北京航空航天大學
BEIHANG UNIVERSITY

自然语言处理

人工智能研究院

主讲教师 沙磊

北京航空航天大學
BEIHANG UNIVERSITY

Seq2seq与机器翻译

Contents

- 今天内容
  - 介绍新的NLP任务：机器翻译
  - 新的架构：sequence-to-sequence
  - 新的机制：attention

机器翻译的基本方法

- 机器翻译的基本方法
  - 基于规则的机器翻译方法
    - 直接翻译法
    - 转换法
    - 中间语言法
  - 基于语料库的机器翻译方法
    - 基于统计的方法
    - 基于实例的方法
  - 混合式机器翻译方法

目前没有任何一种方法能实现机器翻译的完美理想，但在方法论方面的探索已经使得人们对机器翻译问题的认识更加深刻，而且也确实带动了不少虽不完美但尚可使用的产品问世。

机器翻译的基本方法

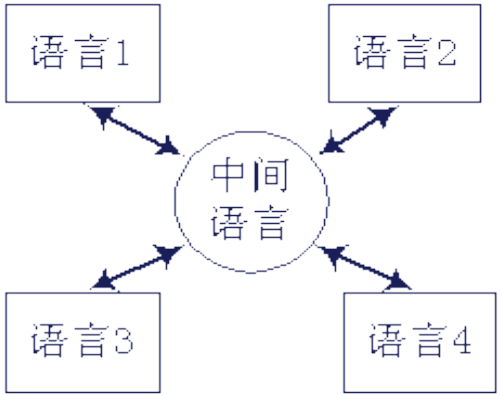
- 20世纪90年代以前，机器翻译方法的主流是基于规则 的方法，因此基于规则的方法也被称为传统的机器翻 译方法。
- 直接翻译法
  - 逐词进行翻译，又称逐词翻译法(word for word translation)
  - 无需对源语言文本进行分析
  - 对翻译过程的认识过度简化，忽视了不同语言之间 在词序、词汇、结构等方面的差异
  - 翻译效果差，属于早期的过时认识，现已无人采用
    - How are you？怎么是你?
    - How old are you？ 怎么老是你?

机器翻译的基本方法

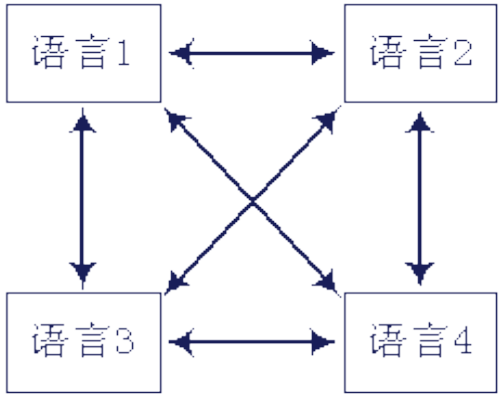
- 中间语言法(interlingua approach)
  - 中间语言(interlingua)是一种中间表达，通常是一种句法-语义表达(syntactic-semantic expression)，中间语言独立于任何具体的自然语言。
  - 源文本经过深层分析得到其对应的中间语言表示。
  - 再由该中间表示生成目标语文本。
  - 翻译过程为两个阶段。

机器翻译的基本方法

- 中间语言定义困难。不同系统采用不同的中间语言， 有的是一种逻辑形式的语言，有的甚至采用类似自 然语言的人工语言，如：荷兰政府支持的DLT计划采用世界语Esperanto做中间语言。
- 中间语言法在理论上非常经济，可有效减少翻译模块的数量。可把$n(n\text{-}1)$个直接翻译模块减少为$2n$个翻译模块。



(1)



(2)

机器翻译的基本方法

- 把任何一种自然语言翻译成为一种独立的中间语言，需要深层次的语言分析和生成技术，目前没有特别成功的基于中间语言的机器翻译系统。

- 转换法(transfer approach)
  - 分析源语言文本，得到源语言的内部表达
  - 将源语言内部表达转换成目标语内部表达
  - 根据目标语内部表达生成目标语文本
  - 翻译过程分成三个阶段

