Advanced Topics in Machine Learning

Alejo Nevado-Holgado

Lecture 14 (NLP 6) - Machine Translation, Seq2seq, & Attention

V 0.3 (26 Feb 2020 - final version)

The Georgetown-IBM experiment

- **The Georgetown-IBM experiment:** In 1952, Researchers at Georgetown and IBM demonstrated an algorithm able to translate from Russian to English, and gave a big press release. The examples were cherry-picked for the experiment, but investigative journalists rarely investigate and nobody noticed. The researchers claimed that language translation was likely going to be fully solved within 4 years[DOI:10.1007/978-3-540-30194-3_12].

- **The ALPAC report:** In 1966 the ALPAC report criticised the design of the experiments of 1952, and how scientists and journalists exaggerated future progress[Library of Congress 66-61843]. Research funding was axed.

- This is how you create winters in AI

Fig. 2: Hurd, Dostert and Watson at the demonstration

The Georgetown-IBM experiment

The New York Times

Google's Next Phase in Driverless Cars: No Steering Wheel or Brake Pedals

May 27, 2014

- Sergey Brin, one of the internet company's founders, expects its autonomous driving system to be ready for the market in five years ← 2013
  [https://www.economist.com/special-report/2013/04/20/look-no-hands]
- Such "autonomous vehicles" will be a reality for "ordinary people" in less than five years, Google co-founder Sergey Brin said Tuesday ← 2012
  [https://www.computerworld.com/article/2491635/self-driving-cars-a-reality-for--ordinary-people--within-5-years--says-google-s-sergey-b.html]
- What year is it today?

Old technology: Bayesian (1990s-2010s)

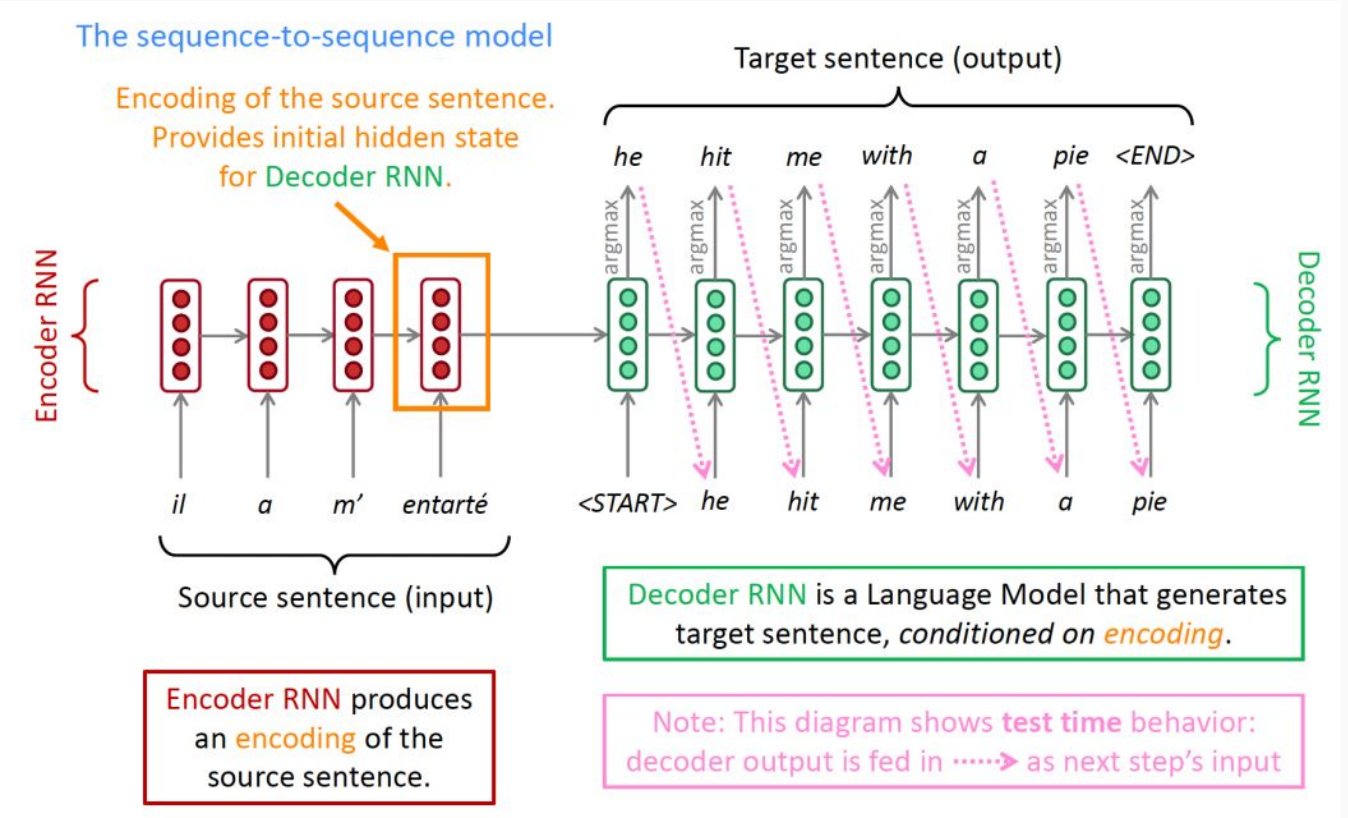
- **Idea:** Using the Bayes rule, break the whole problem into two probability functions. Then use statistical models to model each probability function separately.
- **Example:** Tro translate from french into english

$$\text{argmax}_{\text{english}} P(\text{english} \mid \text{french}) = \text{argmax}_{\text{english}} P(\text{french} \mid \text{english})\, P(\text{english})$$

- Two statistical models:
  - Translation Model for P(french | english): Models how words and phrases should be translated (fidelity). Learnt from parallel data.
  - Language Model for P(english): Models how to write good english (fluency). Learnt from monolingual data.
  - Decoding algorithm $\text{argmax}_{\text{english}}$: Finds the translation that maximises the probabilities of the other 2 models. Usually beam search

New technology: seq2seq NNs (≥2010s)

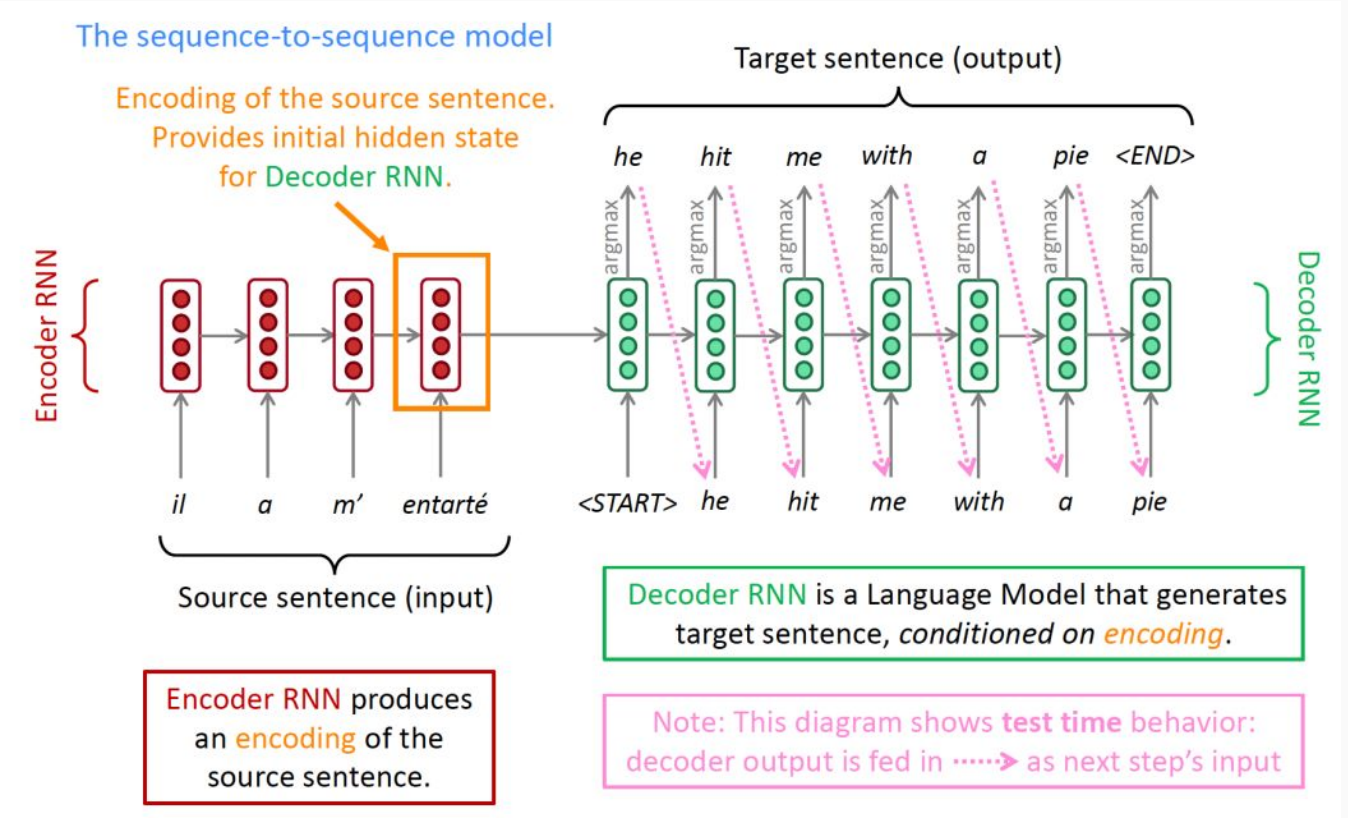
Idea: Use an end-to-end NN to implement "$\text{argmax}_{\text{english}} P(\text{english} \mid \text{french})$" straight away. One RNN reads word-by-word the original french sentence and creates a numerical representation of its meaning. Another RNN reads that numerical representation and creates the english translation word-by-word → Seq2seq NN



New technology: seq2seq NNs (≥2010s)

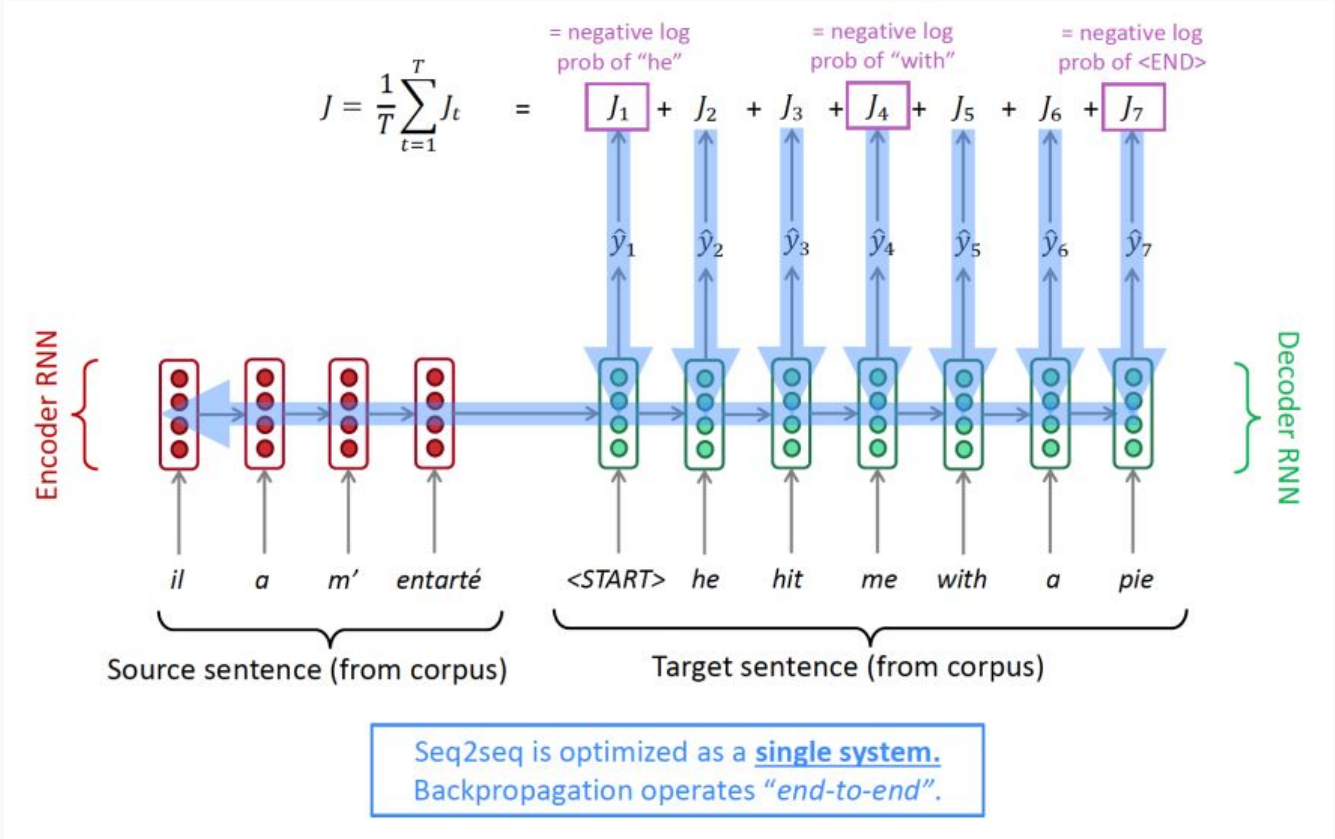
Seq2seq has multiple applications:

