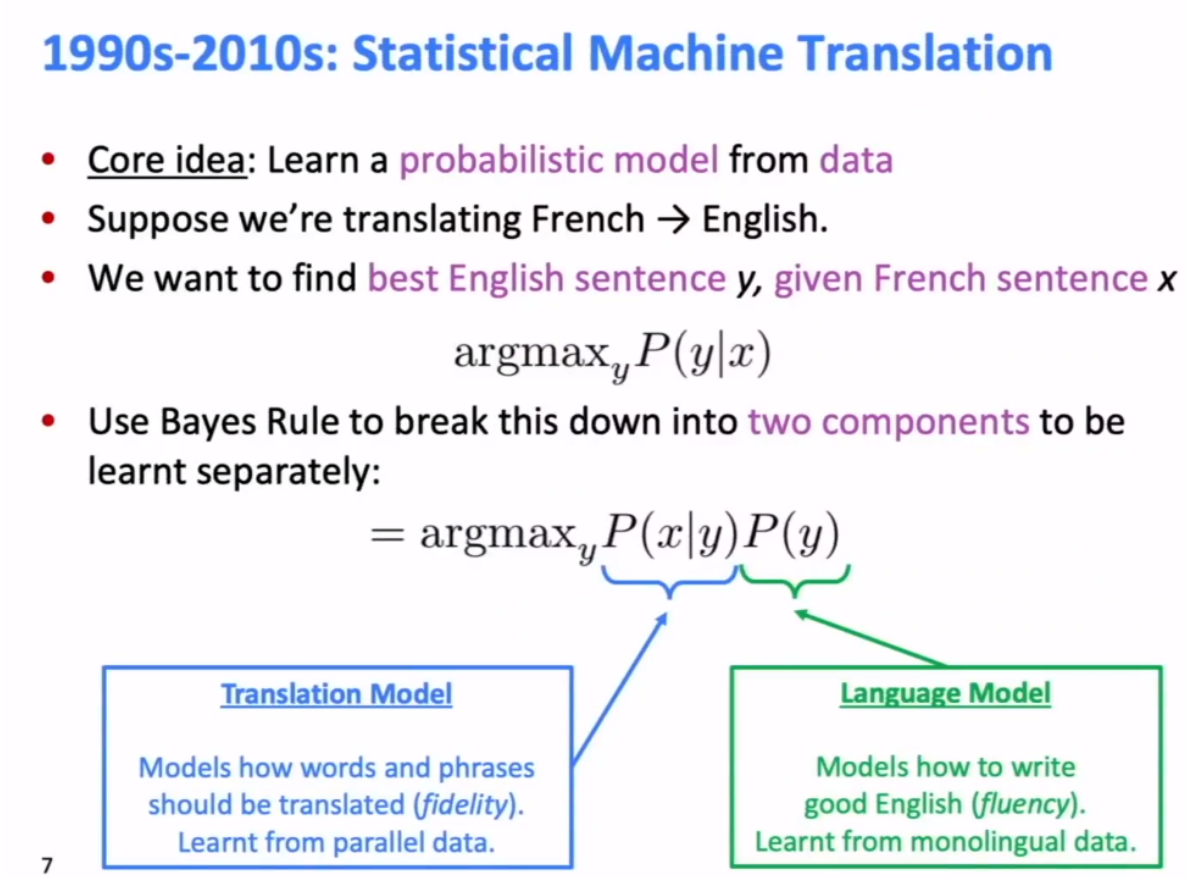
Machine Translation

任务

将source language sentence x翻译成target language sentence y。

Statistical Machine Translation (SMT)



Language Model已经知道如何学习了(n-gram, RNN等)，下面关注如何学习Translation Model。
parallel data指多条表达相同含义的语言数据。

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*

Alignment可能是

- one-to-one
- many-to-one
- one-to-many
- many-to-many (phrase level)

Decoding for SMT

Decoding for SMT

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

Question: How to compute this argmax?