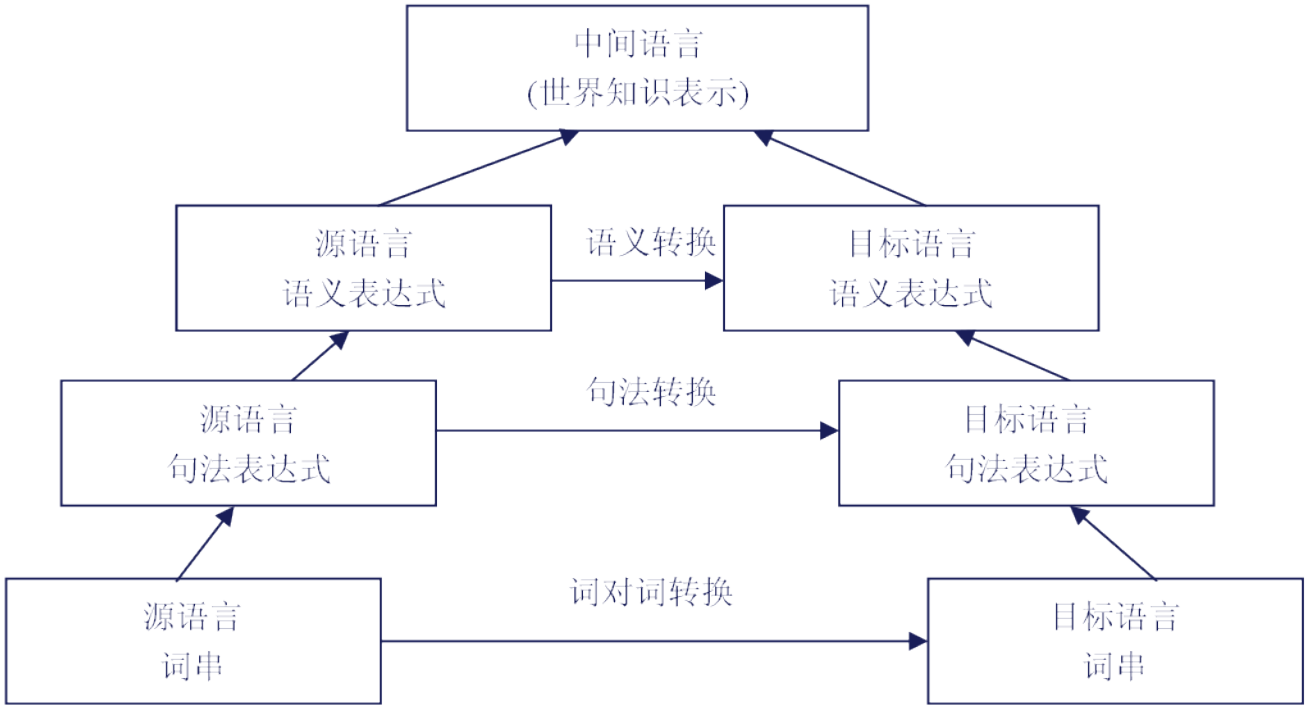
机器翻译的基本方法

- 不同系统采用不同层次内部表示，例如浅层句法表示或深层句法语义表示。
- 商业上较为成功的方法，目前绝大部分商品化机器翻 译系统采用转换式机器翻译方法。

- 基于知识的机器翻译方法(knowledge-based machine translation)
  - 20世纪70年代, 受人工智能、知识工程发展的影响, 而 提出。
  - 强调对源语言进行更为彻底的分析和理解。
  - 不仅进行深层语言学分析，还需要进行世界知识(world knowledge)(常识与领域知识)的显式处理。
  - 需要建立对语言理解有益的本体知识库(ontology)。

机器翻译的基本方法

- 研制代价昂贵，没有特别成功的案例。



基于规则的翻译方法图示

机器翻译的基本方法

- 20世纪80年代中后期，基于语料库的机器翻译技术得到越来越多的关注。
  - 试图避开知识库建设的困难
  - 试图回避对源语言进行深层语言分析
  - 翻译知识主要来自双语平行语料库
  - 基于实例的翻译通过模仿实例库中已有的译文基于类比的策略进行翻译。
  - 基于统计的机翻译通过建立、训练统计翻译模型、 并进而基于统计模型进行翻译。
- 考虑到这些方法背后的哲学背景，也常把基于规则的方法称为理性主义(rationalism)方法，而把基于语料库的方法称为经验主义(empiricism)方法。