- Summarization: long text → short text
- Dialogue: previous utterances → new utterances
- Parsing: input text → parse as a sequence
- Code generation: natural language → python code (don't do this last one, all of us will lose out jobs)



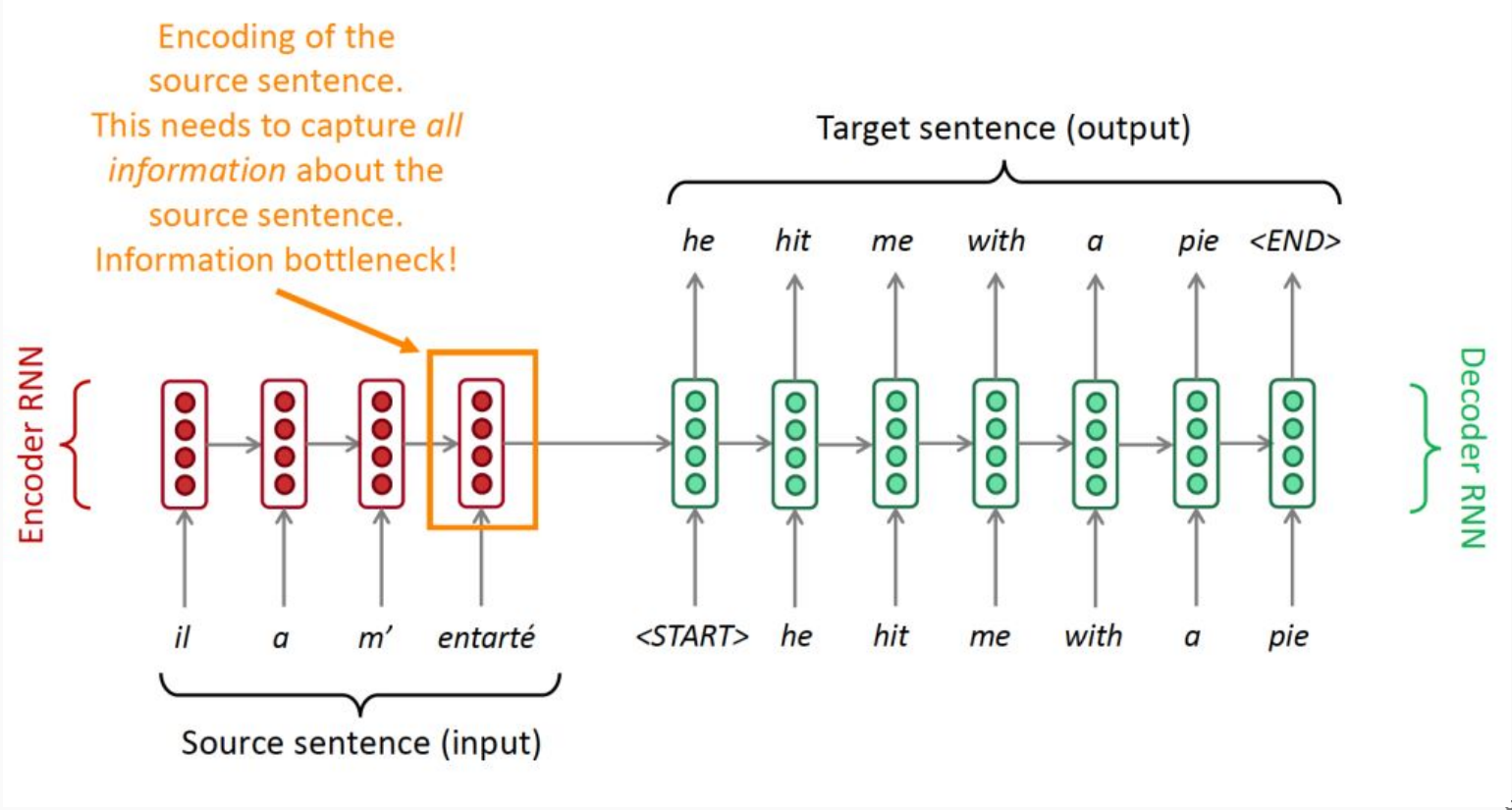
Seq2seq: Training

We can train a seq2seq with the same technology has any other RNN (backpropagation through time)



Translation: Attention

Problem: Too much information needs to be stored into $\mathbf{h}^{(t)}$ in the last step of the encoder

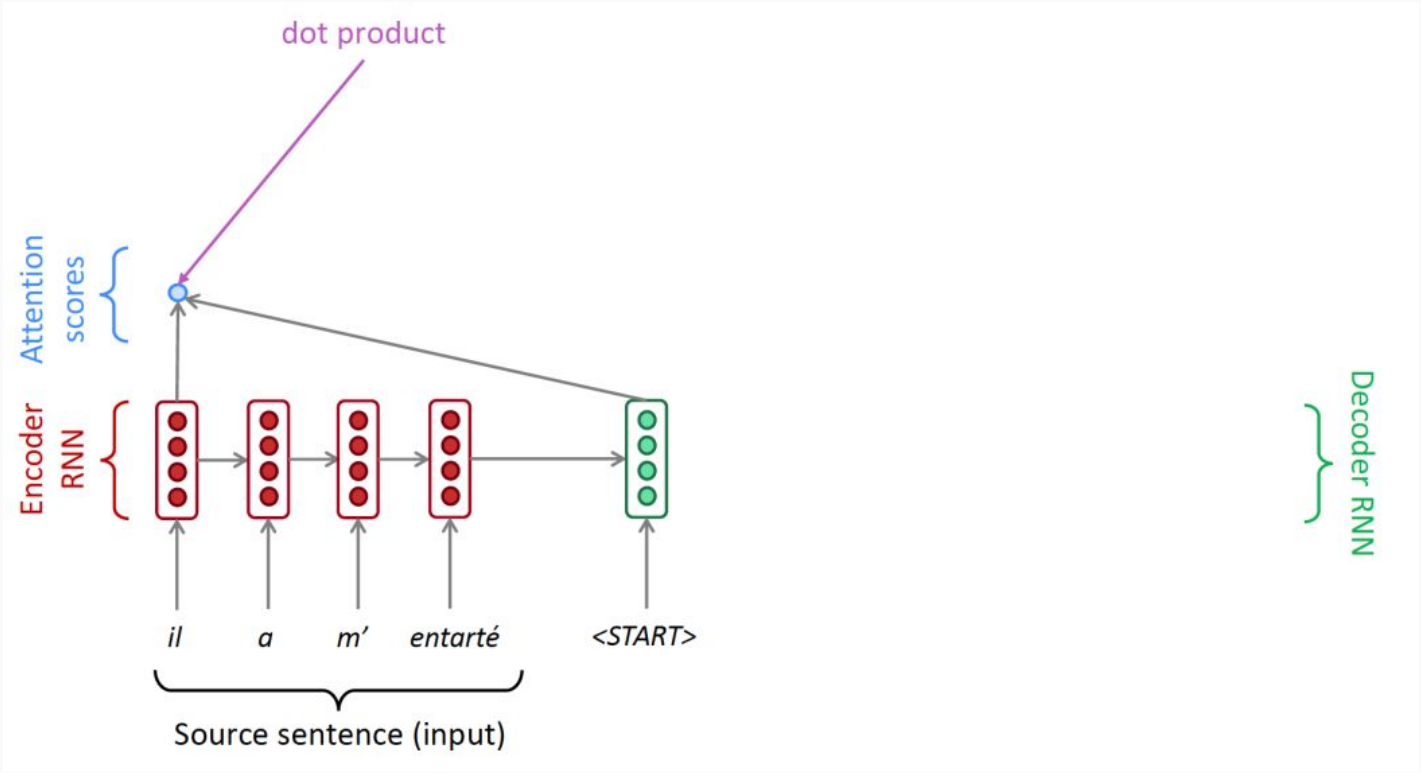


Translation: Attention

- Attention can be used in any NN, not only RNNs and NLP
- **Idea:** Given a set of vector values (in our example, $\mathbf{h}^{(t)}$), and a vector query (in our example, $\mathbf{s}^{(t)}$), attention is a technique to compute a weighted sum of the values dependent on the query (in our example $\mathbf{a}^{(t)}$)
- **Intuition 1:** The weighted sum is a selective summary of the information contained in the values, which the query determines which values to focus on
- **Intuition 2:** 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)

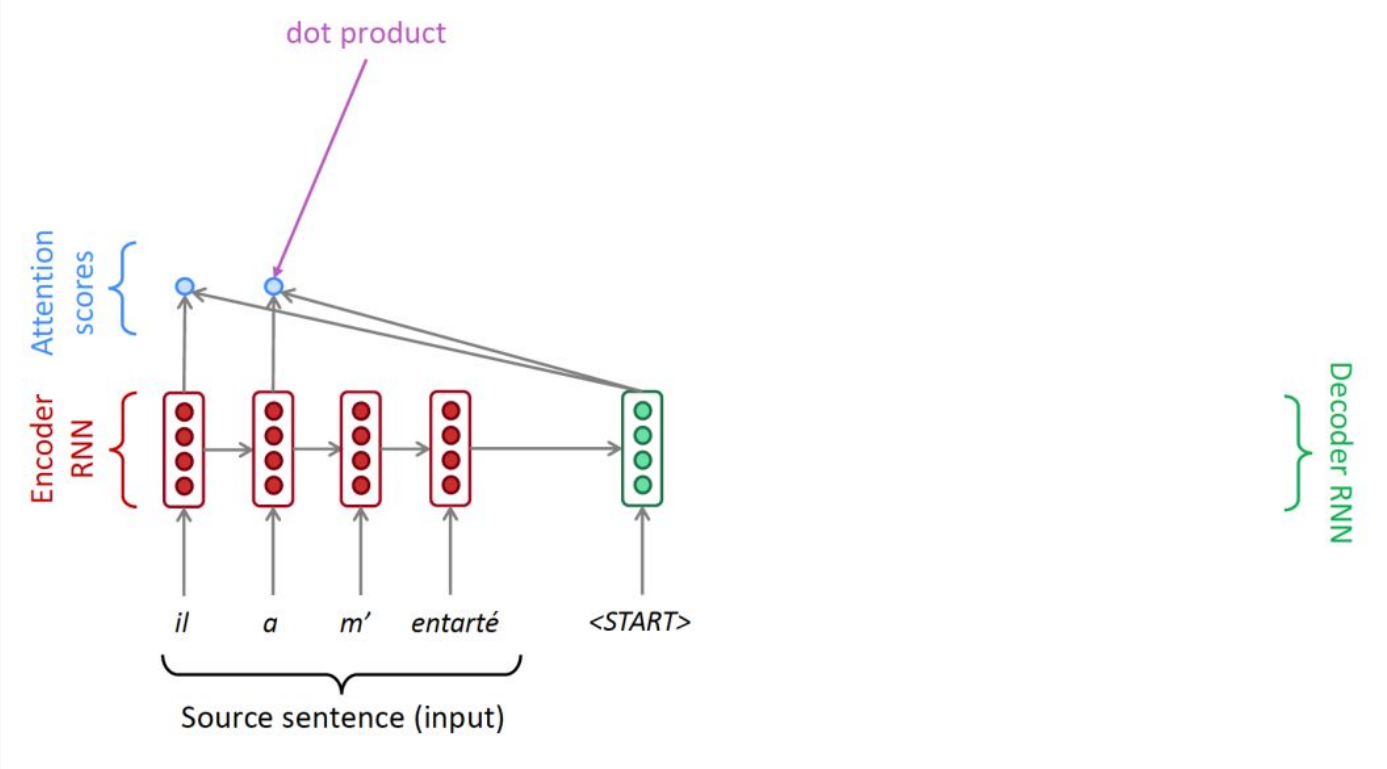
Translation: Attention

Idea: Bypass the bottleneck by allowing the decoder to directly access steps of the encoder



