Sequence-to-Sequence and Attention in the context of Neural Machine Translation

Marta R. Costa-jussà
Universitat Politècnica de Catalunya

Adapted on slides from Cristina España i Bonet
and Abigail See and Christopher Manning, Stanford University, Stanford University, adapted from CS224n Winter 2019 slides: Lecture 6 (RNN), Lecture 8 (NMT) and Lecture 9 (Final Projects)

Background

1. Recurrent Neural Networks/ Recurrent Language Models

2. SMT concepts

Recurrent Language Models

A fixed-window neural Language Model

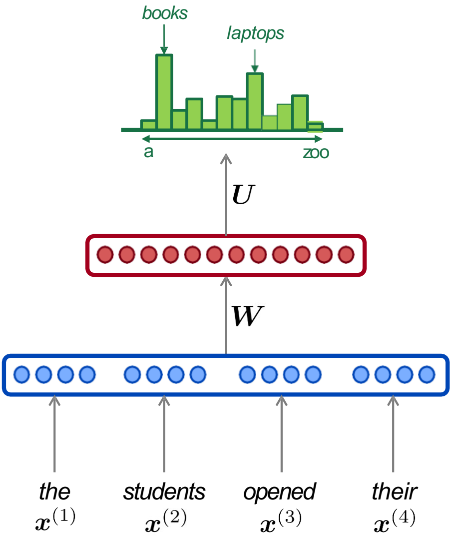
Improvements over *n*-gram LM:

- No sparsity problem
- Don't need to store all observed *n*-grams

Remaining **problems**:

- Fixed window is too small
- Enlarging window enlarges $\boldsymbol{W}$
- Window can never be large enough!
- $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ are multiplied by completely different weights in $\boldsymbol{W}$. No symmetry in how the inputs are processed.

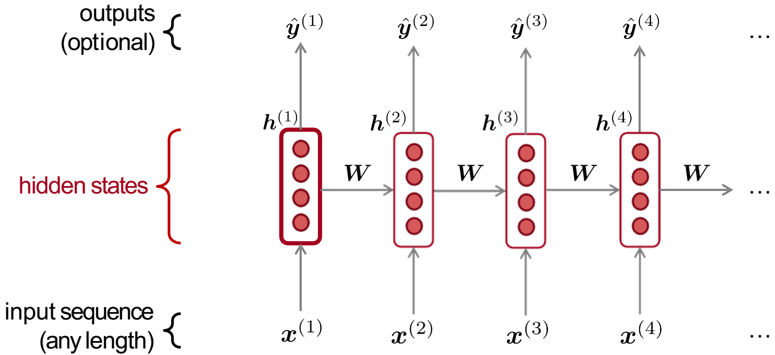
We need a neural architecture that can process *any length input*



Recurrent Neural Networks (RNN)

A family of neural architectures

Core idea: Apply the same weights W *repeatedly*



A RNN Language Model

$$\hat{\boldsymbol{y}}^{(4)} = P(\boldsymbol{x}^{(5)}|\text{the students opened their})$$

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

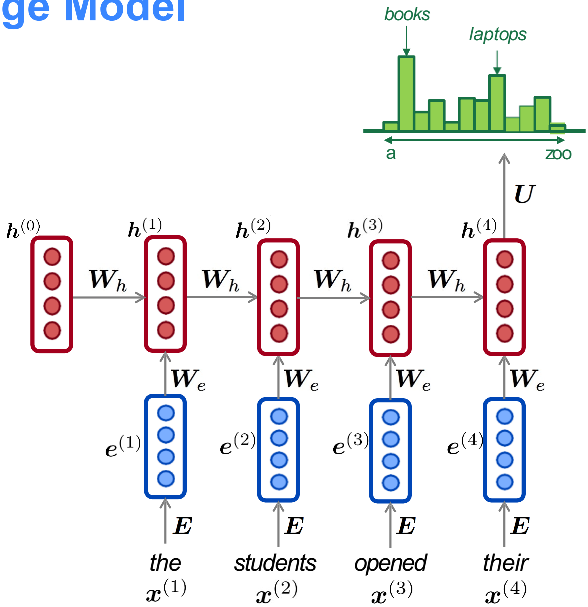
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

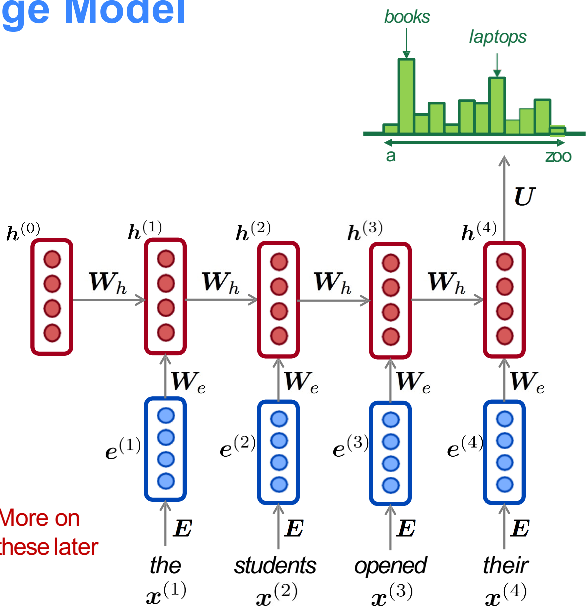
$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



***Note**: this input sequence could be much longer, but this slide doesn't have space!*

A RNN Language Model

$$\hat{\boldsymbol{y}}^{(4)} = P(\boldsymbol{x}^{(5)}|\text{the students opened their})$$



RNN **Advantages**:

- Can process any length input
- Computation for step *t* can (in theory) use information from many steps back
- Model size doesn't increase for longer input
- Same weights applied on every timestep, so there is symmetry in how inputs are processed.

RNN **Disadvantages**:

- Recurrent computation is slow
- In practice, difficult to access information from many steps back

More on these later

Training a RNN Language Model

- Get a big corpus of text which is a sequence of words $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{\boldsymbol{y}}^{(t)}$ *for every step t.*
  - i.e. predict probability dist of *every word*, given words so far

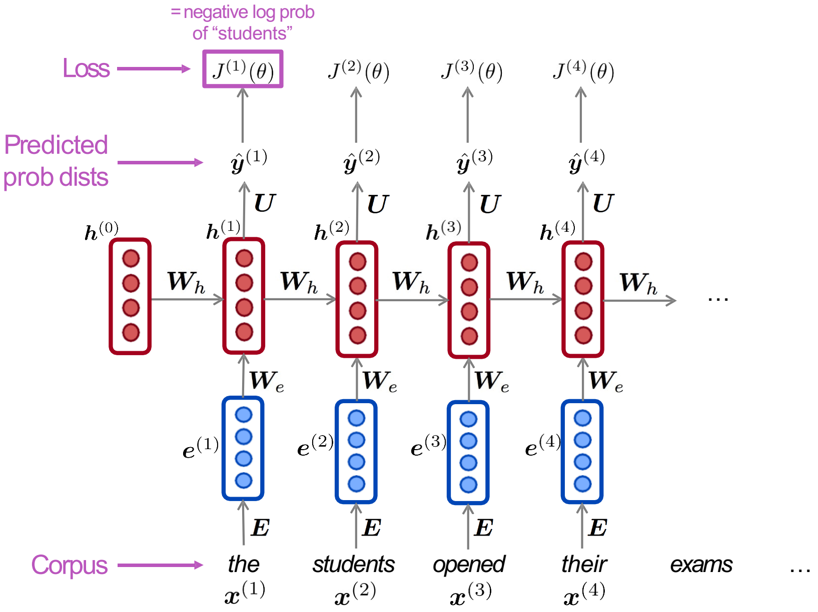
- Loss function on step *t* is cross-entropy between predicted probability distribution $\hat{\boldsymbol{y}}^{(t)}$, and the true next word $\boldsymbol{y}^{(t)}$ (one-hot for $\boldsymbol{x}^{(t+1)}$):