传统机器翻译方法

Machine Translation

- **Machine Translation (MT)** is the task of translating a sentence x from one language (the source language) to a sentence y in another language (the target language)

x: *L'homme est né libre, et partout il est dans les fers*

y: *Man is born free, but everywhere he is in chains*

The early history of MT: 1950s

- 机器翻译研究始于1950年代早期。那时并不比计算器更先进。
- 与一些基础性的研究紧密相关：自动机，形式语言，概率论，信息论
- 机器翻译被军方（PS：US Army）重金资助，基本方法还是基于规则的单词替换。
- 人类语言极其复杂，远远不是规则方法能解决的，不同语言的特性也大不相同
- 人们对自然语言语法，语义，语用的了解还很少
- 很快，MT的研究陷入了泥潭。。。

1990s-2010s: Statistical Machine Translation

- 核心思想：从数据中学习一个概率模型
- 假如训练法语到英语的翻译模型
- 给出法语句子x，我们想找到最好的英语句子y

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

- 利用贝叶斯法则，将其拆成两部分

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

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.

$$\arg\max_y P(y|x)$$

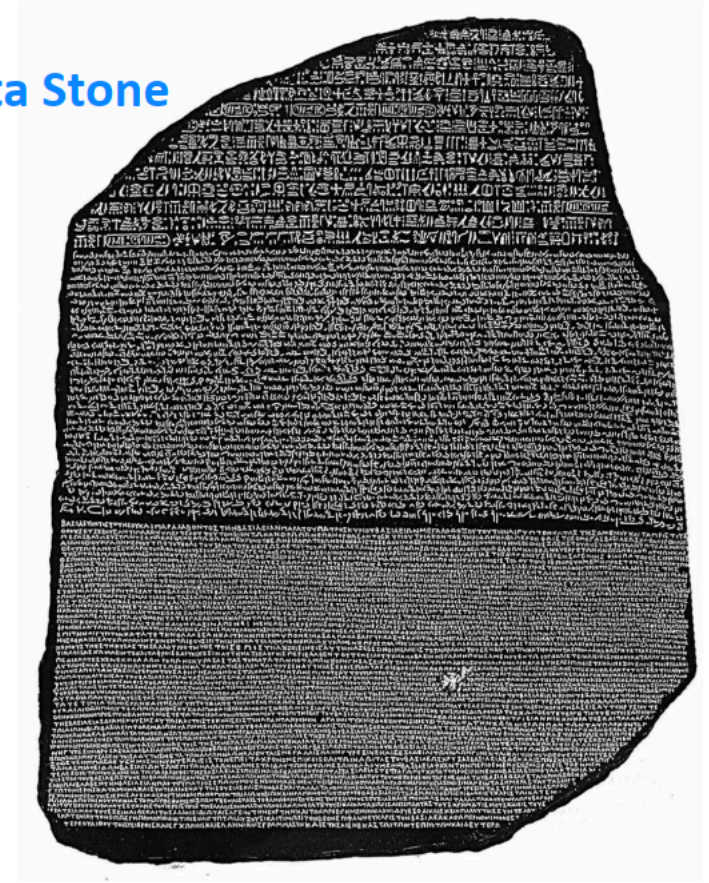
$$= \arg\max_y \frac{P(x|y)P(y)}{P(x)}$$