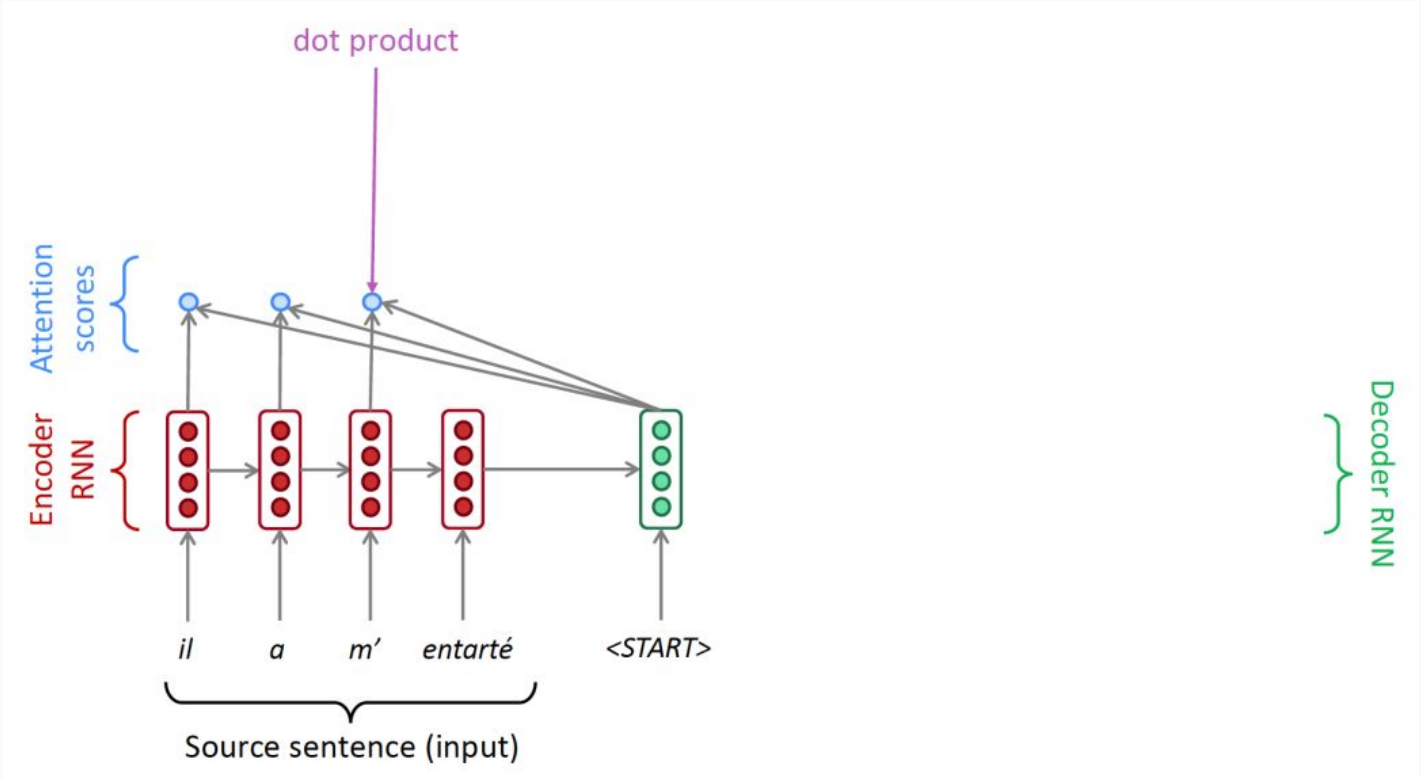
Translation: Attention

Idea: Bypass the bottleneck by allowing the decoder to directly access steps of the encoder



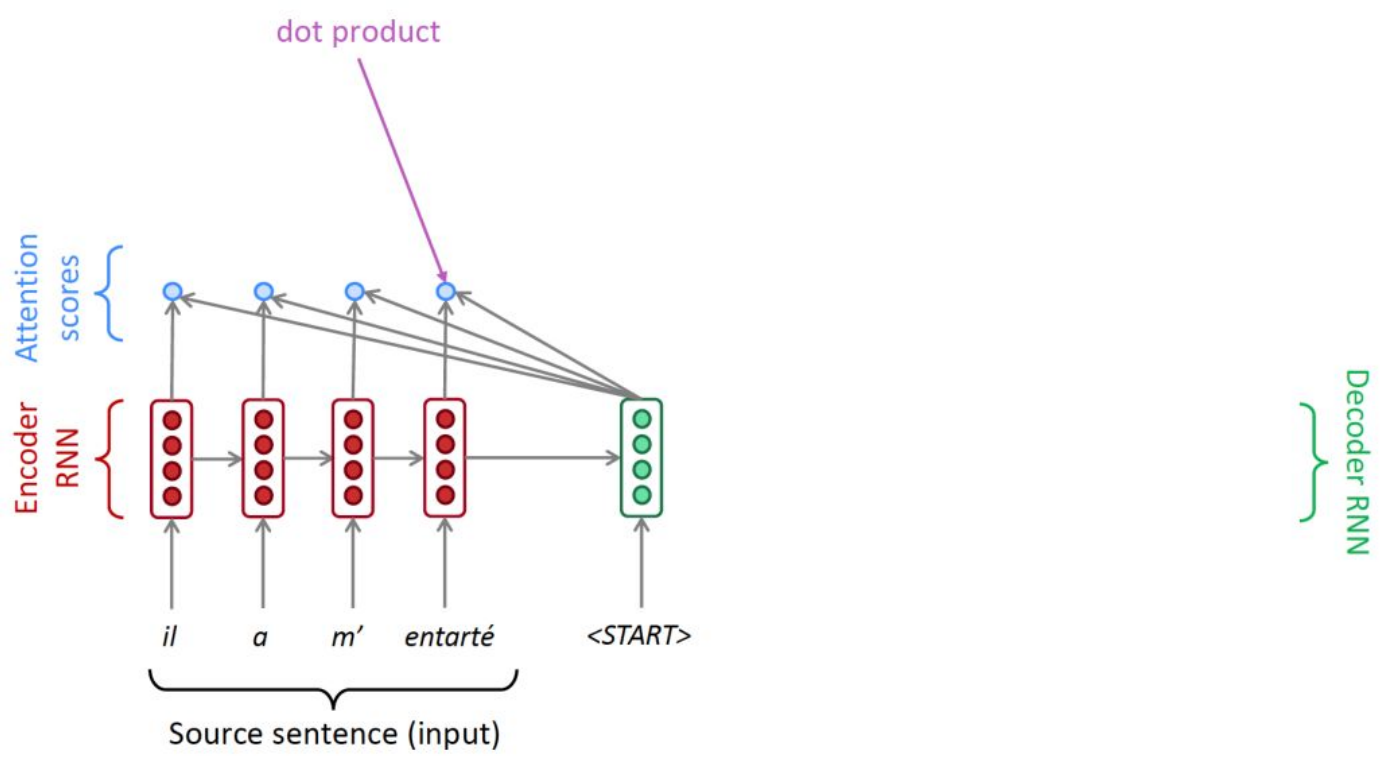
Translation: Attention

Idea: Bypass the bottleneck by allowing the decoder to directly access steps of the encoder



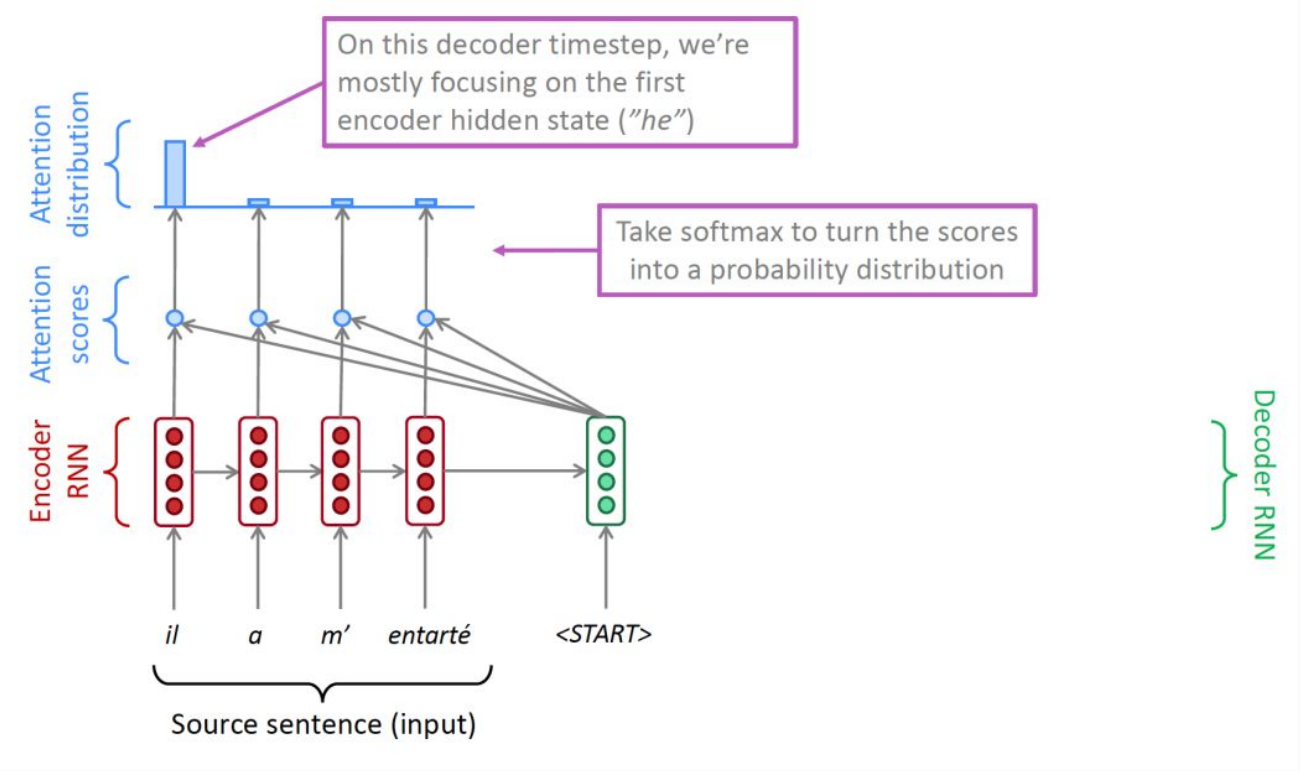
Translation: Attention

Idea: Bypass the bottleneck by allowing the decoder to directly access steps of the encoder