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

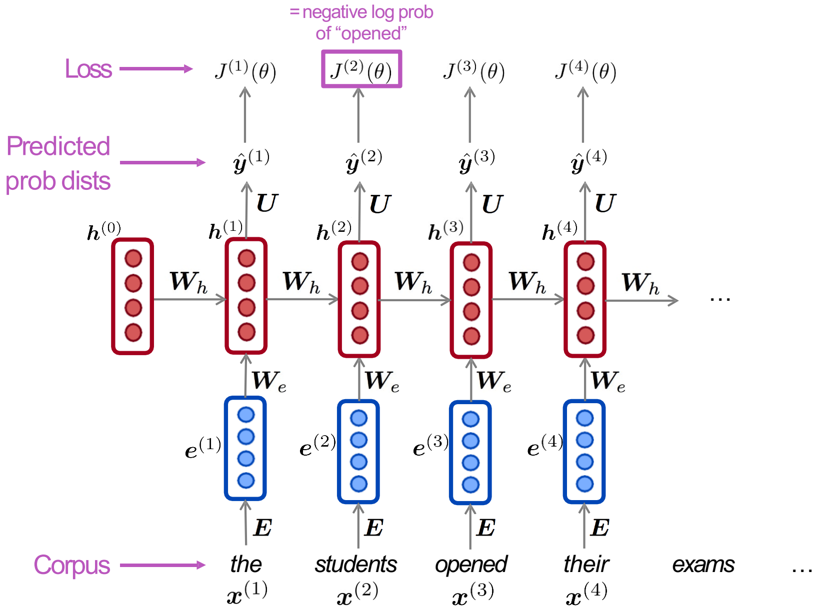
- Average this to get overall loss for entire training set:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

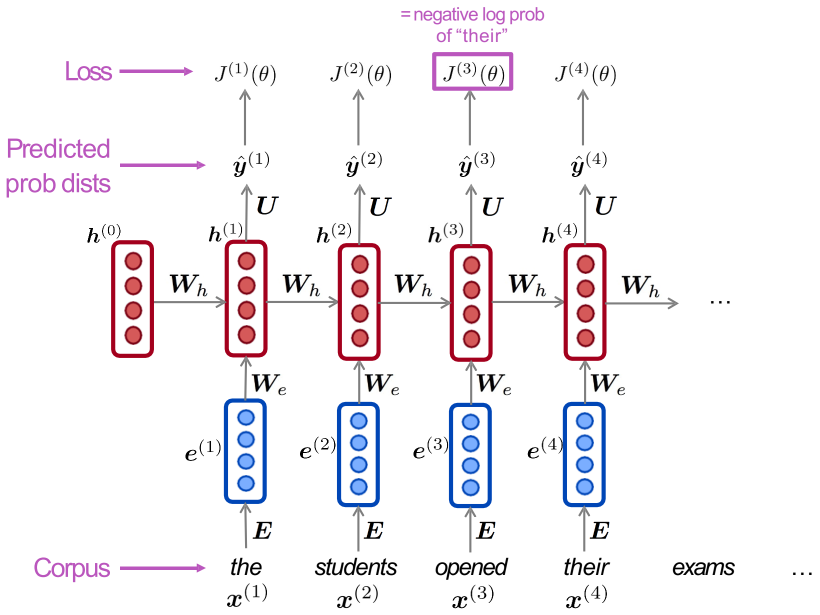
Training a RNN Language Model



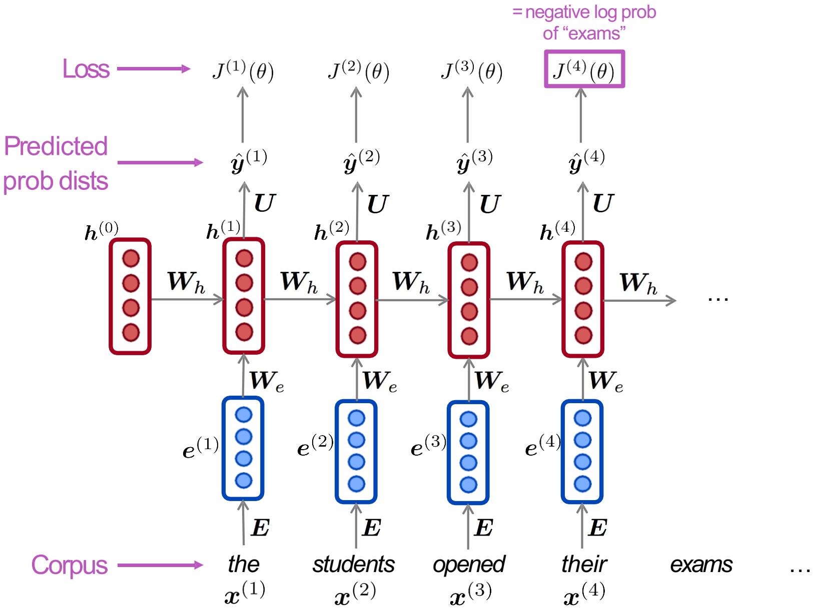
Training a RNN Language Model



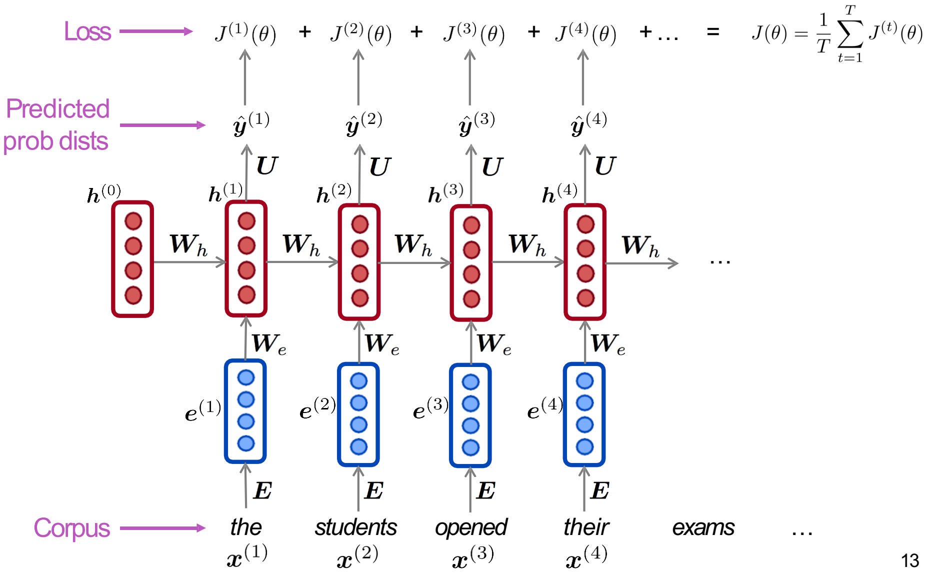
Training a RNN Language Model



Training a RNN Language Model



Training a RNN Language Model



Statistical Machine Translation

1990s-2010s: Statistical Machine Translation

- Core idea: Learn a probabilistic model from data
- Suppose we're translating French → English.
- We want to find best English sentence *y*, given French sentence *x*