Translation Model

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*

Neural Machine Translation (NMT)

用单个神经网络进行翻译。最常用的是Seq2seq，两个串联的RNN。

Seq2seq和SMT不同，前者直接学习$P(y|x)$，后者学习$P(x|y)P(y)$。

- 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 $P(y|x)$:

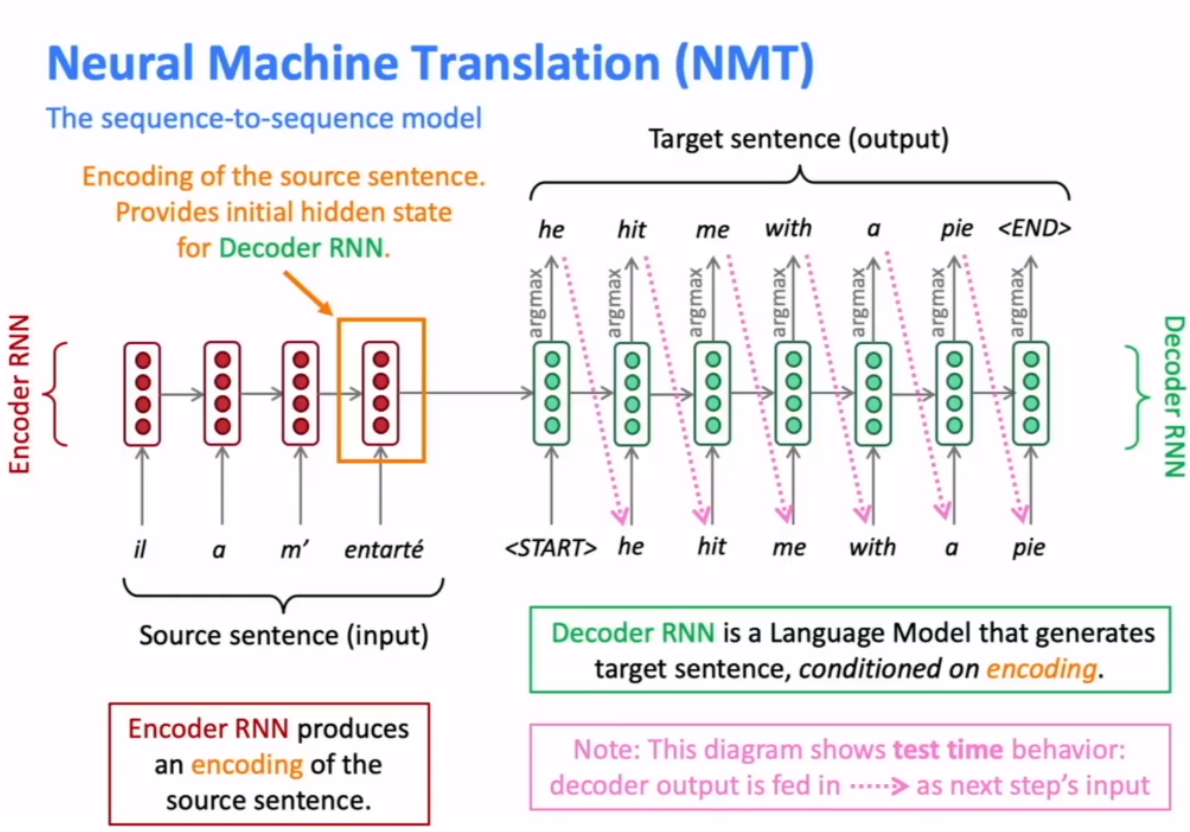
$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots\underbrace{P(y_T|y_1,\ldots,y_{T-1},x)}$$

Probability of next target word, given target words so far and source sentence *x*

- **Question**: How to train a NMT system?