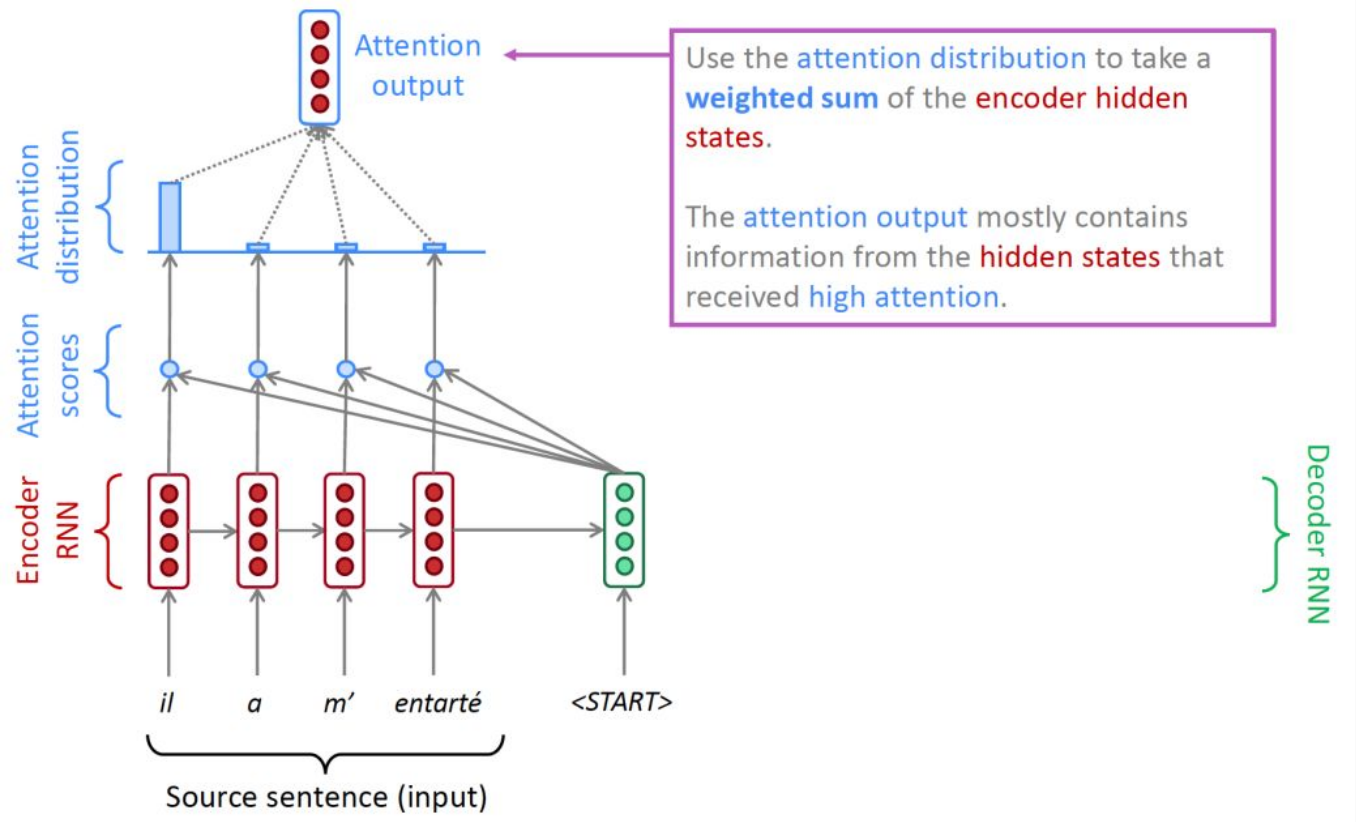


Translation: Attention

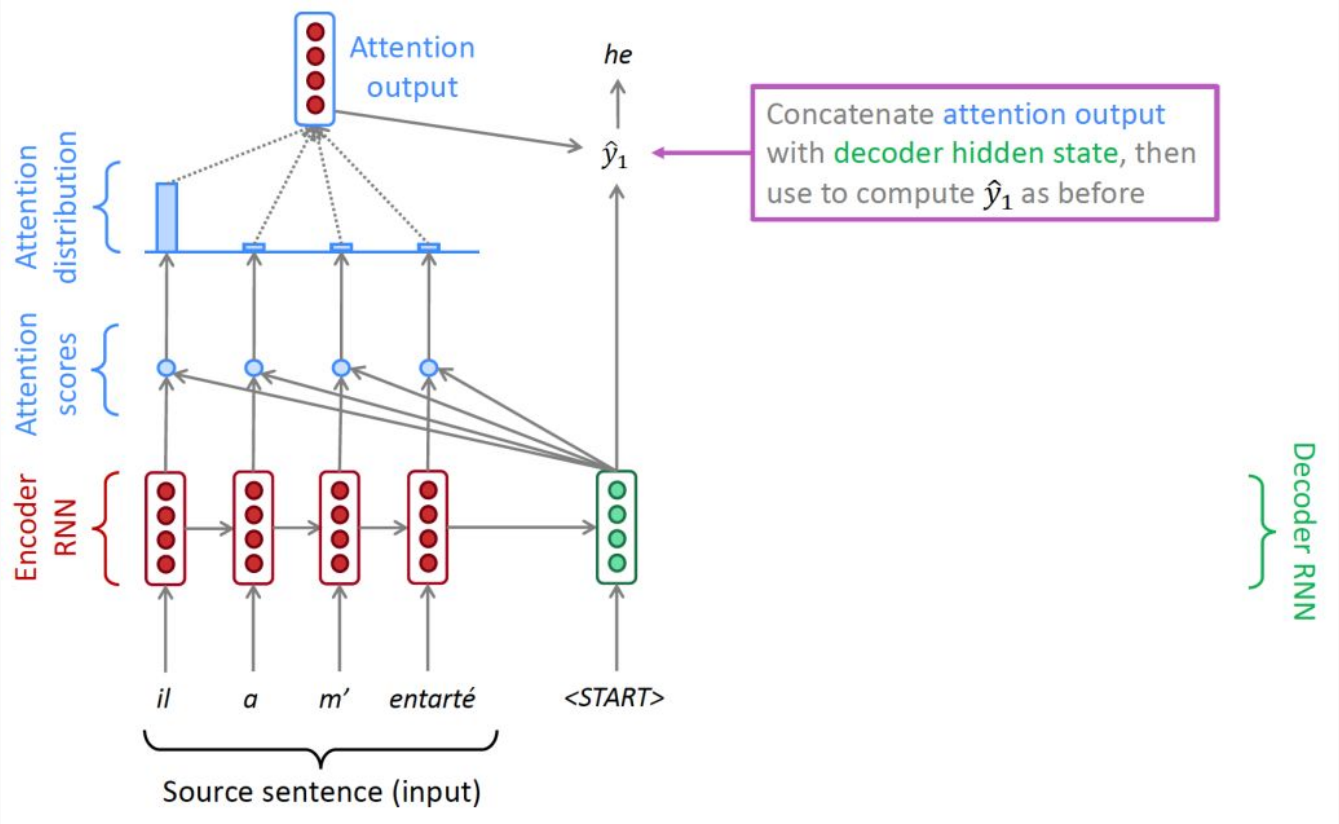
Idea: Bypass the bottleneck by allowing the decoder to directly access steps of the encoder