$$\text{argmax}_y P(y|x)$$

- Use Bayes Rule to break this down into two components to be learnt separately:

$$= \text{argmax}_y \underbrace{P(x|y)}\underbrace{P(y)}$$

Translation Model

Models how words and phrases should be translated (*fidelity*). Learnt from parallel data.

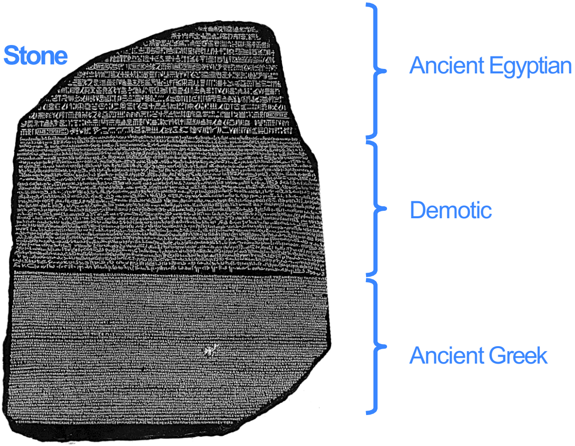
Language Model

Models how to write good English (*fluency*). Learnt from monolingual data.

1990s-2010s: Statistical Machine Translation

- Question: How to learn translation model $P(x|y)$?
- First, need large amount of parallel data
  (e.g. pairs of human-translated French/English sentences)

The Rosetta Stone



Learning alignment for SMT

- Question: How to learn translation model $P(x|y)$ from the parallel corpus?

- Break it down further: we actually want to consider

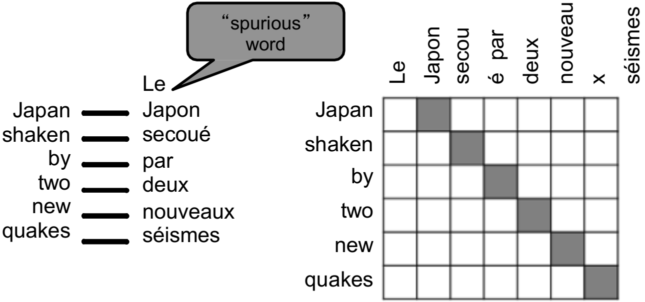
$$P(x, a|y)$$

  where *a* is the alignment, i.e. word-level correspondence between French sentence *x* and English sentence *y*

What is alignment?

Alignment is the correspondence between particular words in the translated sentence pair.

- Note: Some words have no counterpart



Examples from: "The Mathematics of Statistical Machine Translation: Parameter Estimation", Brown et al, 1993. http://www.aclweb.org/anthology/J93-2003

Learning alignment for SMT

- We learn $P(x, a|y)$ as a combination of many factors, including:
  - Probability of particular words aligning (also depends on position in sent)
  - Probability of particular words having particular fertility (number of corresponding words)
  - etc.

Decoding for SMT

$$\underbrace{\text{argmax}_y}_{\text{Question: How to compute this argmax?}} \underbrace{P(x|y)}_{\text{Translation Model}} \underbrace{P(y)}_{\text{Language Model}}$$

- We could enumerate every possible *y* and calculate the probability? → Too expensive!
- **Answer:** Use a heuristic search algorithm to search for the best translation, discarding hypotheses that are too low-probability
- This process is called *decoding*

MT Evaluation: BLEU Evaluation Metric

(Papineni etal, ACL-2002)

Reference (human) translation:

The U.S. island of Guam is maintaining a high state of alert after the Guam airport and its offices both received an e-mail from someone calling himself the Saudi Arabian Osama bin Laden and threatening a biological/chemical attack against public places such as the airport .

Machine translation:

The American [?] international airport and its the office all receives one calls self the sand Arab rich business [?] and so on electronic mail , which sends out ; The threat will be able after public place and so on the airport to start the biochemistry attack , [?] highly alerts after the maintenance.

- N-gram precision (score is between 0 & 1)
  - What percentage of machine n-grams can be found in the reference translation?
    - An n-gram is an sequence of n words
  - Not allowed to match same portion of reference translation twice at a certain n-gram level (two MT words *airport* are only correct if two reference words *airport;* can't cheat by typing out "the the the the the")
  - Do count unigrams also in a bigram for unigram precision, etc.
- Brevity Penalty
  - Can't just type out single word "the" (precision 1.0!)
- It was thought quite hard to "game" the system (i.e., to find a way to change machine output so that BLEU goes up, but quality doesn't)

Today

1. Sequence-to-sequence

2. Attention

in the context of Neural Machine Translation

Microsoft

Translator Home Solutions Products Languages Blog Support More

Neural Machine Translation reaches historic milestone: human parity for Chinese to English translations

Rate this article

Microsoft Translator March 14, 2018

Share 12 7 in 0 0

Achieving Human Parity on Automatic Chinese to English News Translation

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou

Microsoft AI & Research

Abstract

Machine translation has made rapid advances in recent years. Millions of people are using it today in online translation systems and mobile applications in order to communicate across language barriers. The question naturally arises whether such systems can approach or achieve parity with human translations. In this paper, we first address the problem of how to define and accurately measure human parity in translation. We then describe Microsoft's machine translation system and measure the quality of its translations on the widely used WMT 2017 news translation task from Chinese to English. We find that our latest neural machine translation system has reached a new state-of-the-art, and that the translation quality is at human parity when compared to professional human translations. We also find that it significantly exceeds the quality of crowd-sourced non-professional translations.

1 Introduction

Recent years have seen human performance levels reached or surpassed in tasks ranging from games such as Go [32] to classification of images in ImageNet [20] to conversational speech recognition on the Switchboard task [49].

