Attention (developed in 2014s)

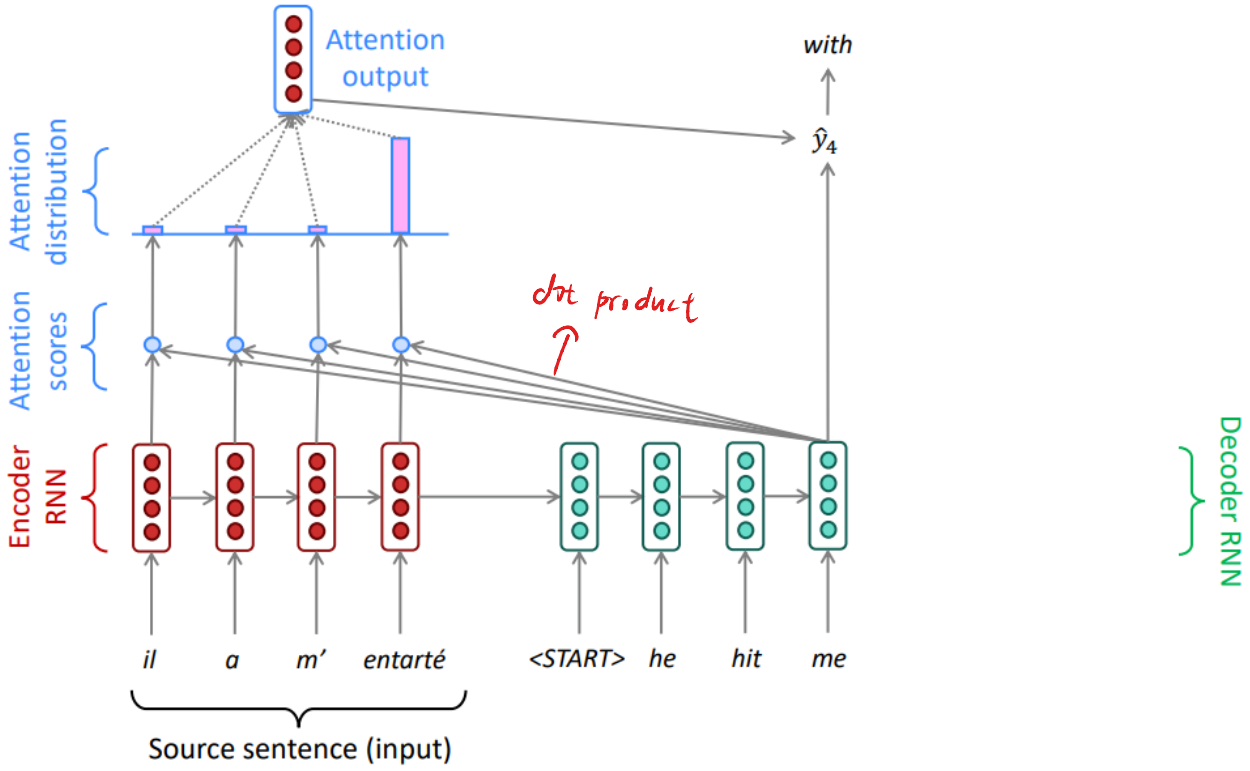
Why it appeared?

in LSTM model, everything useful about original sentence has to be stuffed into the context vector. If the input of encoder is a large sentence, it seems to be impossible to try to fit everything about that input sentence into a hidden state (Even if it's a multi-layer LSTM model, it still seems not good)

So how would new method (namely attention) looks like?

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

Sequence-to-sequence with attention



19

1. the decoder hidden state dot product each encoder hidden state to get the attention scores (Besides dot product, there are many ways to do this)
2. take softmax() to turn the scores into a probability distribution
3. we'll mostly focus on the encoder hidden state with heighest probability

4. use the attention distribution to take a weighted sum of the encoder hidden states, and this sum called attention output, it mostly contains information from the hidden states that received high attention

5. Concatenate attention output with decoder hidden state, then compute $\hat{y}_t$ as before

In math ↴

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 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)

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

- 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

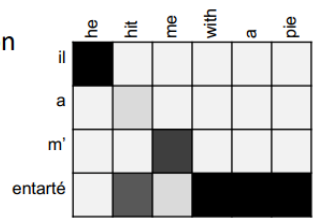
$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

22

Advantages

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



23

because it directly connect things

the example here is ↑

just like residual connections

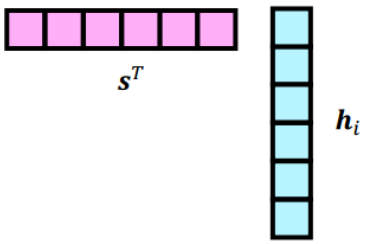
that entorté is correspond to have sb. pied in English