Seq2seq

Seq2seq (Inference)



上图是Inference阶段的seq2seq流程。

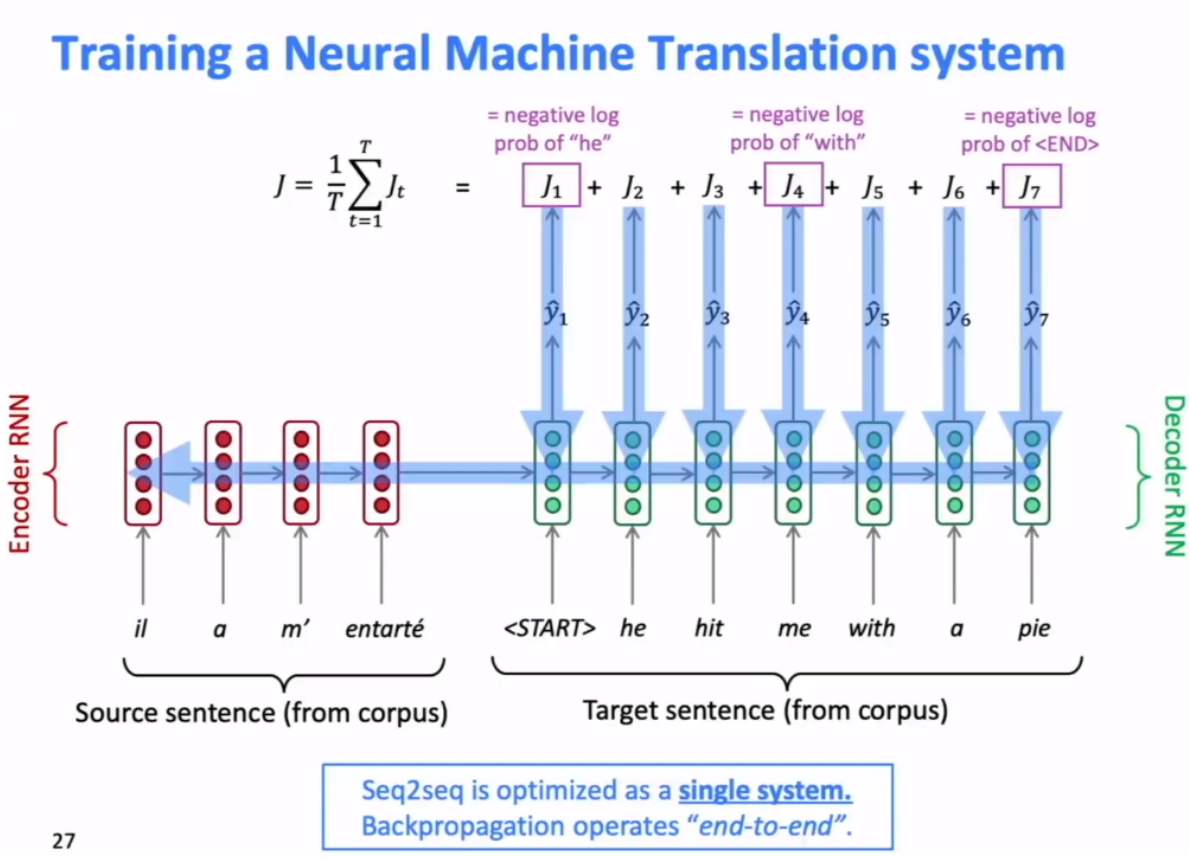
注：

- 法语和英语单词都需要各自一套wordvec，因为都需要作为输入。

Seq2seq的其他用途

- 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)

Seq2seq (Train)

