CS224N : Lecture 7 - Translation, S2S, Attention

1. Pre-Neural Machine Translation

Machine Translation

- Task of translating a sentence x from one language (the source language) to a sentence y in another language (the target language).

Alignment

- The correspondence between particular words in the translated sentence pair
- Typological differences between languages lead to complicated alignments
    - Many-to-One
    - One-to-Many
    - Many-to-Many

2. Neural Machine Translation

Neural Machine Translation (NMT)

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

▼ 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

Advantages of NMT

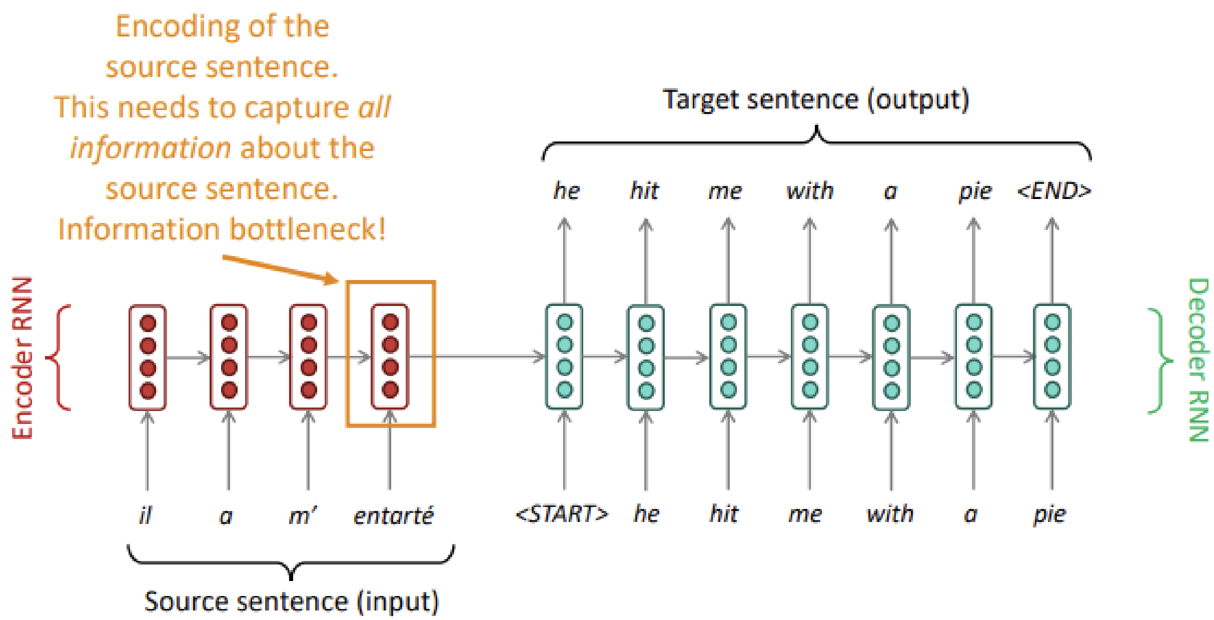
1. Better performance
   - More fluent
   - Better use of context
   - Better use of phrase similarities
2. A single neural network to be optimized end-to-end
   - No subcomponents to be individually optimized
3. Requires much less human engineering effort
   - No feature engineering
   - Same method for all language pairs

Disadvantages of NMT

1. NMT is less interpretable
   - Hard to debug
2. NMT is difficult to control
   - For example, can't easily specify rules or guidelines for translation
   - Safety concerns!

3. Attention

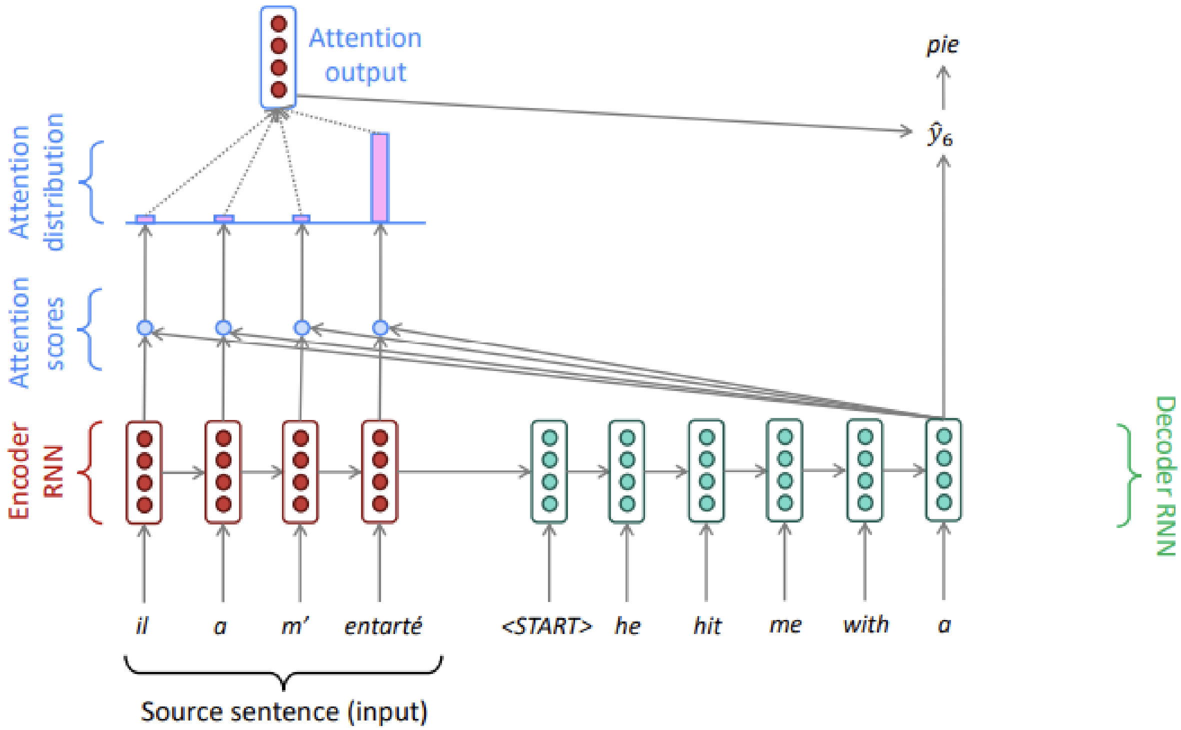
Sequence-to-sequence: the bottleneck problem



Attention

- Provides a solution to the bottleneck problem
- On each step of the decoder, use direct connection to the encoder to focus on a particular part of the source sequence
- 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.

▼ Sequence-to-sequence w/ attention



▼ Attention in equation

- 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

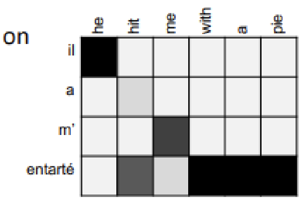
75

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

▼ Attention's Pros

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 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



▼ Attention Variants

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