In the area of machine translation, we have seen dramatic improvements in quality with the advent of attentional encoder-decoder neural networks [34, 3, 38]. However, translation quality continues to vary a great deal across language pairs, domains, and genres, more or less in direct

Sequence-to-sequence
RNN, BiRNN

Neural Machine Translation: Basic Model

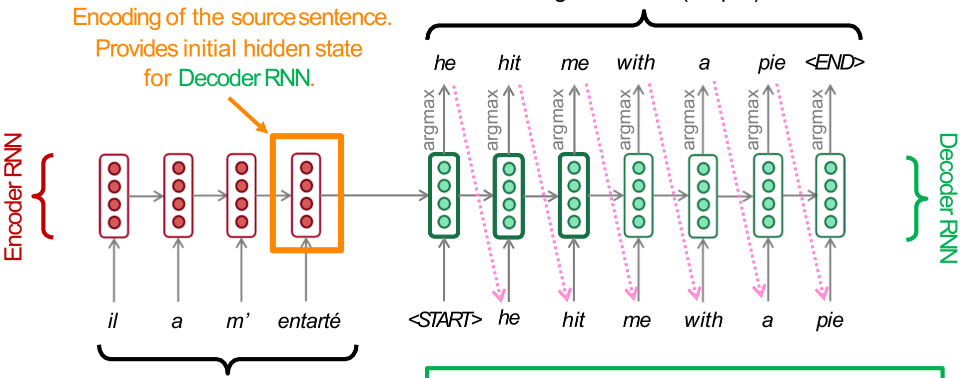
The Encoder–Decoder Model

encodes a sequence of word vectors into a fixed-sized context vector

decodes the fixed-sized vector back into a variable-length sequence

Neural Machine Translation

The sequence-to-sequence model



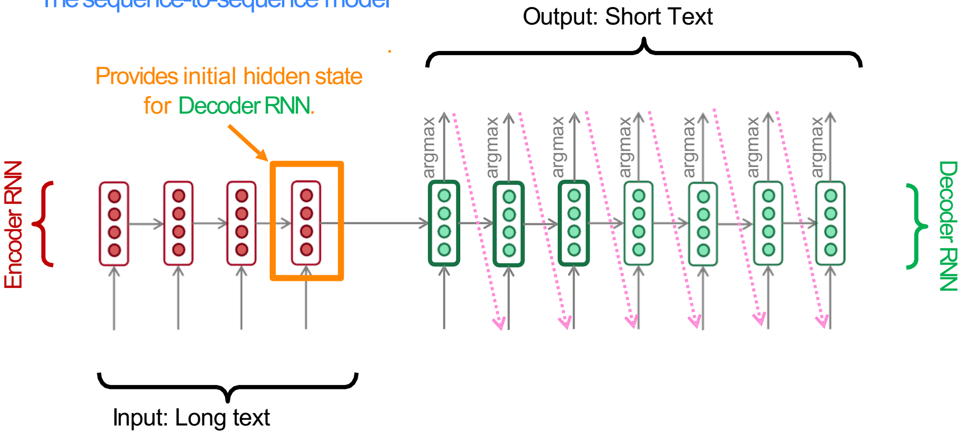
Decoder RNN is a Language Model that generates target sentence, *conditioned on* encoding.

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

Sequence-to-Sequence is versatile

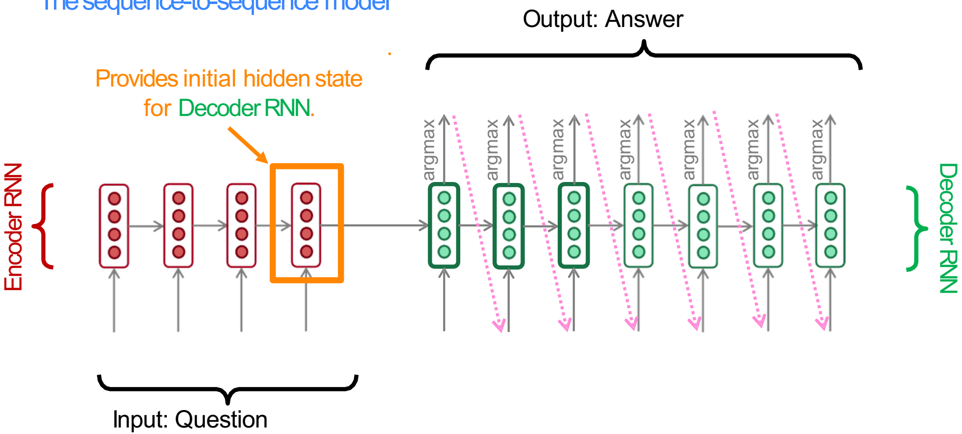
The sequence-to-sequence model



Summarization

Sequence-to-Sequence is versatile

The sequence-to-sequence model



Question-Answering

Sequence-to-Sequence is versatile

- Sequence-to-sequence is useful for *more than just MT*

- Many more NLP tasks:
  - Parsing (input text → output parse as sequence)
  - Code generation (natural language → Python code)

- Other Speech or Image tasks:
  - Speech recognition (speech utterance → transcription)
  - Image captioning (image → caption)

- ...

Sequence-to-Sequence is a conditional LM

- 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*

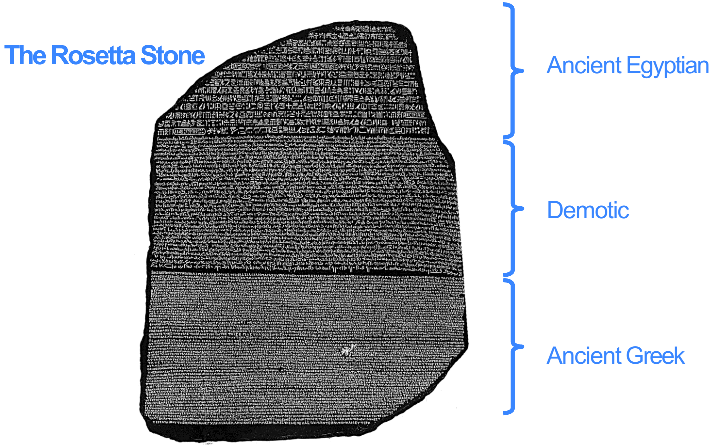
- NMT directly calculates $P(y|x)$:

$$P(y|x) = P(y_1|x)\, P(y_2|y_1, x)\, P(y_3|y_1, y_2, x) \ldots \underbrace{P(y_T|y_1, \ldots, y_{T-1}, x)}_{\text{Probability of next target word, given target words so far and source sentence } x}$$