给定的，所以已经确定

1990s-2010s: Statistical Machine Translation

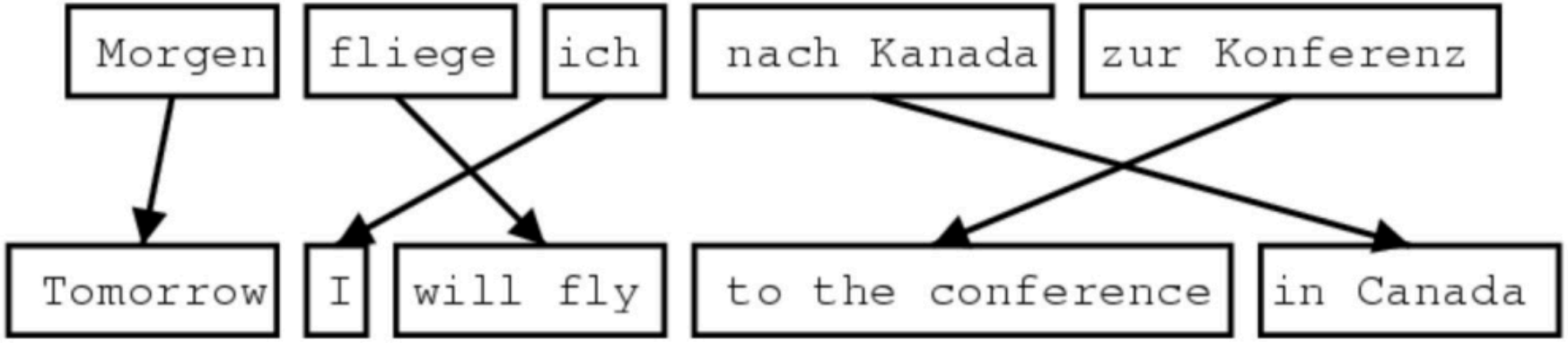
- 问题：如何去学习翻译模型 $P(x|y)$
- 首先，需要大规模平行数据
  - 人工翻译的法语-英语数据对