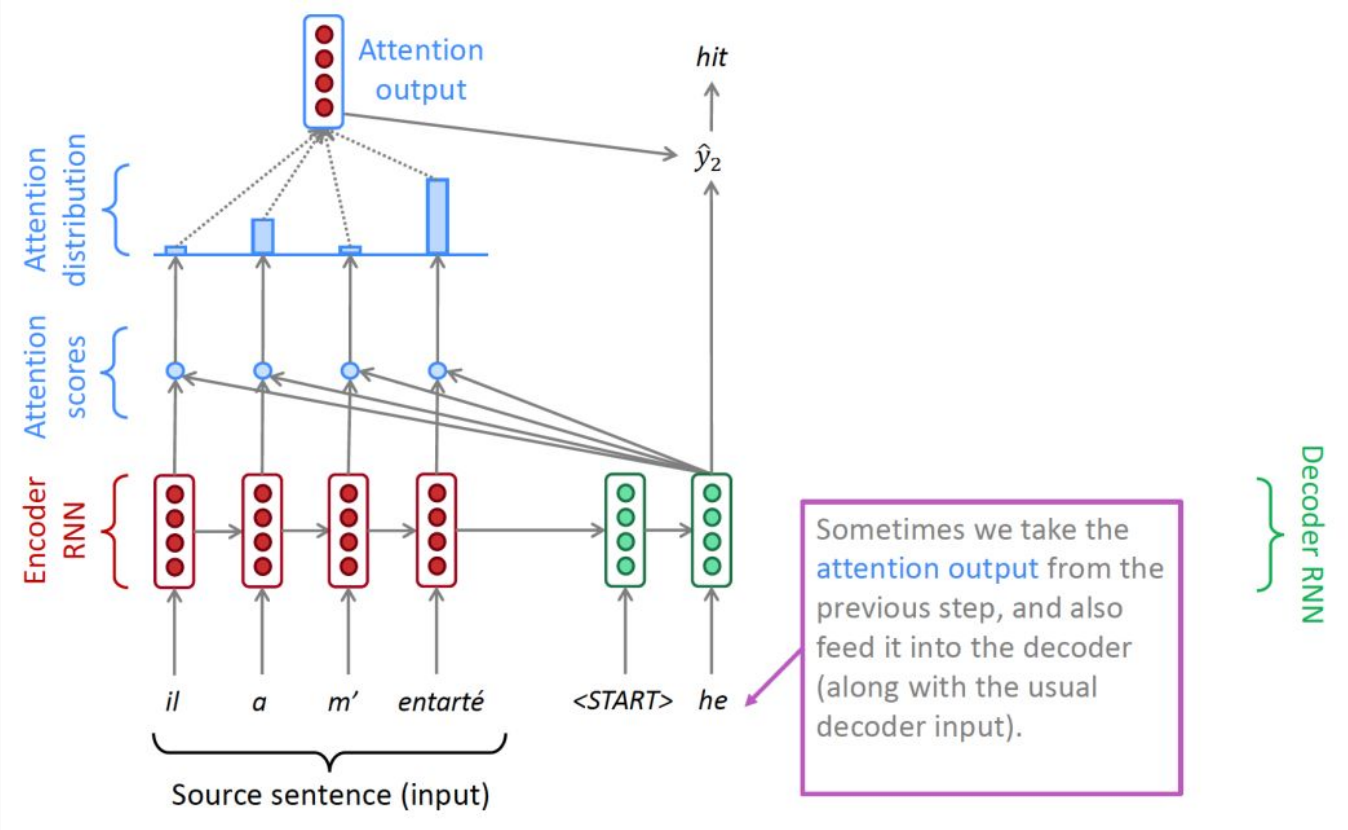
Translation: Attention



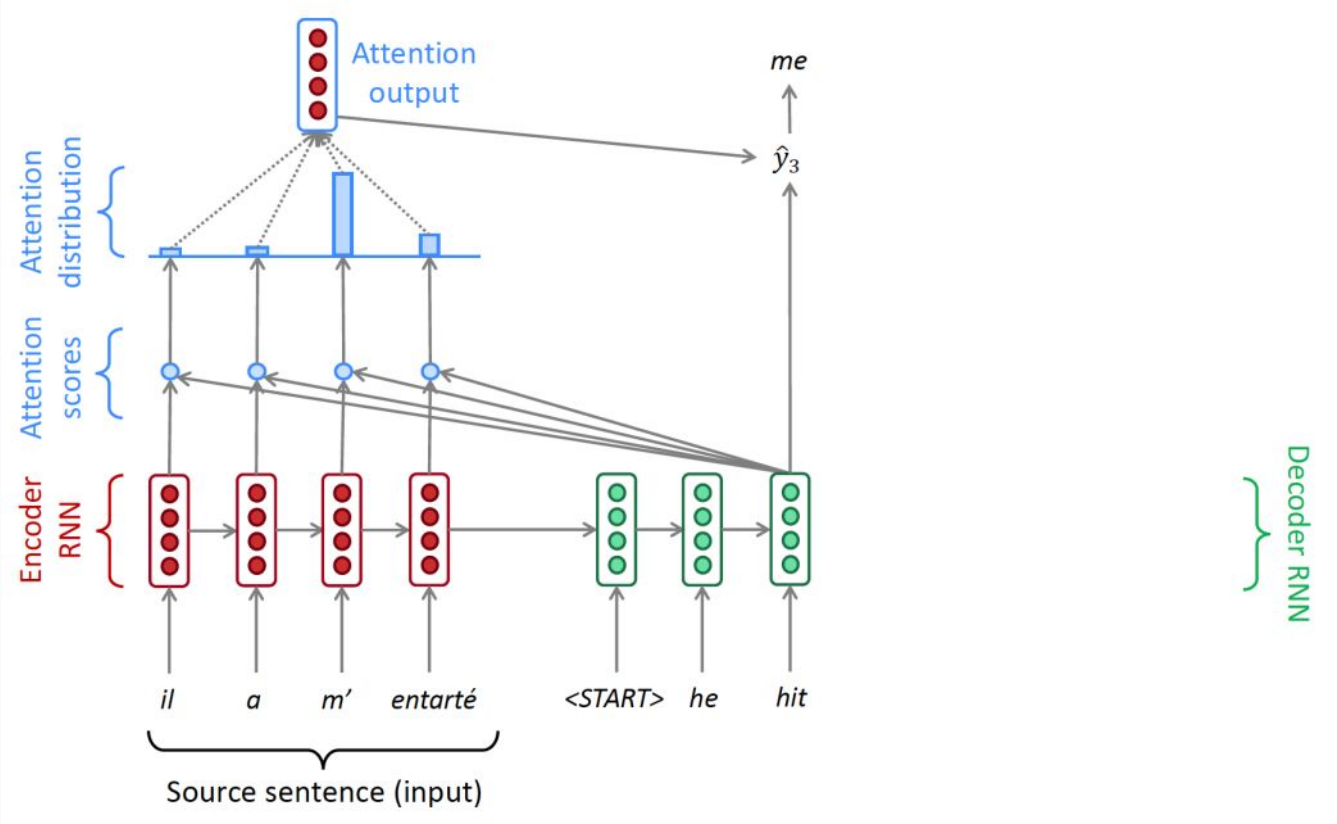
Translation: Attention



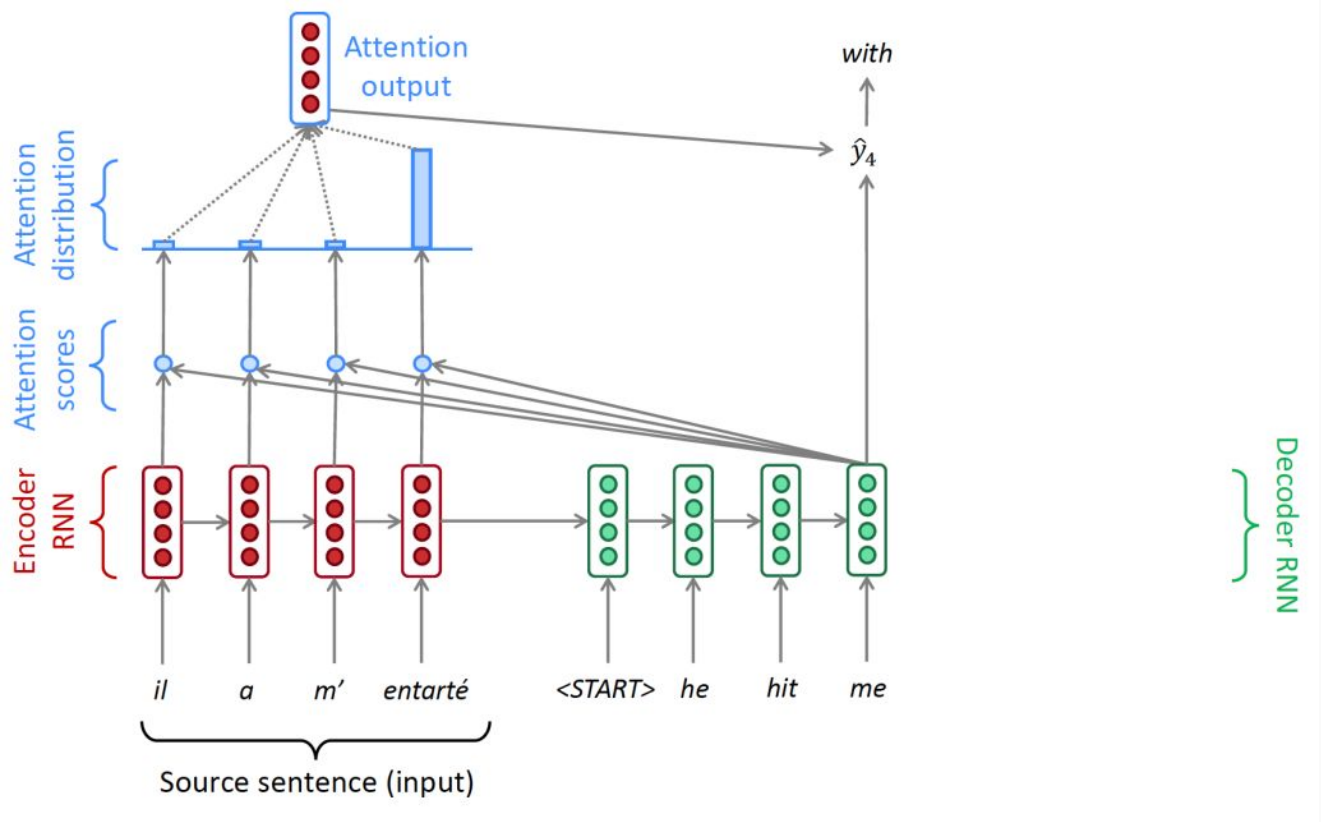
Translation: Attention



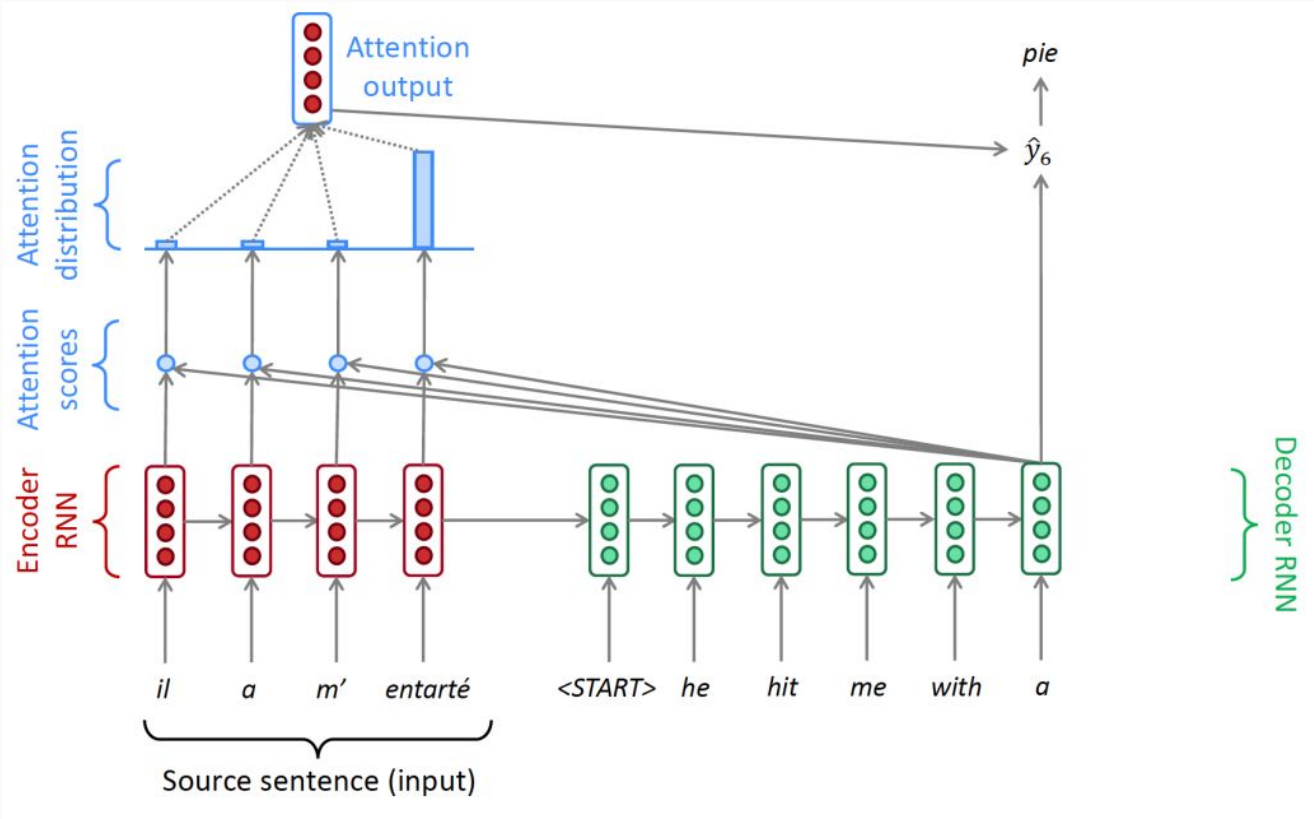
Translation: Attention



Translation: Attention



Translation: Attention



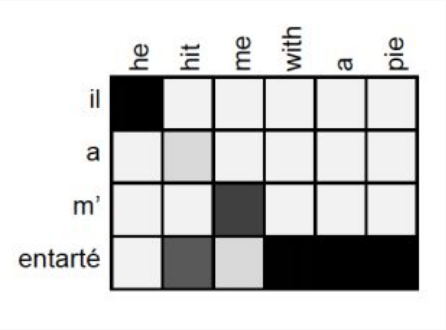
Translation: Attention

- Equations are your friends:
- Hidden states of the encoder → $\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(T)} \in \mathbb{R}^H$
- Hidden states of the decoder → $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}, \ldots, \mathbf{s}^{(T)} \in \mathbb{R}^H$ ← same dimension as h!
- Attention score at time t → $\mathbf{e}^{(t)} = [\, \mathbf{s}^{(t)} \cdot \mathbf{h}^{(1)}, \mathbf{s}^{(t)} \cdot \mathbf{h}^{(2)}, \ldots, \mathbf{s}^{(t)} \cdot \mathbf{h}^{(T)} \,] \in \mathbb{R}^T$ ← dot product
  [**a**, **b**] = concatenation of vector **a** with vector **b**
- Attention output at time t → $\mathbf{a}^{(t)} = \sum_{k \in [1,T]} \mathbf{h}^{(k)} \, \text{softmax}(\, \mathbf{e}^{(t)} \,)(k) \in \mathbb{R}^H$ ← e-wise product
- At each time point t, we then concatenate the attention output and the hidden state of the decoder. We further process this as in the standard seq2seq model

$$[\, \mathbf{a}^{(t)}, \mathbf{s}^{(t)} \,] \in \mathbb{R}^{2H}$$

Translation: Attention

- Attention tends works extremely well:
  - Improves NN performance in NLP
  - Solves the bottleneck problem
  - Helps with vanishing gradients
  - Provides some interpretability



- By inspecting the distribution of the attention in each time step, we can interpret what the decoder was focusing on
  - We can some word alignment for free. This was a very hard problem in traditional statistical models of NLP
  - This is specially cool because we never explicitly designed or trained the NN to do this
  - The NN learned by itself that it is sensible to align words to translate well

Translation: Attention

- There are several typical versions of attention:
- **Dot-product attention:** values and query are dot-multiplied to obtain attention score

$$\mathbf{e}^{(t)} = [\, \mathbf{s}^{(t)} \cdot \mathbf{h}^{(1)}, \mathbf{s}^{(t)} \cdot \mathbf{h}^{(2)}, \ldots, \mathbf{s}^{(t)} \cdot \mathbf{h}^{(T)} \,] \in \mathbb{R}^T$$

- **Multiplicative attention:** the query is linearly transformed with **W**

$$\mathbf{e}^{(t)} = [\, \mathbf{s}^{(t)} \mathbf{W} \mathbf{h}^{(1)}, \mathbf{s}^{(t)} \mathbf{W} \mathbf{h}^{(2)}, \ldots, \mathbf{s}^{(t)} \mathbf{W} \mathbf{h}^{(T)} \,] \in \mathbb{R}^T$$

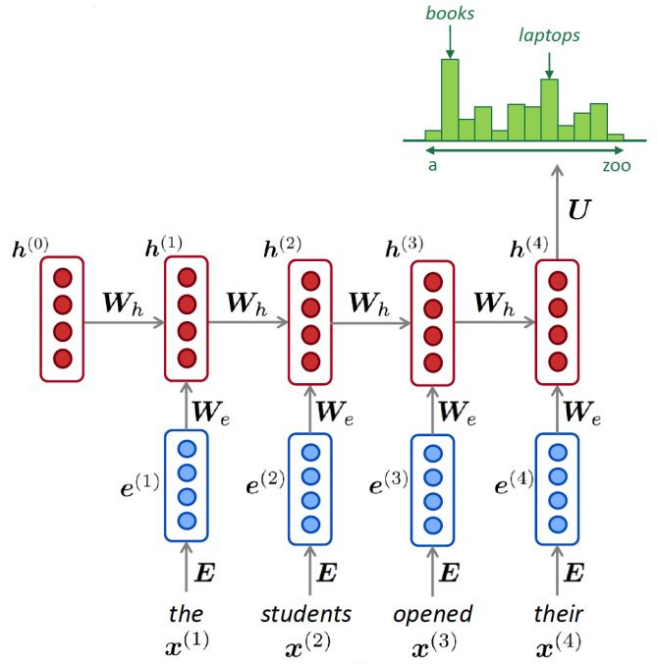
- **Additive attention:** values and query are both linearly transformed by $\mathbf{W}_h$ and $\mathbf{W}_{s'}$ respectively. The result, is averaged with average-weights **v**

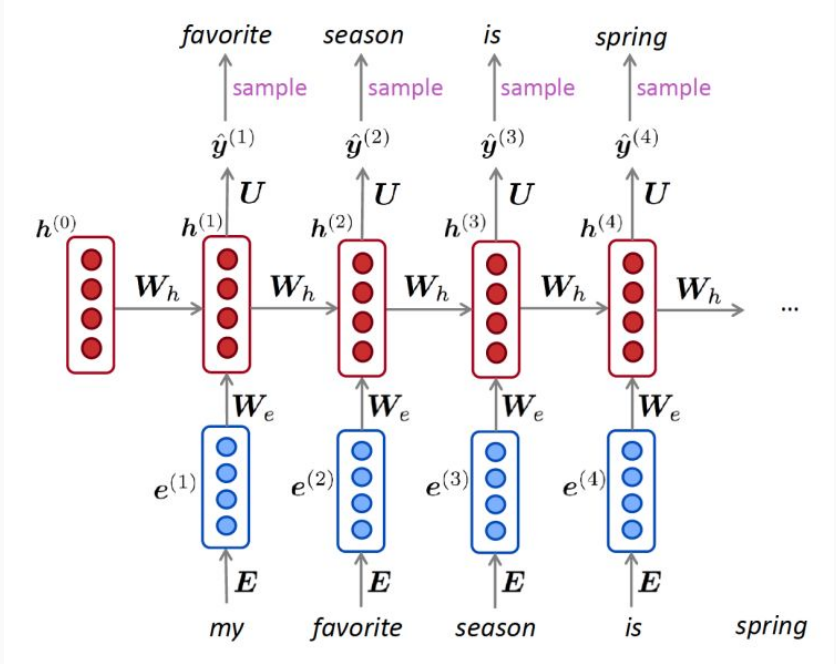
$$\mathbf{e}^{(t)} = [\, \mathbf{v} \cdot \tanh(\, \mathbf{W}_s \mathbf{s}^{(t)} + \mathbf{W}_h \mathbf{h}^{(1)} \,), \mathbf{v} \cdot \tanh(\, \mathbf{W}_s \mathbf{s}^{(t)} + \mathbf{W}_h \mathbf{h}^{(2)} \,), \ldots \,] \in \mathbb{R}^T$$

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

Seq2seq: Generating translation

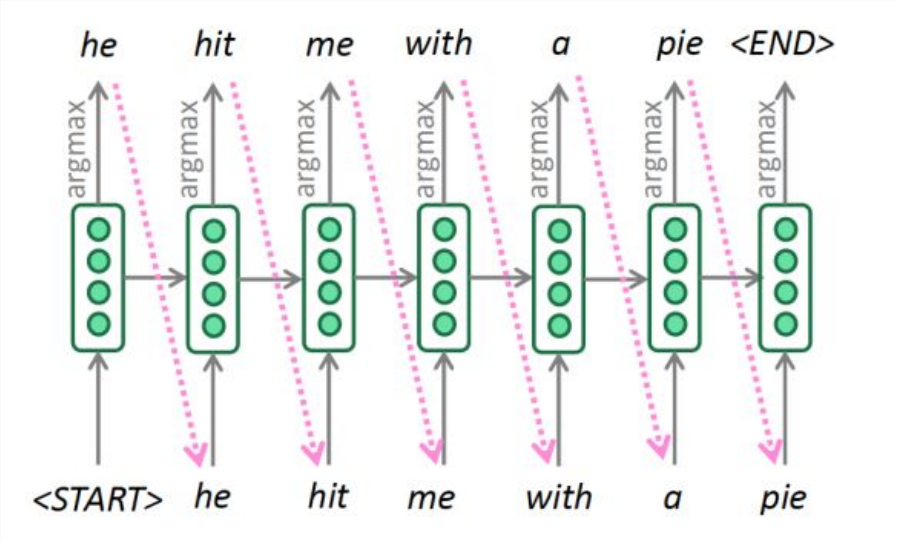
Idea: We do similarly as in the language model of a couple of lectures ago. In each step (i.e. time point 't'), we generate the word that the decoded RNN (i.e. $\mathbf{y}^{(t)}$) gives max probability to. Then we input that word to the RNN (i.e. $\mathbf{x}^{(t+1)} \leftarrow \mathbf{y}^{(t)}$), and generate next word





Seq2seq: Generating translation

Idea: We do similarly as in the language model of a couple of lectures ago. In each (i.e. time point 't'), we generate the word that the decoded RNN (i.e. $y^{(t)}$) gives max probability to. Then we input that word to the RNN (i.e. $e^{(t+1)} \leftarrow y^{(t)}$), and generate as next.



This is called Greedy Decoding, and it is simple to implement but has a few problems.

Seq2seq: Generating translation

➢ **Example:**

- Encoder: $\mathbf{x}^{(0,1,\ldots)}$ = il a m’ entarté (he hit me with a pie)
- Decoder: $\mathbf{x}^{(1)}$ = <START> $y^{(1)}$ = he
- Decoder: $\mathbf{x}^{(2)}$ = he $y^{(2)}$ = hit
- Decoder: $\mathbf{x}^{(3)}$ = hit $y^{(3)}$ = a (ups! no way to go back now)

➢ **Problem:** If we make a mistake at any time point, there is no way to go back un undo the mistake.