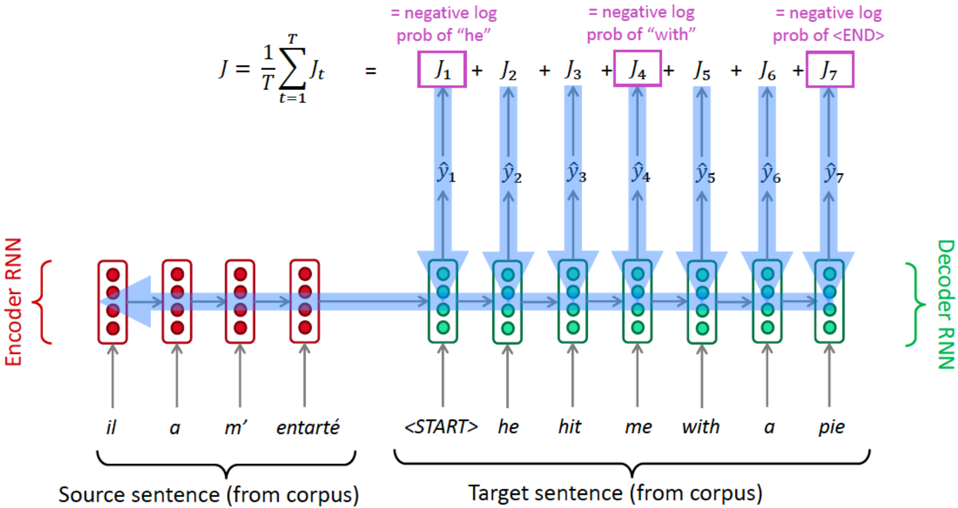
- __**Question**__: How to train a NMT system?
- __**Answer**__: Get a big parallel corpus…

Parallel corpus

- Need large amount of parallel data $P(x|y)$
  (e.g. pairs of human-translated French/English sentences)



Training a Neural Machine Translation System

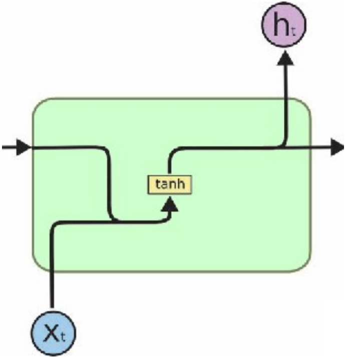


Seq2seq is optimized as a **single system.**
Backpropagation operates "*end-to-end*".

RNN Cells

- RNN cells can be any kind

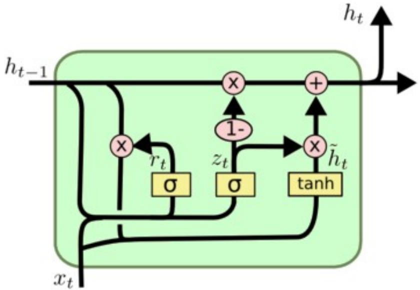
tanh



RNN Cells

- RNN cells can be any kind

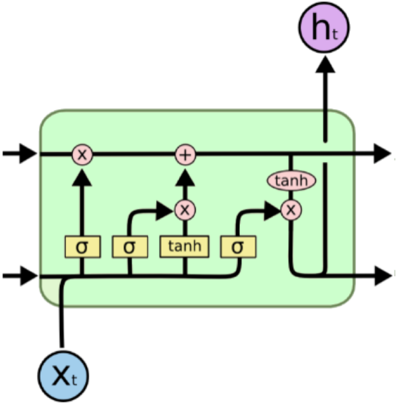
  GRU, Gated Recurrent Unit



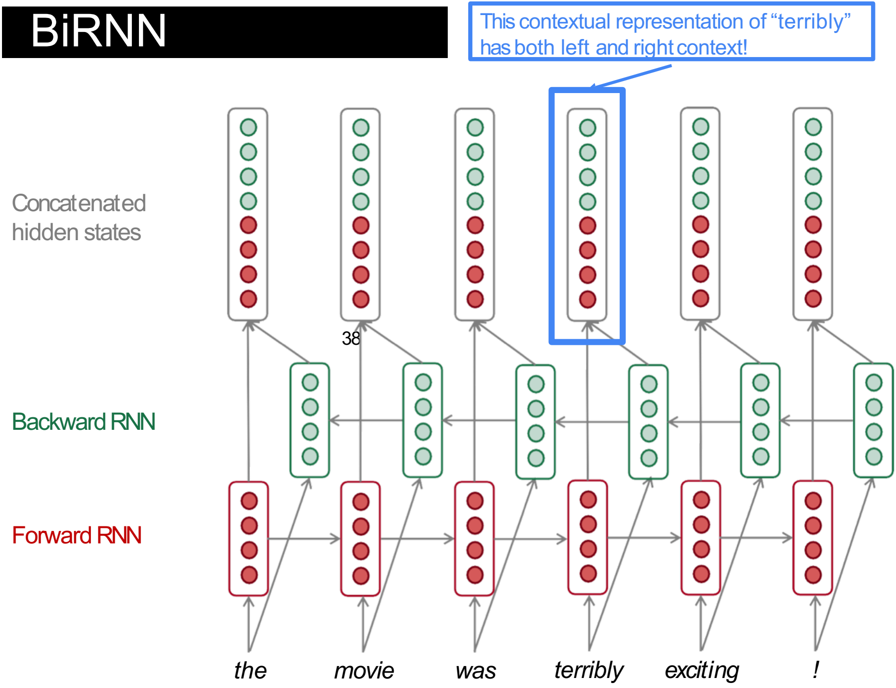
RNN Cells

- RNN cells can be any kind

LSTM, Long Short Term Memory Networks



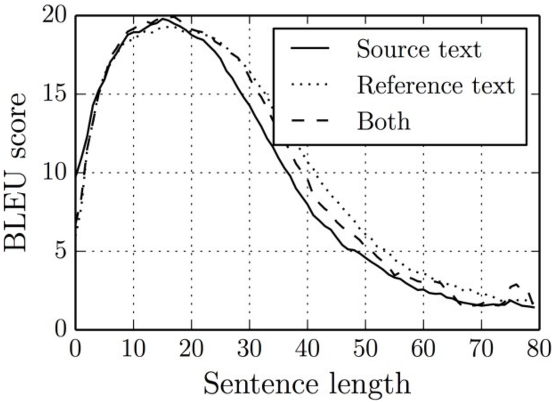
BiRNN



This contextual representation of "terribly" has both left and right context!
Concatenated hidden states
Backward RNN
Forward RNN
the
movie
was
terribly
exciting
!

Limitations in performance

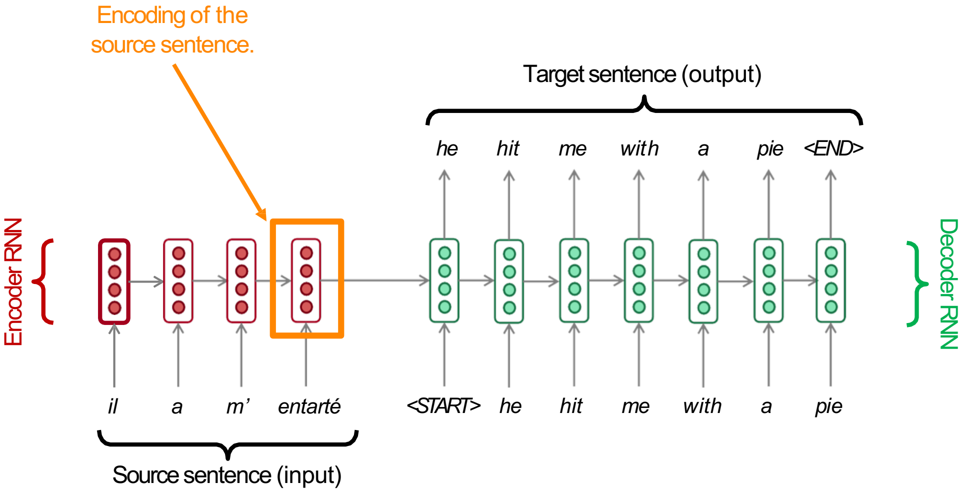
- Performance drops with long sentences



[Cho et al., 2014]