The Rosetta Stone



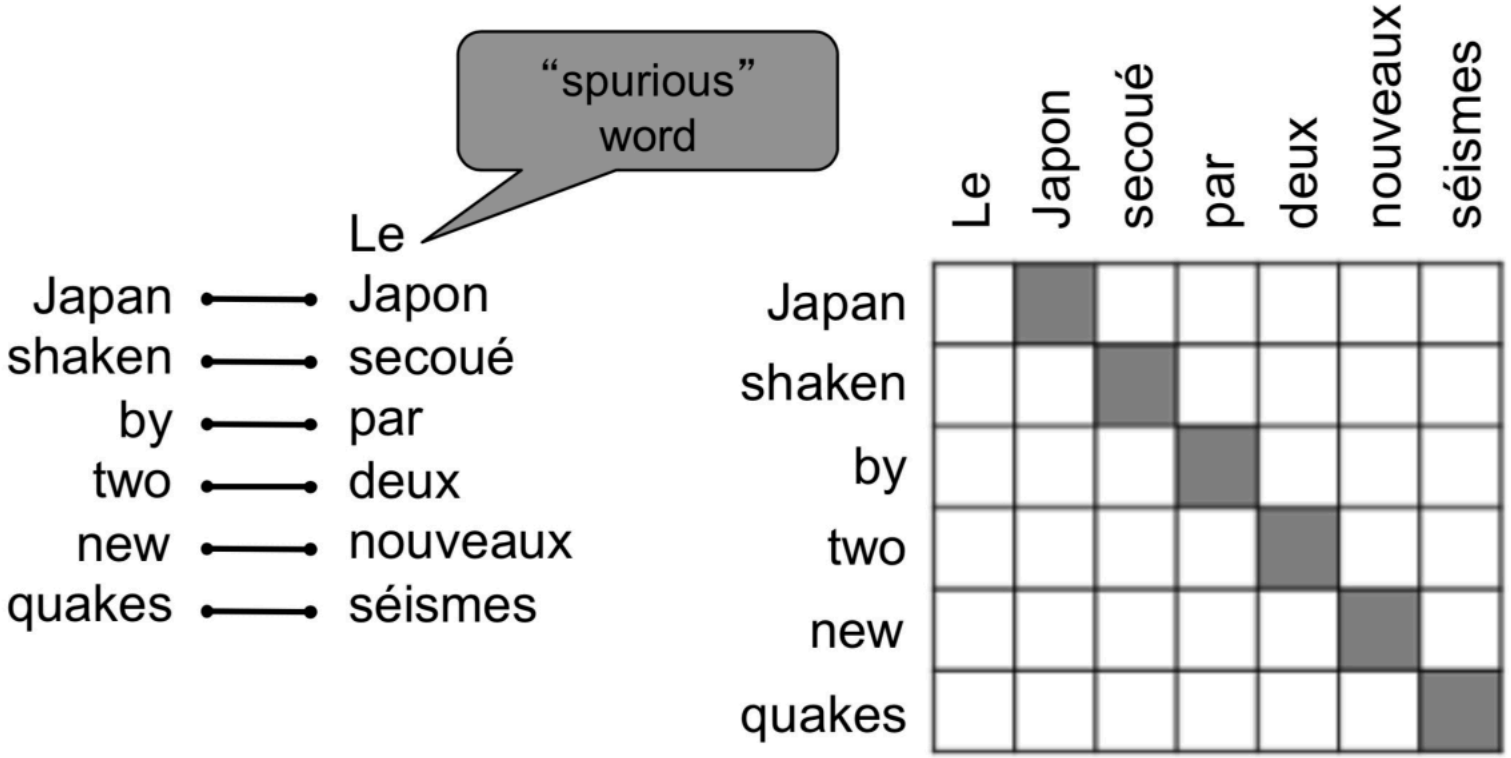
Learning alignment for SMT

- 问题：如何从平行语料中学习翻译模型 $P(x|y)$
- 拆分问题：可以往模型中引入一个隐变量an $P(x, a|y)$
- a代表一个对齐方案：源句子和目标句子的一个词级别的对应关系



What is alignment?

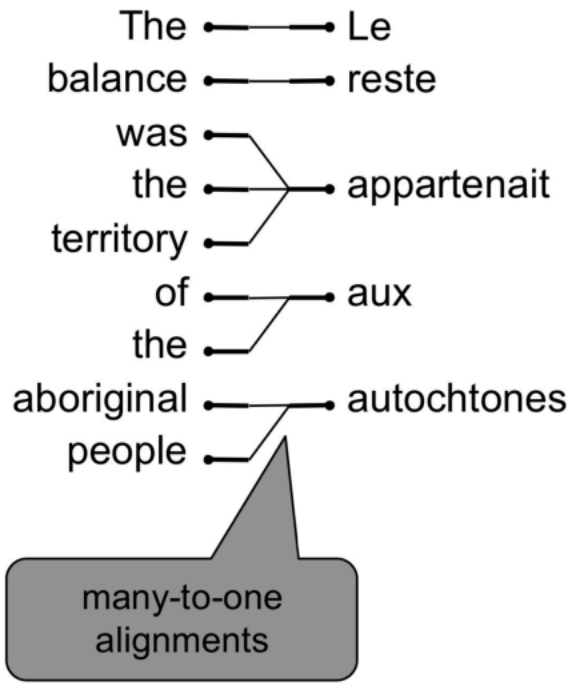
- Alignment is the correspondence between particular words in the translated sentence pair.
- Typological differences between languages lead to complicated alignments!
- Note: Some words have no counterpart

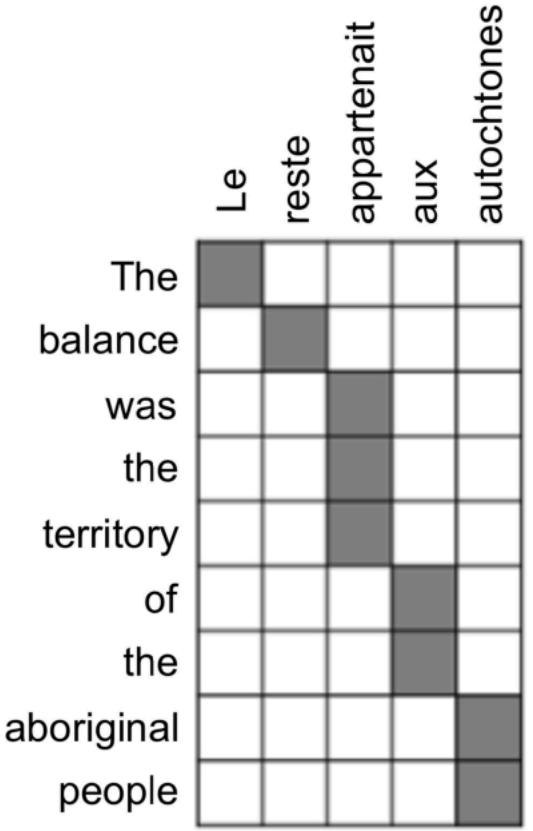


Examples from: "The Mathematics of Statistical Machine Translation: Parameter Estimation", Brown et al, 1993. http://www.aclweb.org/anthology/J93-2003