Seq2seq: Generating translation

- **Idea:** On each step of the decoder, keep track of the ‘k’ most probable partial translations. We will call these partial translations ‘hypotheses’ → Beam Search of beam size ‘k’
- Having into account that:

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

Then the log-probability of an hypothesis is:

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

- Beam search is not guaranteed to find the optimal solution, but it usually gets close to the optimal while being much lext expensive than exhaustive search

Seq2seq: Generating translation

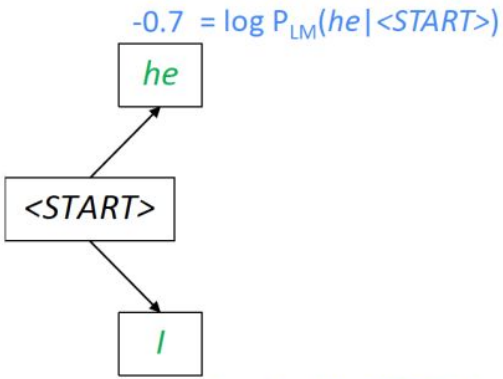
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

<START>

Calculate prob dist of next word

Seq2seq: Generating translation

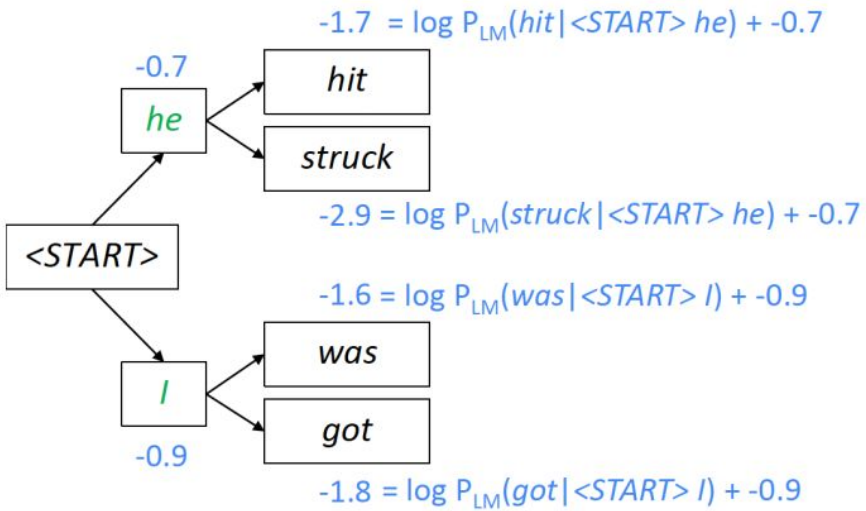
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Take top *k* words and compute scores

Seq2seq: Generating translation

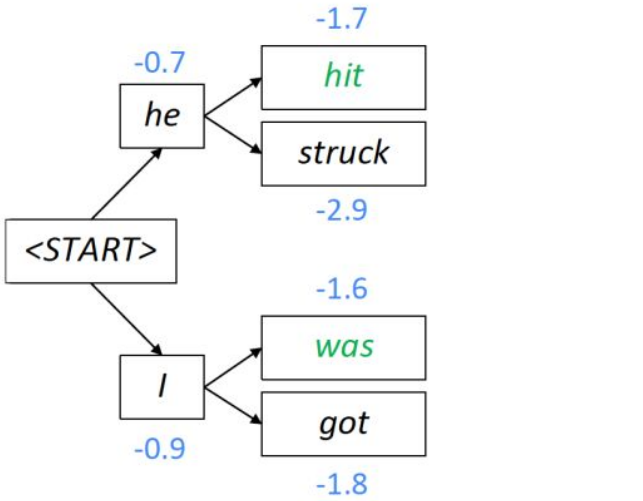
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Seq2seq: Generating translation

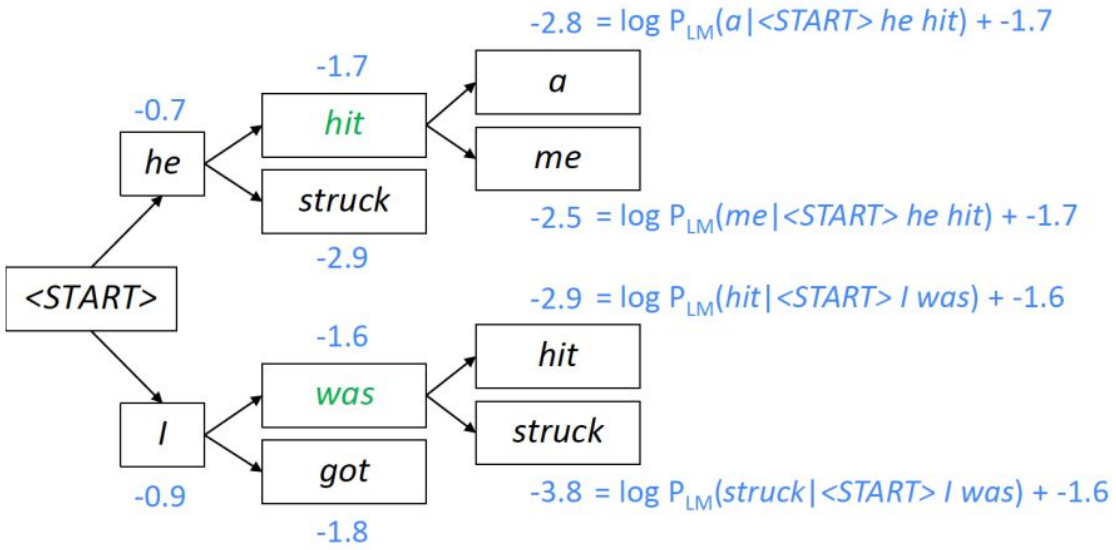
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Of these k^2 hypotheses, just keep k with highest scores

Seq2seq: Generating translation

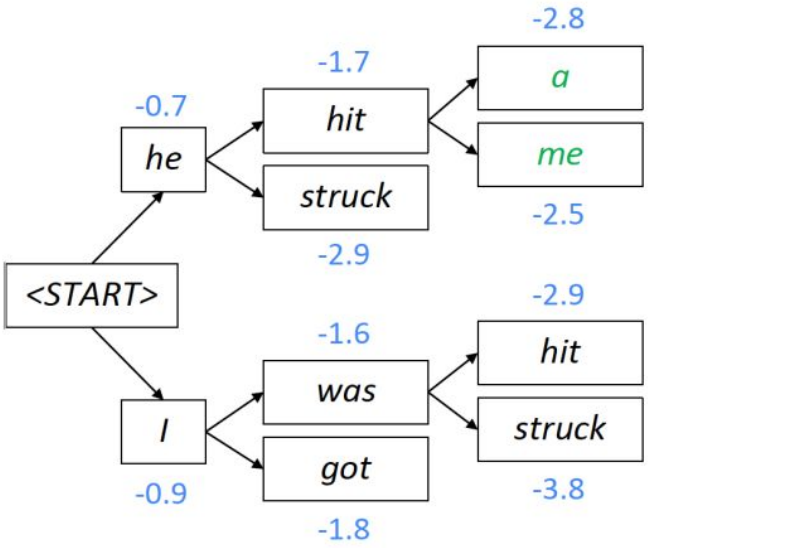
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Seq2seq: Generating translation

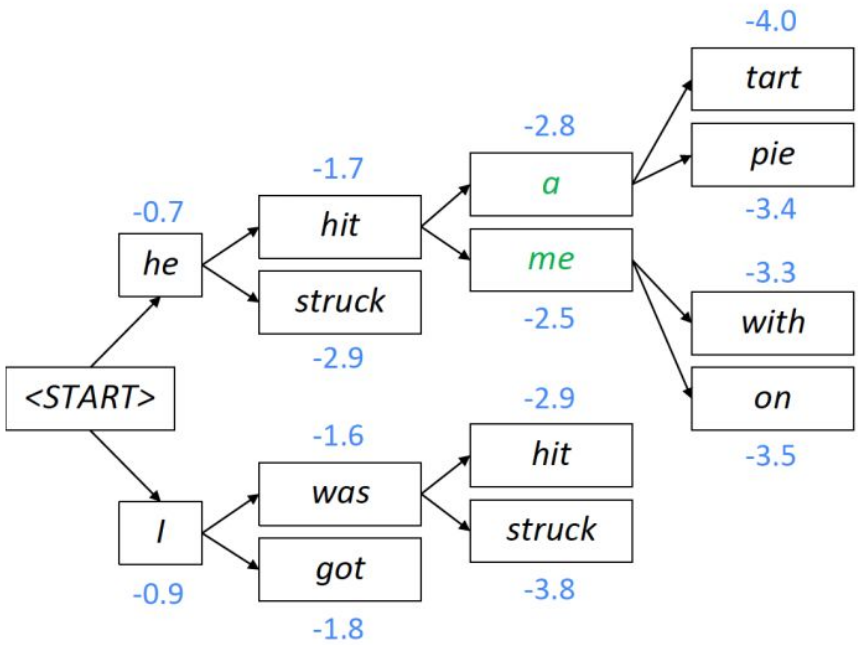
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Of these k^2 hypotheses, just keep k with highest scores

Seq2seq: Generating translation

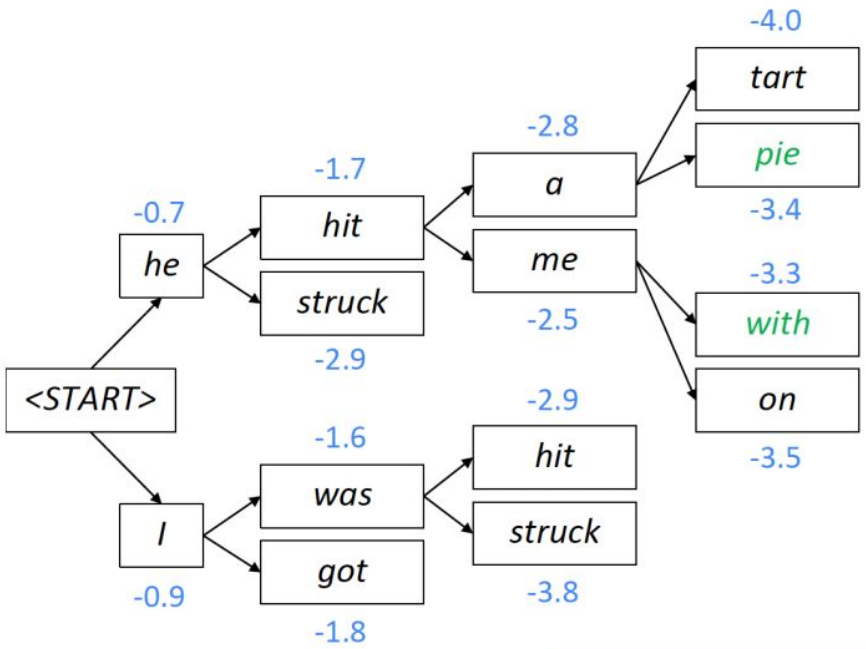
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Seq2seq: Generating translation

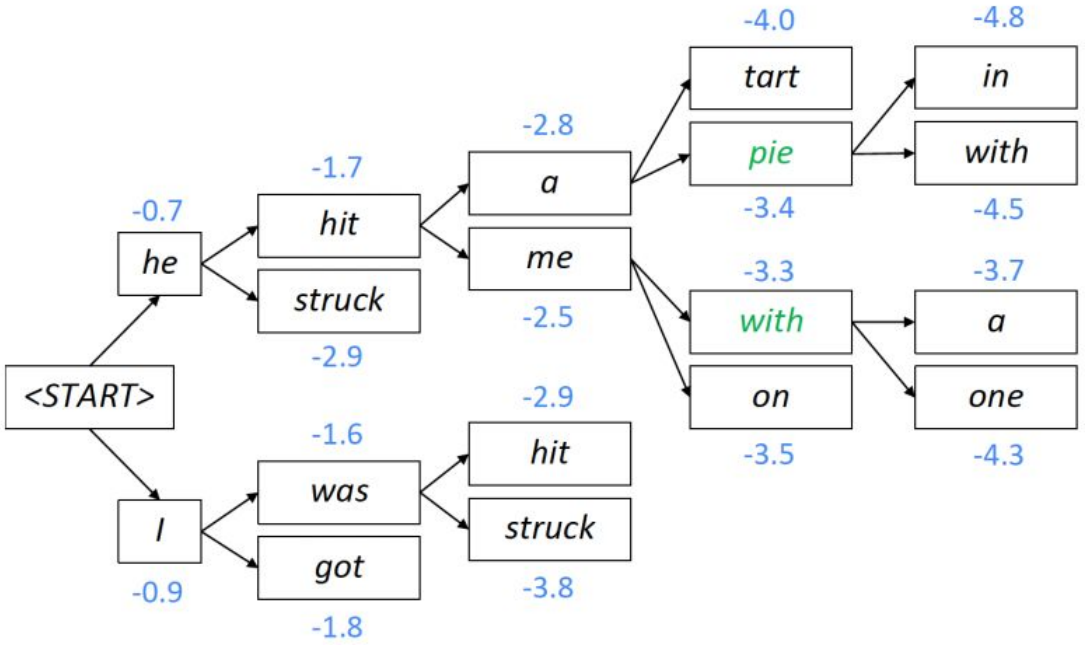
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Of these k^2 hypotheses, just keep k with highest scores