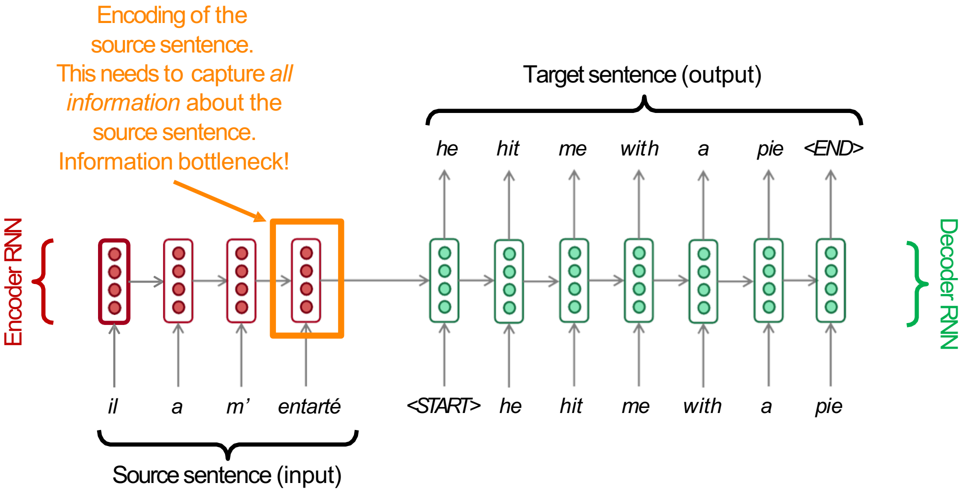
Attention

RNN architecture



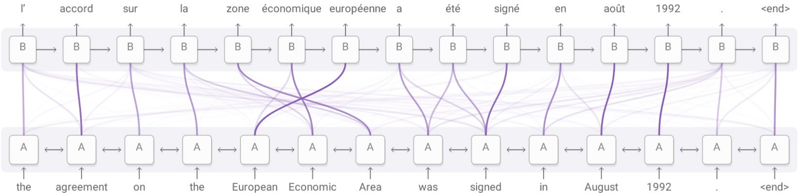
Problems with this architecture?

RNN architecture



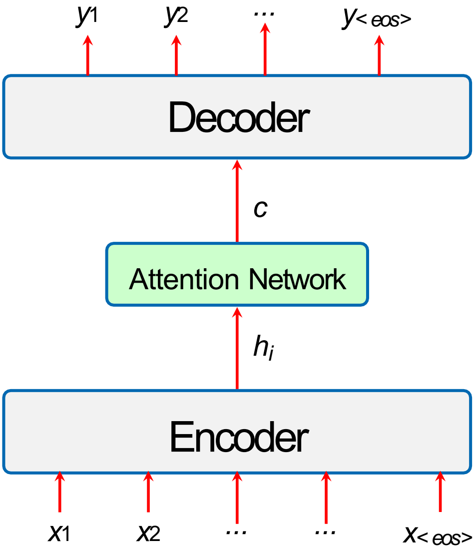
RNN architecture with attention

- Intuition: Not all the words contribute equally for a translation
- Let's weight! (weights, softmaxs, nns...)



https://distill.pub/2016/augmented-rnns

RNN architecture with attention

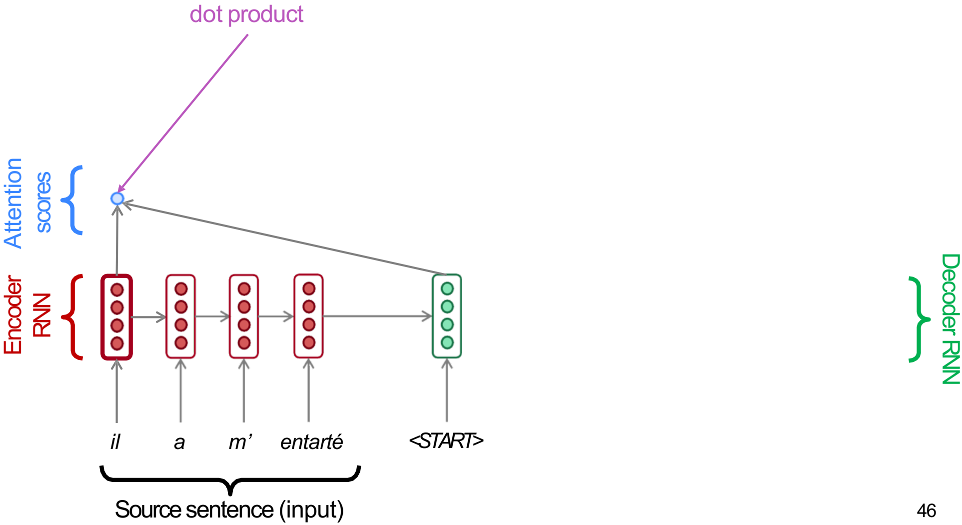


RNN architecture with attention

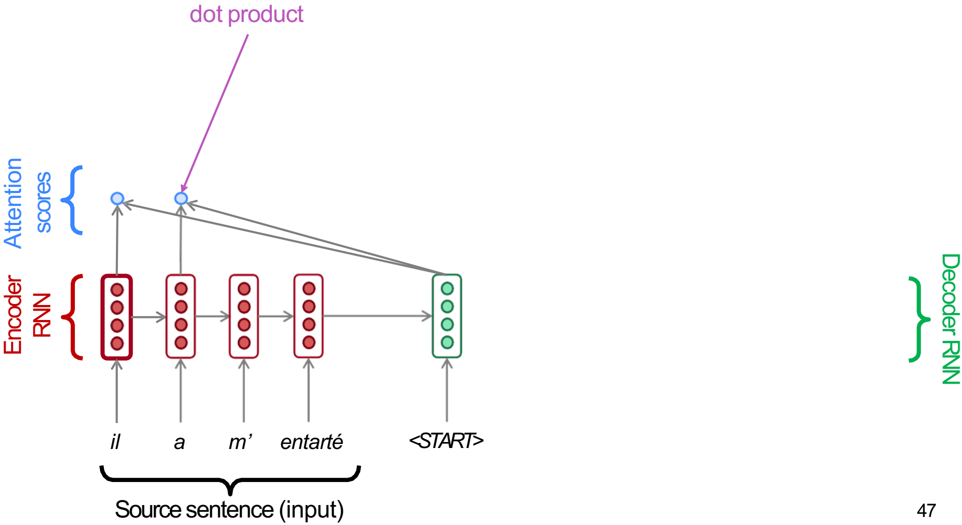
- **Attention** provides a solution to the bottleneck problem.
- Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence