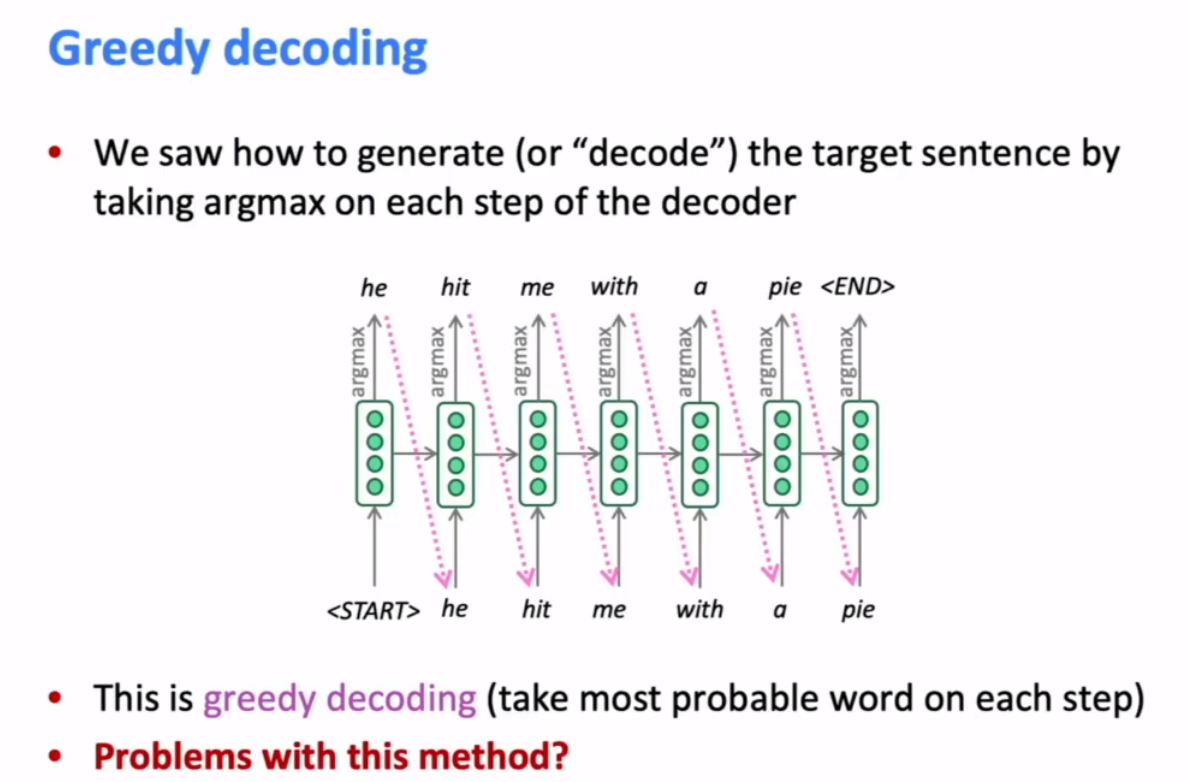


上图是Train阶段的seq2seq流程。

注：

- 如果Inference时预测出<END>，就停止预测；在Train时预测出<END>，仍然要继续预测，因为有可比较的训练数据。
- encoder和decoder也可以分开train，比如载入预训练的模型再fine-tune。
- encoder和decoder可以分开组合，来构成通用(任何两种语言)的翻译器。

Greedy decoding



Greedy decoding的$argmax$只是当前单词的最优，不能保证全局的$argmax(P(y|x))$。因为：

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots P(y_T|y_1,\ldots,y_{T-1},x)$$

确定了y_1后，后续都是基于y_1的，未必保证全局最优的y。

产生的问题是，一个单词错了，后面可能都错了，无法回头。

Beam search decoding

遍历搜索需要$O(V^T)$的复杂度，V是词汇表大小，T是句子长度，保证找到最优解，但显然不可能实现。

采用Beam search，每步保存k个最有可能的partial translations (即$P(y_T|y_1,\ldots,y_{T-1})$)。

Beam search decoding

- Core idea: On each step of decoder, keep track of the *k* most probable partial translations (which we call *hypotheses*)
  - *k* is the beam size (in practice around 5 to 10)