Seq2seq: Generating translation

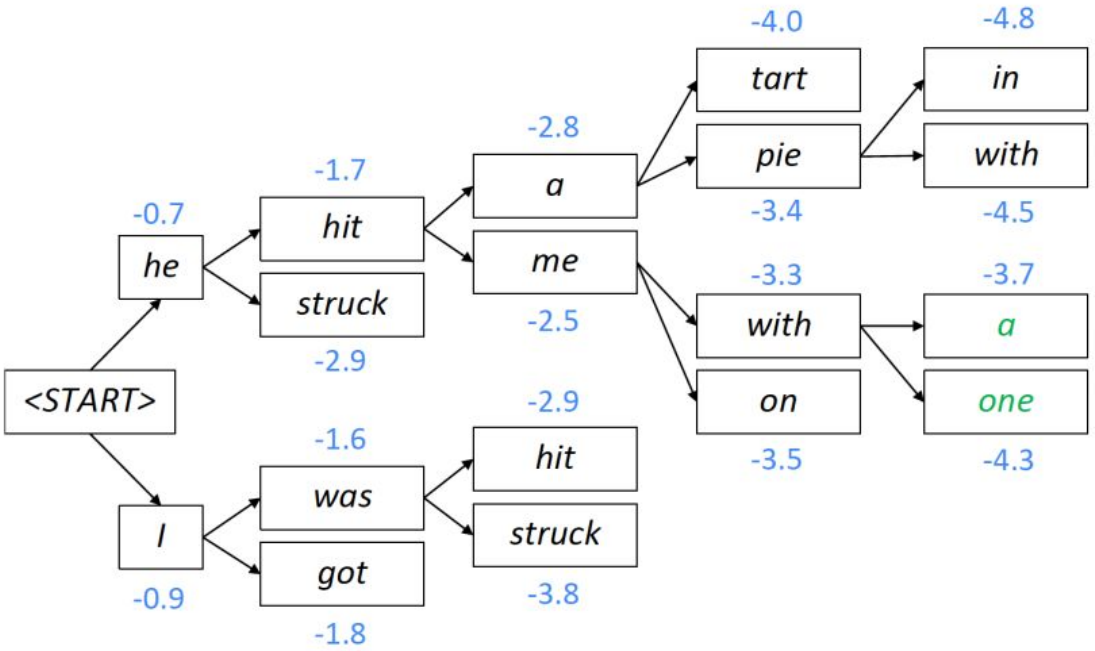
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

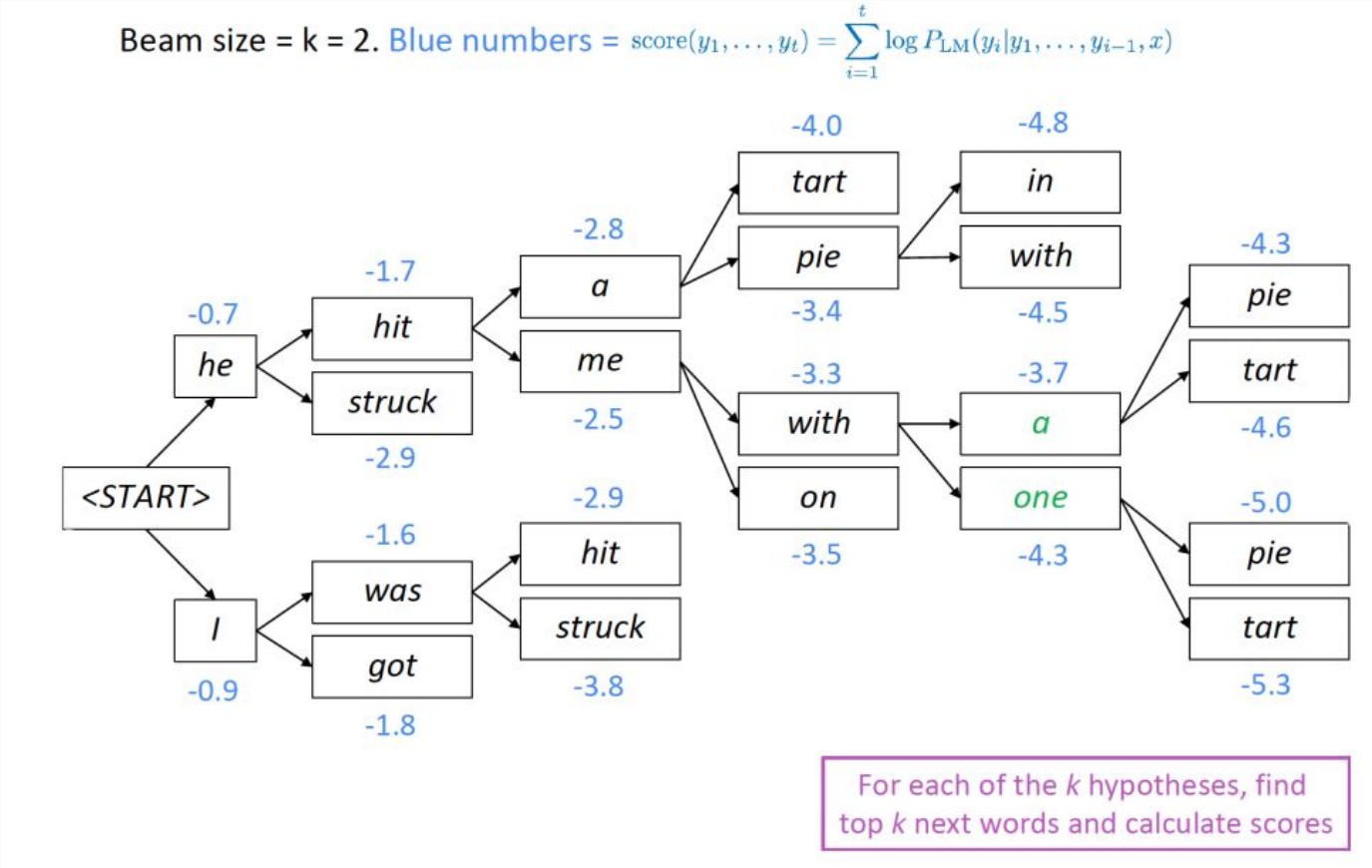
Seq2seq: Generating translation

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

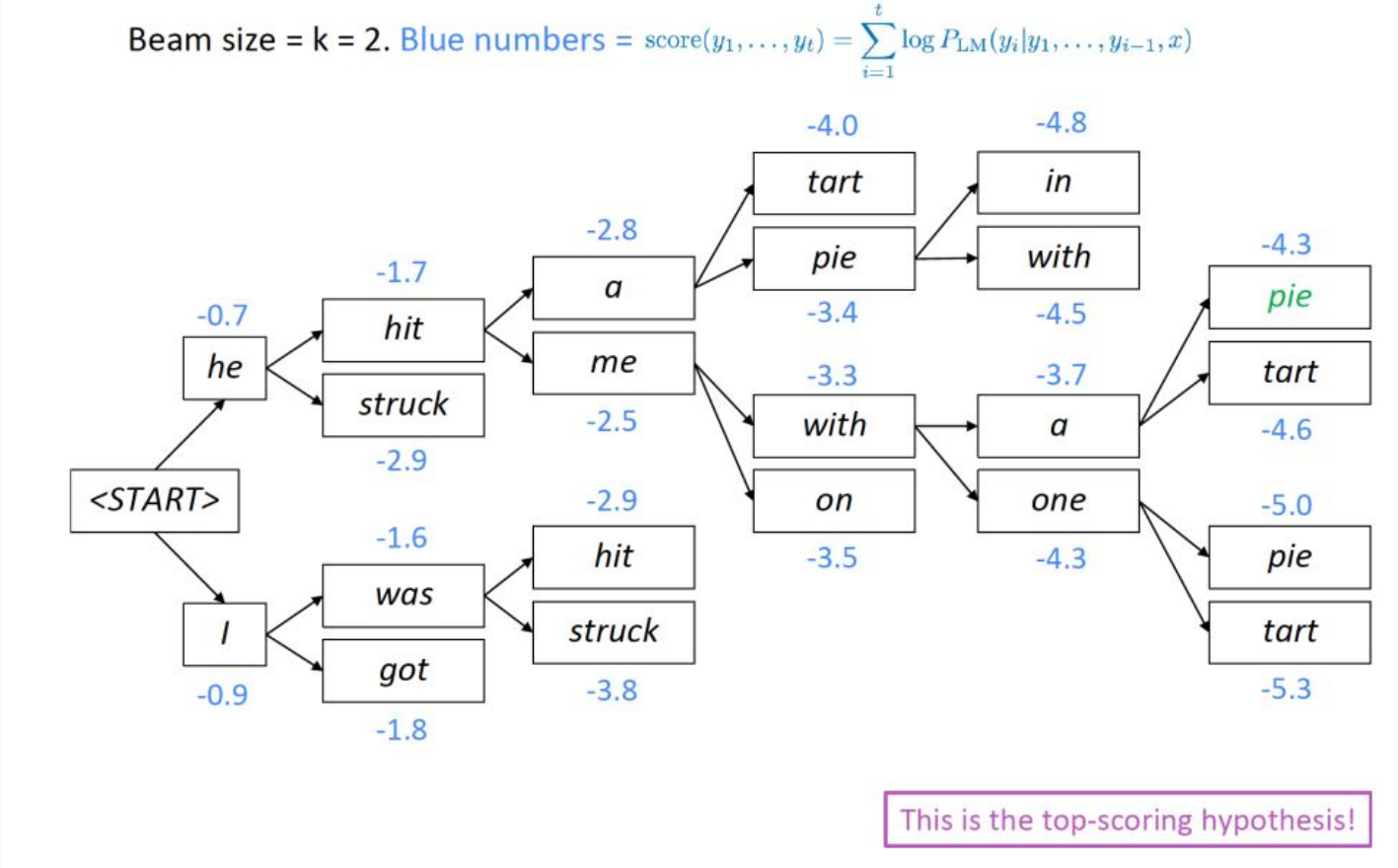


Of these k^2 hypotheses, just keep k with highest scores

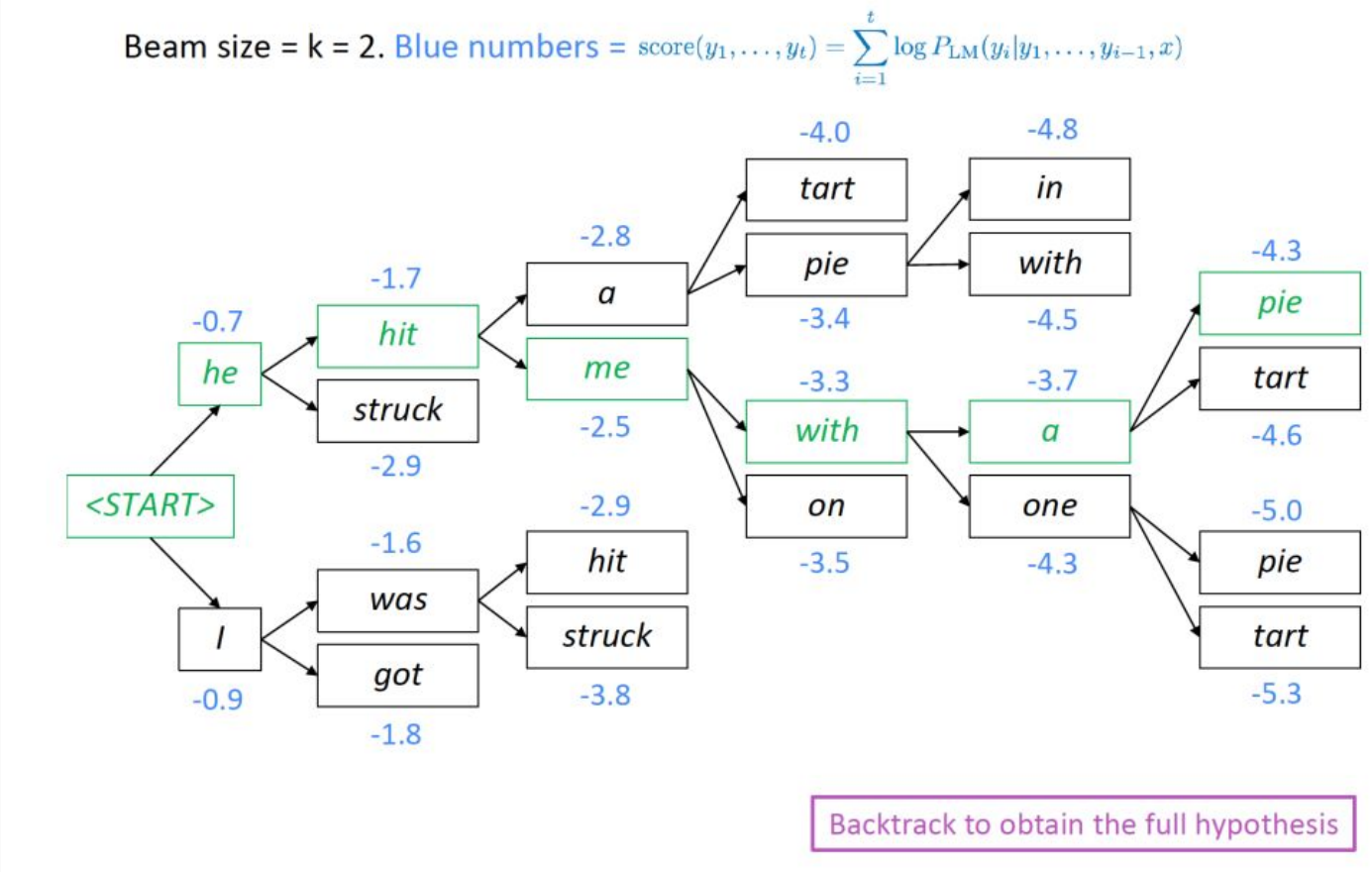
Seq2seq: Generating translation



Seq2seq: Generating translation



Seq2seq: Generating translation



Seq2seq: Generating translation

- But we still need to decide when to stop generating text → Stop Criterion
- Greedy decoding: We generate text until the token <END> is selected
- Beam search: Different hypotheses may produce the <END> token on different time steps (i.e. may produce alternative translations of different lengths). When an hypothesis produces the <END> token, we mark it as completed, put it aside, and continue searching. Then there are different alternative criteria to stop:
  - We stop when we reach time step T, with T being a pre-defined number. Then we select the completed hypothesis that had highest score
  - We stop once we have N completed hypotheses. Then we select the best one
- **Problem:** Longer hypotheses have lower log probabilities $\sum_{i=1}^{t} \log P_{\mathrm{LM}}(y_i|y_1,\ldots,y_{i-1},x)$
- **Solution:** Normalise by length $\frac{1}{t}\sum_{i=1}^{t} \log P_{\mathrm{LM}}(y_i|y_1,\ldots,y_{i-1},x)$