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

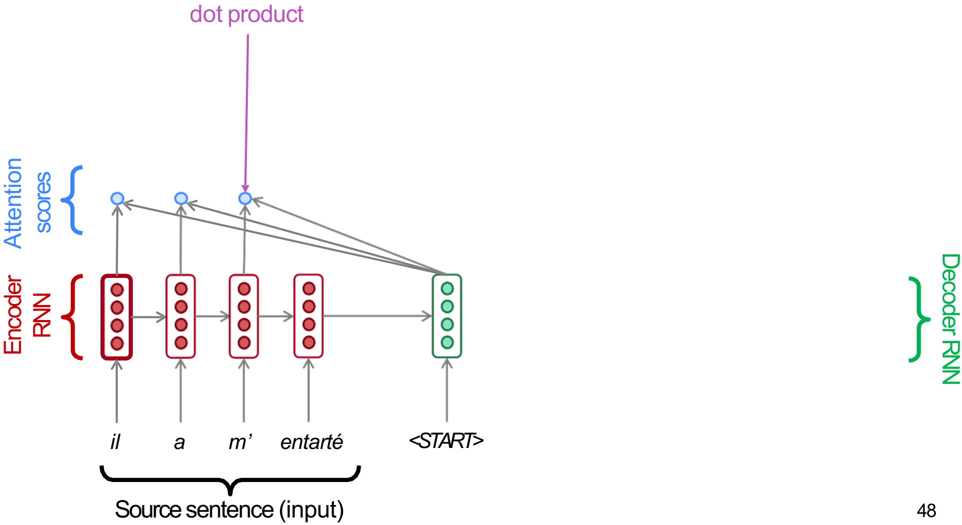
RNN architecture with attention



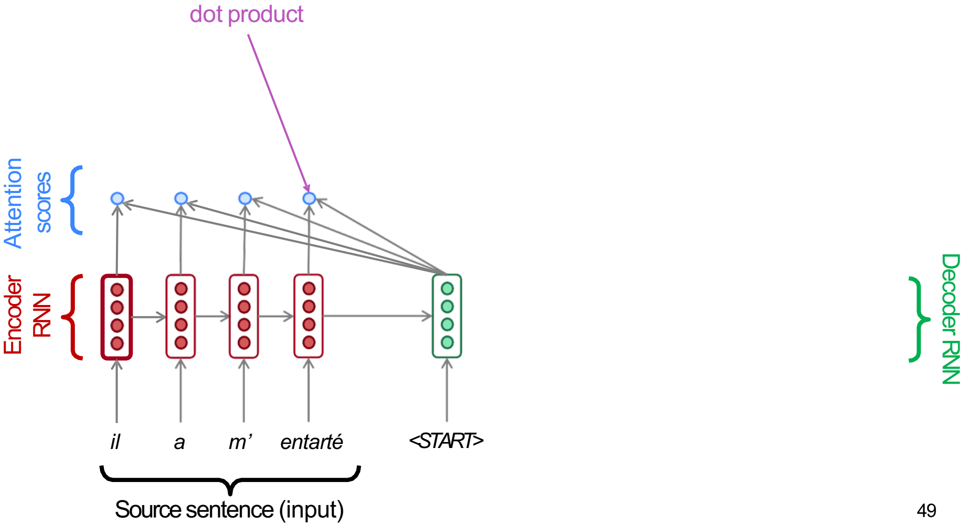
RNN architecture with attention



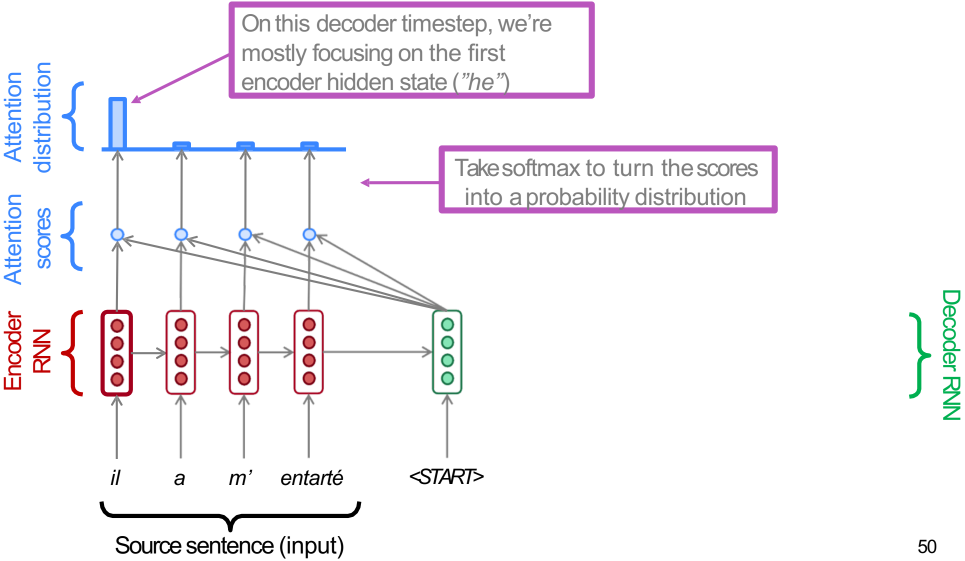
RNN architecture with attention



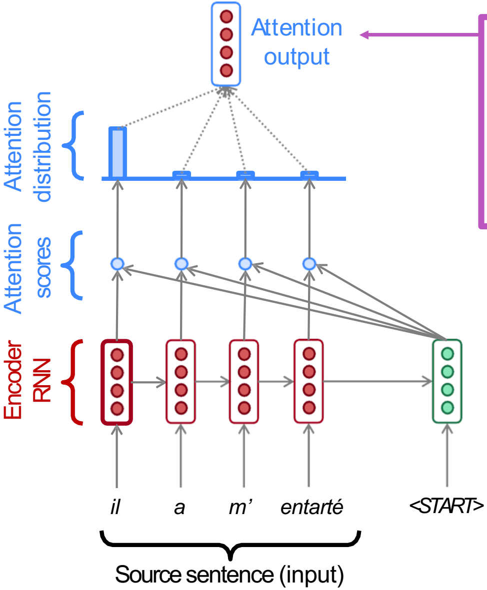
RNN architecture with attention



RNN architecture with attention



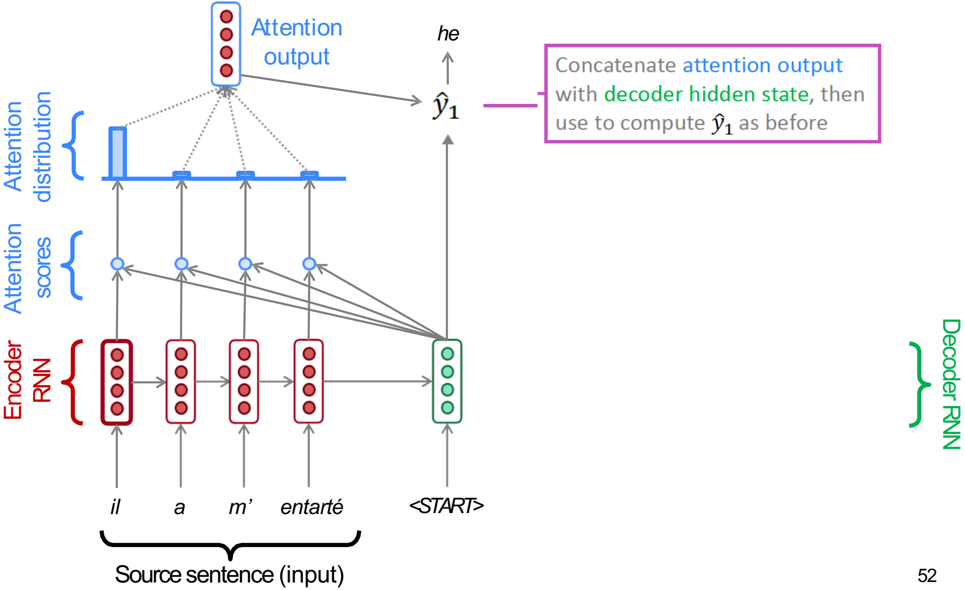
RNN architecture with attention



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

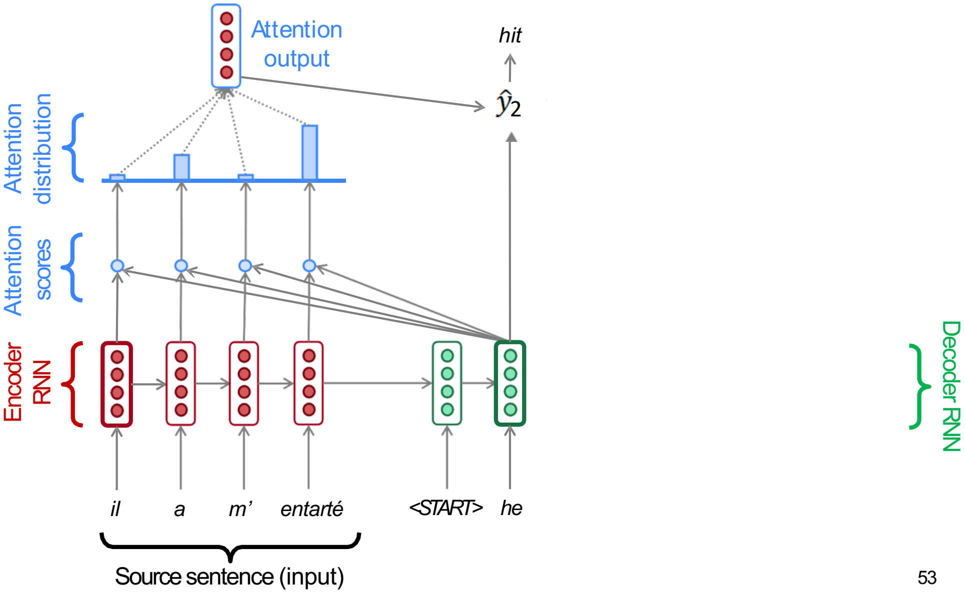
The attention output mostly contains information from the hidden states that received high attention.

RNN architecture with attention

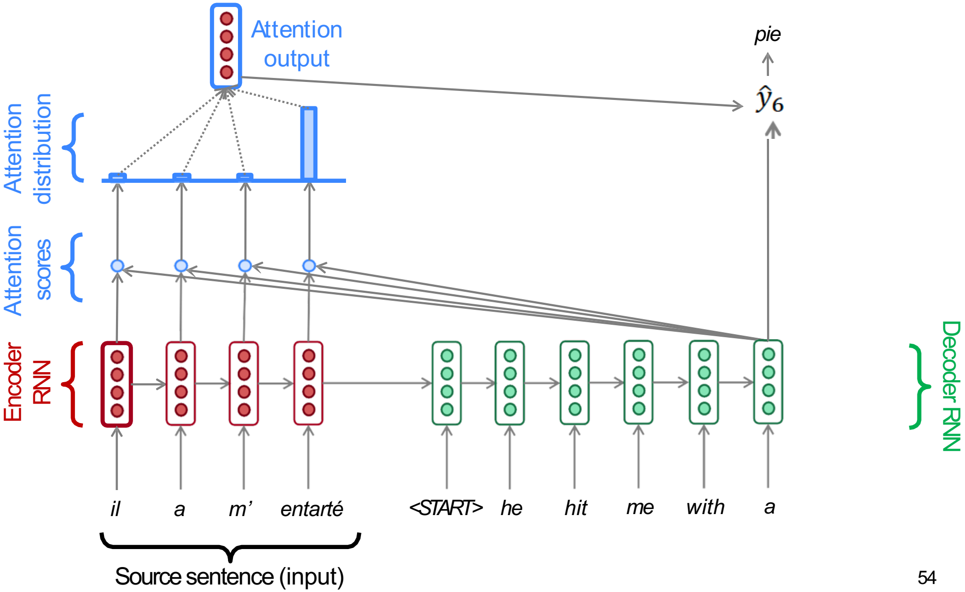


Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

RNN architecture with attention



RNN architecture with attention



Attention in Equations

- 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 $\boldsymbol{e}^t$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

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

$$\alpha^t = \mathrm{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

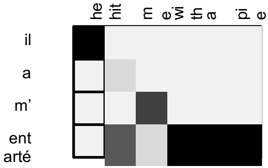
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Attention is great

- 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



Attention versatility

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- <u>However</u>: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)

- **<u>More general definition of attention</u>**:
  - Given a set of vector *values*, and a vector *query*, **<u>attention</u>** 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).

Attention versatility

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).

Attention variants