- A hypothesis $y_1, \ldots, y_t$ has a score which is its log probability:

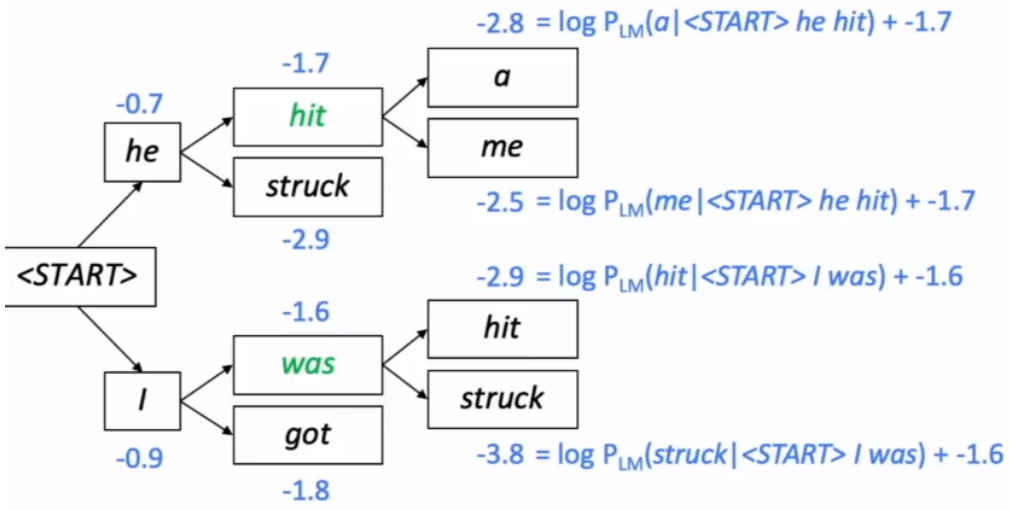
$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses, tracking top *k* on each step

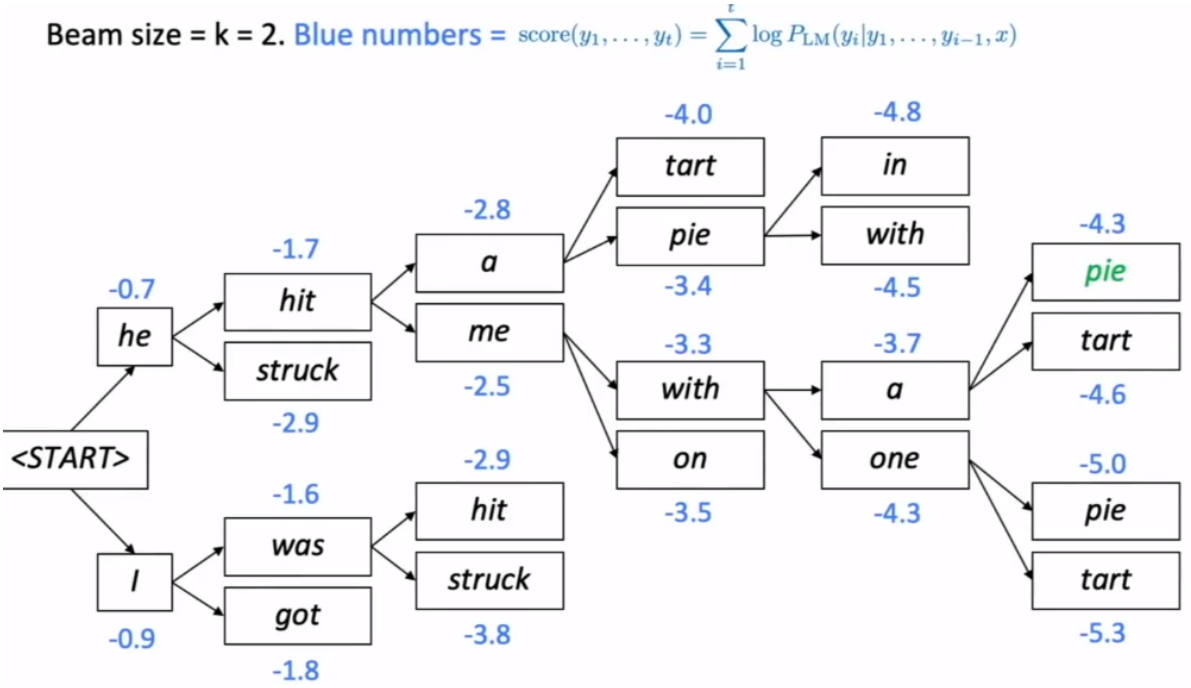
- Beam search is not guaranteed to find optimal solution
- But much more efficient than exhaustive search!