Seq2seq: Generating translation

- Compared to traditional statistical machine translation, NNs have many advantages:
- Better performance:
  - More fluent
  - Better use of context
  - Better use of phrase similarities
- Simpler to build:
  - Traditional statistical models were extremely complex and had many subcomponents. NNs can be trained end-to-end
  - No feature engineering required, only a lot of data
  - Same algorithm, or even same NN, for all language pairs
- Some disadvantages: less interpretable, hard do debug, difficult to control

Translation: Evaluation

- **Idea:** Compare a machine translation to one or several human translations, and measure similarity based on:
  - n-gram precision (n is any value between 1 and 5) = How many times sequences of ‘n’ words in the NN translation appear in the real translation
  - Multiplied by a penalisation for too short translations

  → This is BLEU

- BLEU is good but imperfect:
  - A single piece of text can be translated in different ways
  - So a good translation may still get a poor BLEU score because it has low n-gram similarity with the human translations

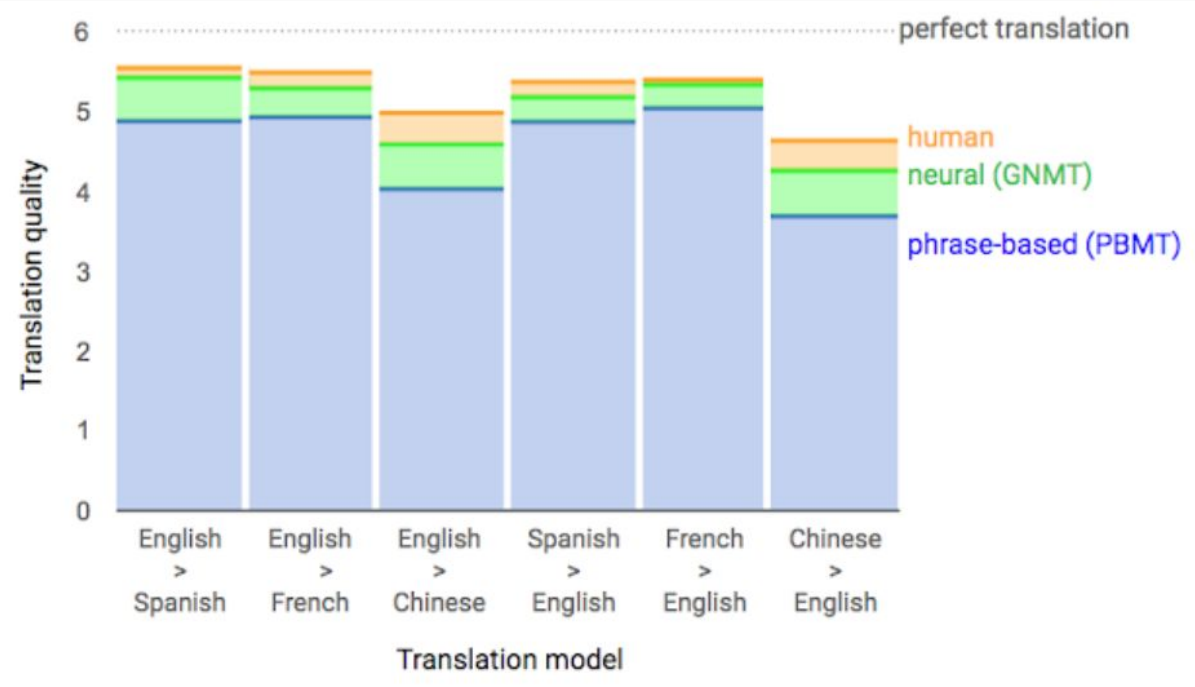
“BLEU: a Method for Automatic Evaluation of Machine Translation", Papineni et al., 2002.
https://www.aclweb.org/anthology/P02-1040.pdf

Translation: Evaluation

- NNs for translation went from a fringe research activity in **2014** to the leading standard method since **2016**
  - **2014**: First seq2seq model published
  - **2016**: Google switches from statistical models to NNs in their translation system
- This is amazing!: Statistical models built by 100s of engineers over many years, outperformed by NNs systems trained by a handful of engineers in a few months
- But the problem is not fully solved yet:
  - Out-of-vocabulary words
  - Using common sense is still hard
  - Models reflect biases existing in the data
  - Maintaining context and long term relationships in long text
  - Language pairs with very few examples

Translation: Evaluation

- NNs for translation went from a fringe research activity in **2014** to the leading standard method since **2016**

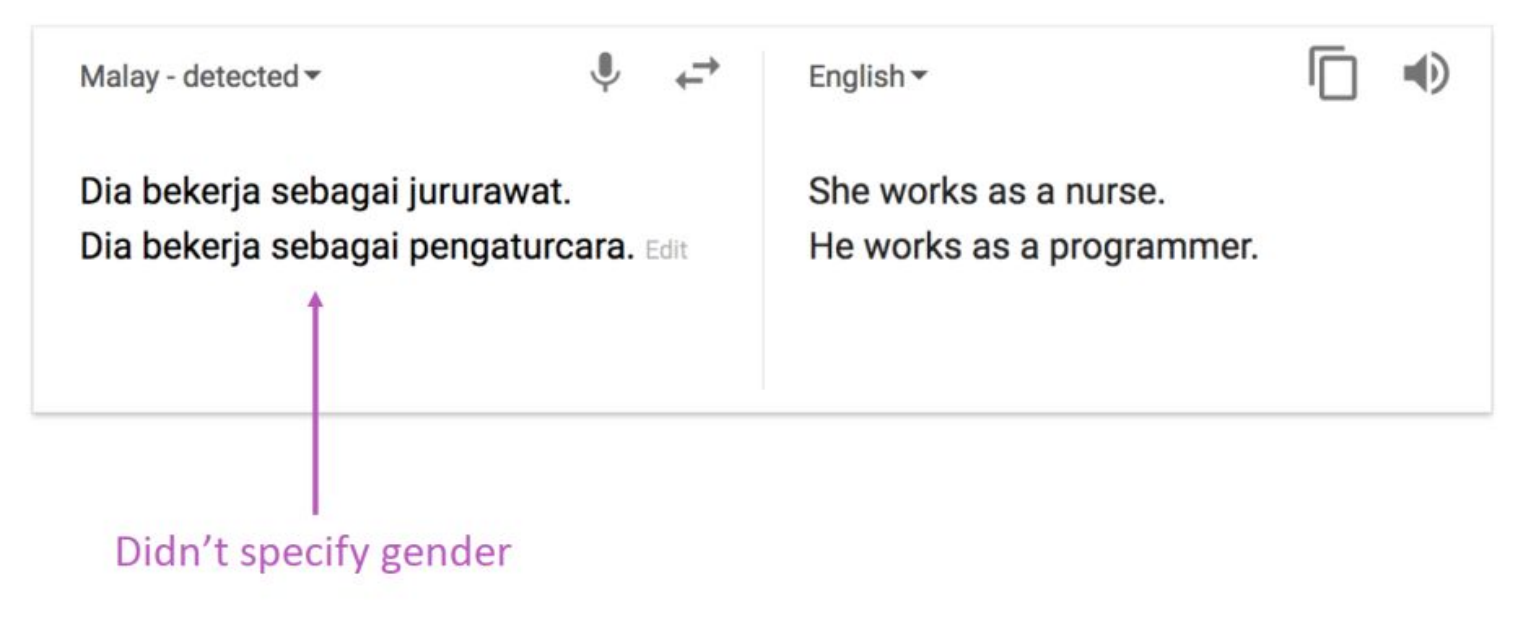


Translation: Evaluation

- But the problem is not fully solved yet: Using common sense is still hard

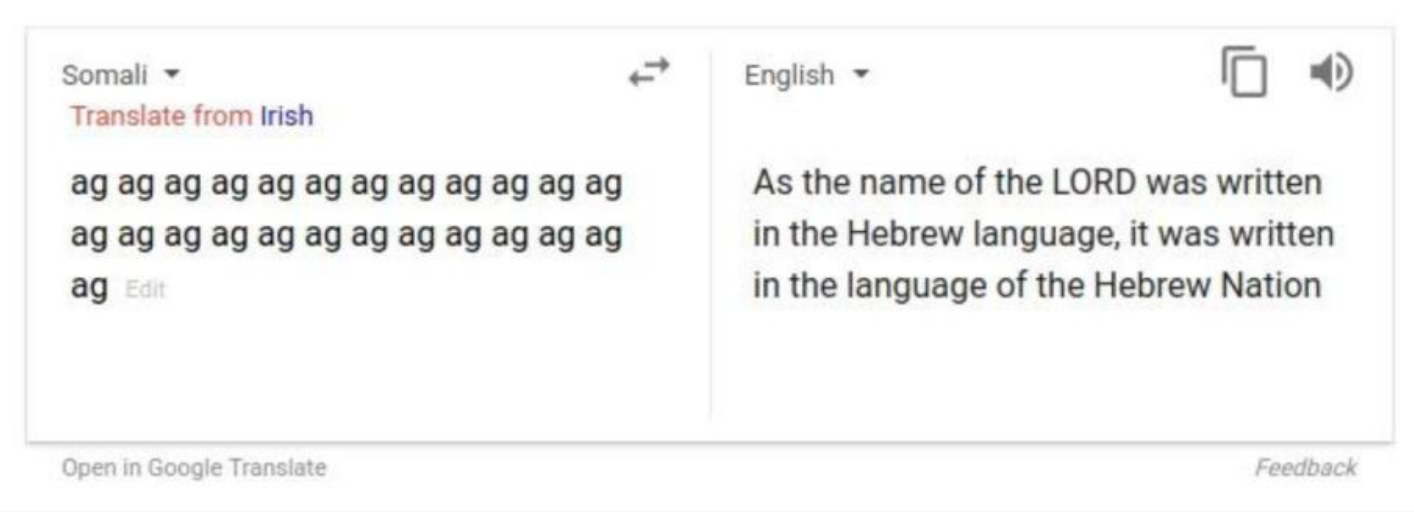
Translation: Evaluation

- But the problem is not fully solved yet: Models reflect biases existing in the data



Translation: Evaluation

- This last one is not much of a problem… but it is quite funny



Course structure

- **Language modelling:** How to represent the meaning of full pieces of text
  - Old technology: N-grams ← Lecture 12 (NLP 4)
  - Recursive NNs language models (RNNs) ← Lecture 12 (NLP 4)
  - Evaluating performance of language models ← Lecture 12 (NLP 4)
  - Vanishing gradients: Problem. Gradient clipping ← Lecture 13 (NLP 5)
  - Improved RNNs: LSTM, GRU, Bidirectional... ← Lecture 13 (NLP 5)
- **Machine translation:** How to translate text
  - Old technology: Georgetown–IBM experiment and ALPAC report ← Lecture 16 (NLP 6)
  - Seq2seq: Greedy decoding, encoder-decoder, beam search ← Lecture 16 (NLP 6)
  - Attention: Simple attention, transformers, reformers ← Lecture 16 (NLP 6)
  - Evaluating performance: BLEU ← Lecture 16 (NLP 6)

Literature

- **Papers** =
- “Statistical Machine Translation", Koehn, 2009. http://www.statmt.org/book/
- “BLEU", Papineni et al., 2002. https://www.aclweb.org/anthology/P02-1040.pdf
- “Sequence to sequence learning with neural networks", Sutskever et al., 2014. https://arxiv.org/pdf/1409.3215
- “Sequence transduction with recurrent neural networks", Graves, 2012. https://arxiv.org/pdf/1211.3711
- “Neural machine translation by jointly learning to align and translate", Bahdanau et al., 2016. https://arxiv.org/pdf/1409.0473
- “Attention and augmented recurrent neural networks", Olah et al., 2016. https://distill.pub/2016/augmented-rnns/
- “Massive exploration of neural machine translation architectures", Britz et al., 2017. https://arxiv.org/pdf/1703.03906
- “Has AI surpassed humans at translation? Not even close!" https://www.skynettoday.com/editorials/state_of_nmt
- “Googles Neural Machine Translation System", Wu et al., 2016. https://arxiv.org/pdf/1609.08144
- “Achieving human parity on automatic Chinese to English news translation", Hassan et al., 2018. https://arxiv.org/pdf/1803.05567
- “Findings of the 2018 Conference on MT", Bojar et al., 2018. http://www.statmt.org/wmt18/pdf/WMT028.pdf