Variants

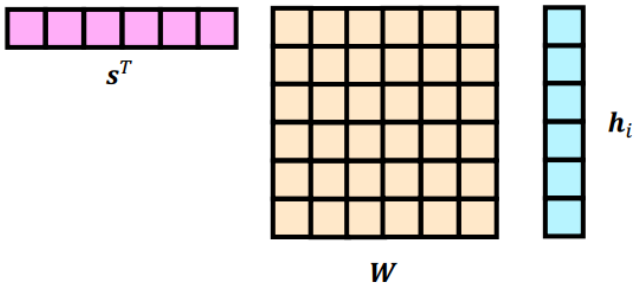
Attention variants

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

- Basic dot-product attention: $e_i = s^T h_i \in \mathbb{R}$
  - 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"



What the Basic dot-product attention's problem is

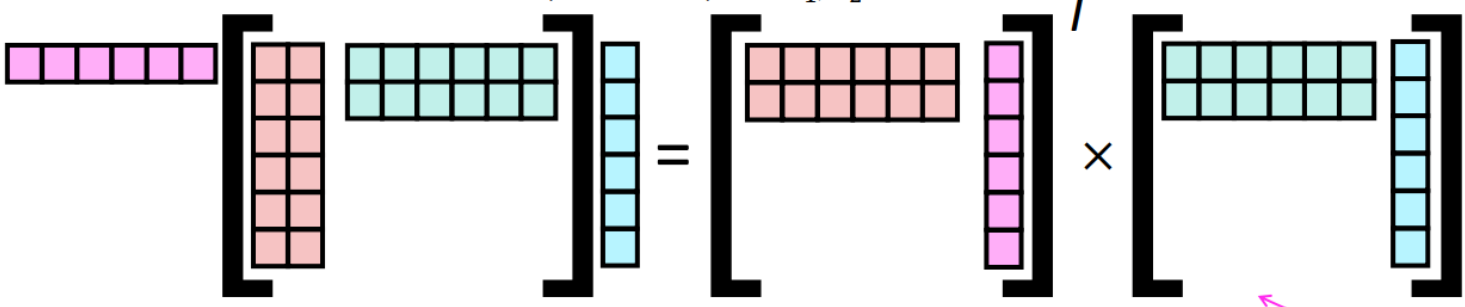
↳ because the hidden state of an LSTM is complete memory

it's got all kinds of memory, so some of it would be useless for finding up

↓↓

Then people developed multiplicate attention which its matrix X can learn where it should find the corresponding place in hidden state

- Reduced-rank multiplicative attention: $e_i = s^T (U^T V) h_i = (Us)^T (Vh_i)$
  - For low rank matrices $U \in \mathbb{R}^{k \times d_2}$, $V \in \mathbb{R}^{k \times d_1}$, $k \ll d_1, d_2$



- Additive attention: $e_i = v^T \tanh(W_1 h_i + W_2 s) \in \mathbb{R}$ [Bahdanau, Cho, and Bengio 2014]
  - Where $W_1 \in \mathbb{R}^{d_3 \times d_1}, W_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $v \in \mathbb{R}^{d_3}$ is a weight vector.
  - d_3 (the attention dimensionality) is a hyperparameter

Remember this when we look at Transformers

- a_3 (the attention dimensionality) is a hyperparameter
- "Additive" is a weird/bad name. It's really using a feed-forward neural net layer.

next week!

by linear transition, we can think the matrix multiplication as that take each of these two vectors and project them to a low-dimensional space using this low rank transformation matrix, then we do dot product in this low-dimensional space

acutally it's earlier than other two

defect: the small neural network in it is slow and complex to compute

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).

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!

down stream / upstream

downstream : more specific and application-oriented tasks or stages

eg. Text classification、QA system、NER. MT...

upstream : more fundamental and general tasks or stages

eg. Pretraining. general model、fundamental data...

概念	角色与定义	与Seq2Seq的关系
RNN / LSTM	**具体神经网络单元**，用于处理序列数据。	是**实现**Seq2Seq模型（编码器/解码器）的常用、经典的**底层组件**。
Seq2Seq	**一种抽象的模型架构**，用于解决序列到序列的映射问题。	**本身**。它定义了编码器-解码器的框架，可以用RNN、LSTM或Transformer来实现。
Attention	**一种机制**，用于增强Seq2Seq等模型，让解码器能动态聚焦于编码器的不同部分。	是**对原始Seq2Seq架构的重大改进**。它被“加装”在基于RNN/LSTM的Seq2Seq上，解决了信息瓶颈问题。
Transformer	**一个完全基于Self-Attention的全新模型架构。**	是**Seq2Seq架构的另一种、更先进的实现方式**。它用Self-Attention层完全取代了RNN/LSTM作为编码器和解码器的组件。