例子

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



For each of the *k* hypotheses, find top *k* next words and calculate scores



停止条件

在greedy decoding中，通常出现第一个<END>就停止。

但是beam search中，不同分支各自会产生<END>，某一分支达到<END>就终止。

最后整个beam search满足以下任一条件就停止：

- 分支达到T步
- 有了n个hypotheses (n个完整的预测句子)

注：

- 句子越长分数一定越低，因为每个word的分数为负。不需要除以T来标准化。

Beam search decoding: finishing up

- We have our list of completed hypotheses.
- How to select top one with highest score?

- Each hypothesis $y_1, \dots, y_t$ on our list has a score

$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

- Problem with this: longer hypotheses have lower scores

- Fix: Normalize by length. Use this to select top one instead:

$$\frac{1}{t} \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

Advantages and disadvantages of NMT

优点

Compared to SMT, NMT has many advantages:

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

缺点

Compared to SMT:

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

Evaluation of MT

BLEU计算目标语言机翻和人翻的n-gram相似度(顺序必须一致)，对较短的翻译有惩罚。

How do we evaluate Machine Translation?

BLEU (**B**ilingual **E**valuation **U**nderstudy)

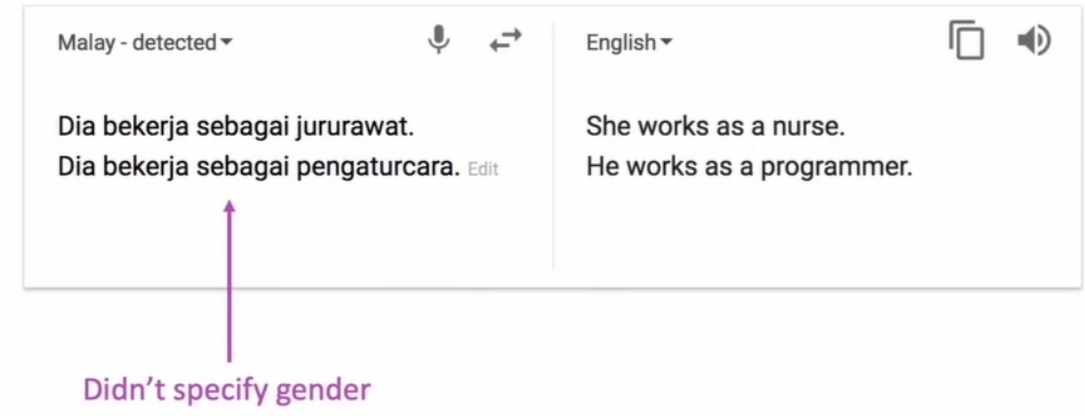
You'll see BLEU in detail in Assignment 4!