- We have some *values* $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a *query* $\boldsymbol{s} \in \mathbb{R}^{d_2}$

- Attention always involves:
  1. Computing the *attention scores* $\boldsymbol{e} \in \mathbb{R}^N$
  2. Taking softmax to get *attention distribution* α:

$$\alpha = \mathrm{softmax}(\boldsymbol{e}) \in \mathbb{R}^N$$

  3. Using attention distribution to take weighted sum of values:

$$\boldsymbol{a} = \sum_{i=1}^{N} \alpha_i \boldsymbol{h}_i \in \mathbb{R}^{d_1}$$

thus obtaining the *attention output* ***a*** (sometimes called the *context vector*)

There are multiple ways to do this

Attention variants

You'll think about the relative advantages/disadvantages of these in Assignment4!

There are several ways you can compute $\boldsymbol{e} \in \mathbb{R}^N$ from $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and $\boldsymbol{s} \in \mathbb{R}^{d_2}$:

- Basic dot-product attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
  - Note: this assumes $d_1 = d_2$
  - This is the version we saw earlier

- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
  - Where $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix

- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$
  - Where $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}, \boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $\boldsymbol{v} \in \mathbb{R}^{d_3}$ is a weight vector.
  - d_3 (the attention dimensionality) is a hyperparameter

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

Summary

- Sequence-to-sequence uses 2 RNNs

- Attention is a way to *focus on particular parts* of the input
  - Improves sequence-to-sequence a lot!

- We learnt this in the context of Neural MT, but they are really versatile