the query determines which values to focus on.
- Attention is a way to obtain a *fixed-size representation of an arbitrary set of representations* (the values), dependent on some other representation (the query).

### Upshot:

- Attention has become the powerful, flexible, general way pointer and memory manipulation in all deep learning models. A new idea from after 2010! From NMT!