- BLEU compares the machine-written translation to one or several human-written translation(s), and computes a similarity score based on:
  - *n*-gram precision (usually for 1, 2, 3 and 4-grams)
  - Plus a penalty for too-short system translations
- BLEU is useful but imperfect
  - There are many valid ways to translate a sentence
  - So a good translation can get a poor BLEU score because it has low *n*-gram overlap with the human translation ☹

Problem to be solved in MT

待解决的问题(研究很活跃):

- Out-of-vocabulary words。一些词汇库里没有的词。
- Domain mismatch。比如在wiki语料上训练，但是用于chat。
- Maintain context over longer text。翻译一本书，需要考虑前文的context。
- Low-resource language pairs。语料较少的语言。
- Common sense。一些俚语、常识性语句无法翻译。
- Biases。



- Nonsense。输入噪声，可能也会基于训练数据生成目标语言。类似GAN可以通过输入高斯噪声生成图片。

Attention

起因

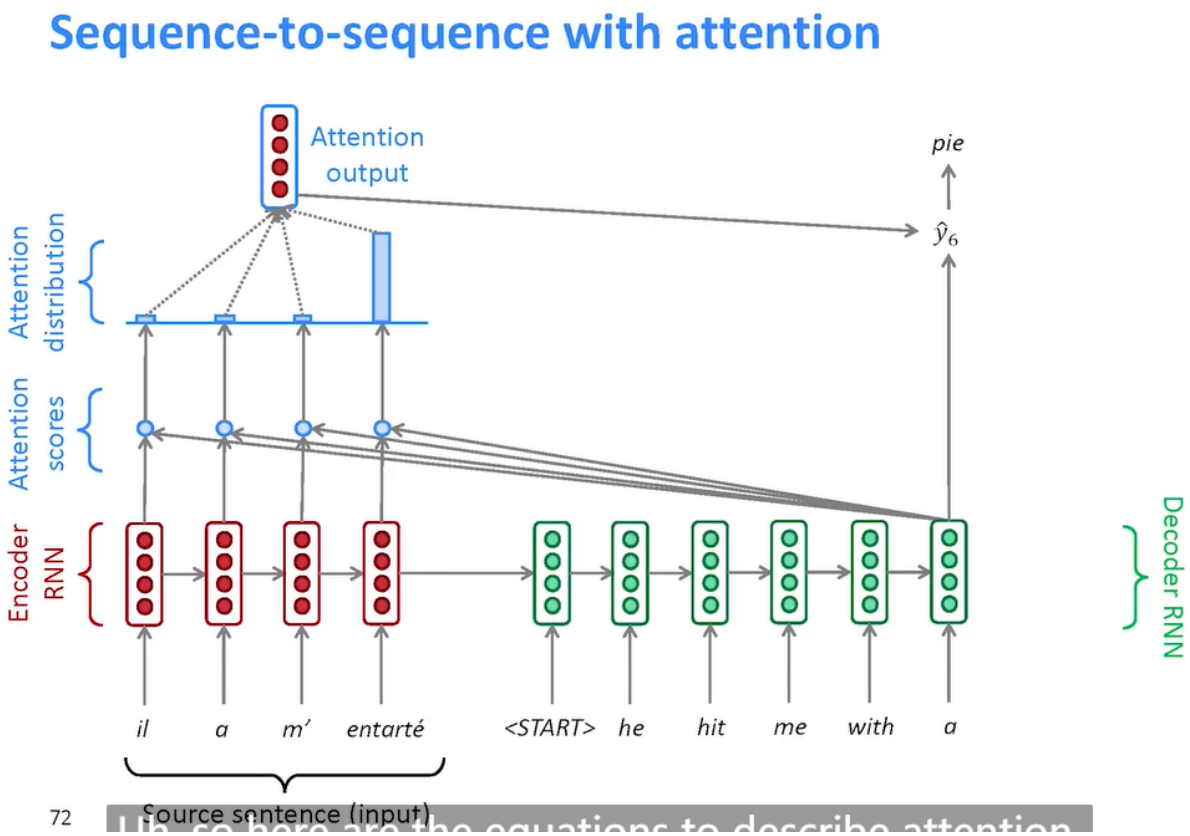
Seq2seq中encoder把最后一层的hidden layer作为整句话的embedding，会丢失前文的信息，会产生information bottleneck(信息瓶颈)。

核心

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

Seq2seq with attention

attention结构：



attention数学计算：

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

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

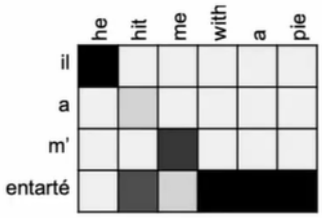
73

最后用计算y时用的权重矩阵是wordve矩阵还是任意初始化的权重矩阵?

Advantages of attention

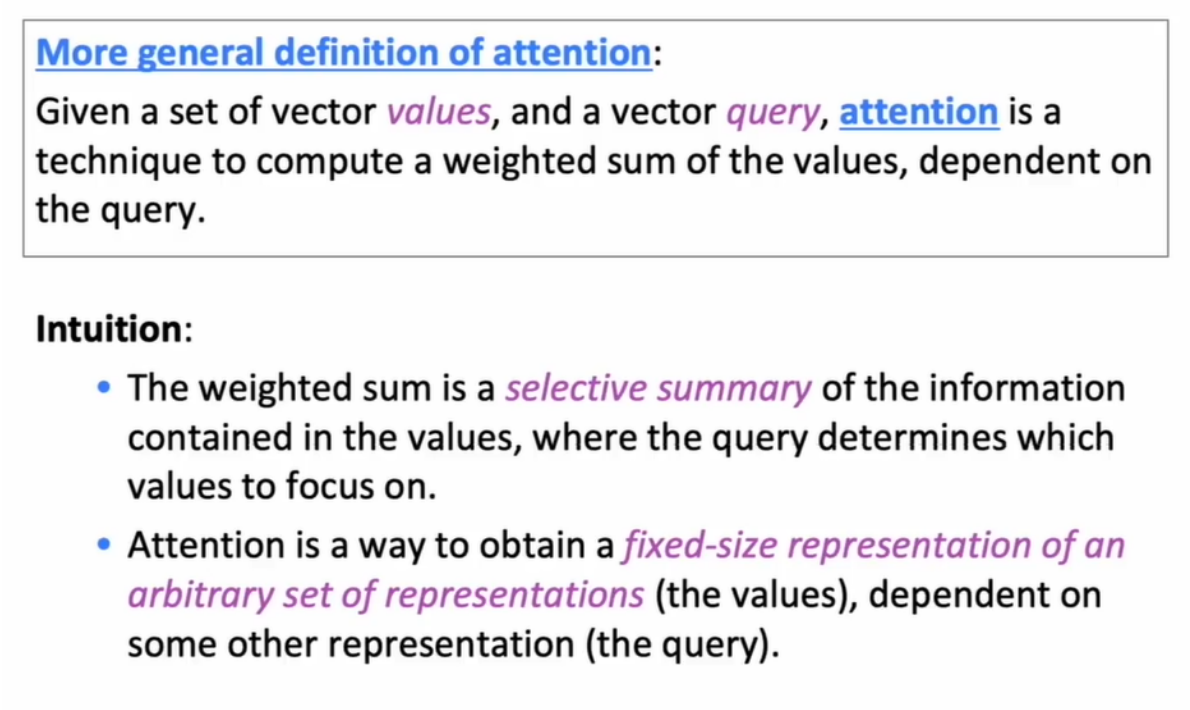
Attention is great

- 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 generalization

attention已经变成了一种通用手段。



Attention variants

variants主要集中在三个方面：

1. 计算attention分数
   - $e_i = s^T h_i$: embedding点积(原版本)。
   - $e_i = s^T W h_i$: 乘以一个权重矩阵，这样不要求s和h维度相同。
   - $e_i = v^T tanh(W_1 h_i + W_2 s)$: 其中$W1, W2, v$都是权重矩阵。类似加了一个线性层。
2. 从attention分数计算分布
3. 从分布计算value