Alignment is complex

- Alignment can be many-to-one

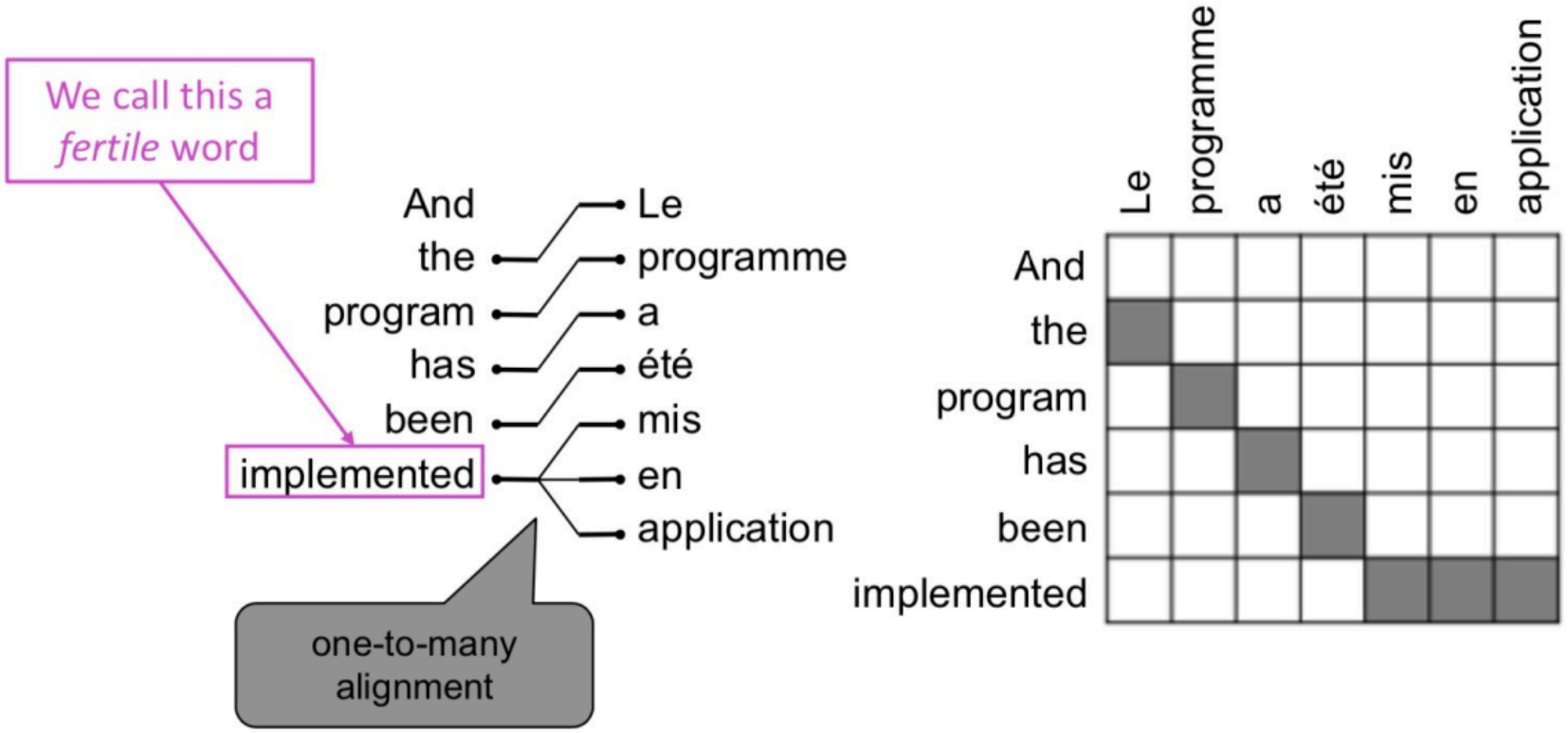




Examples from: “The Mathematics of Statistical Machine Translation: Parameter Estimation”, Brown et al, 1993. http://www.aclweb.org/anthology/J93-2003

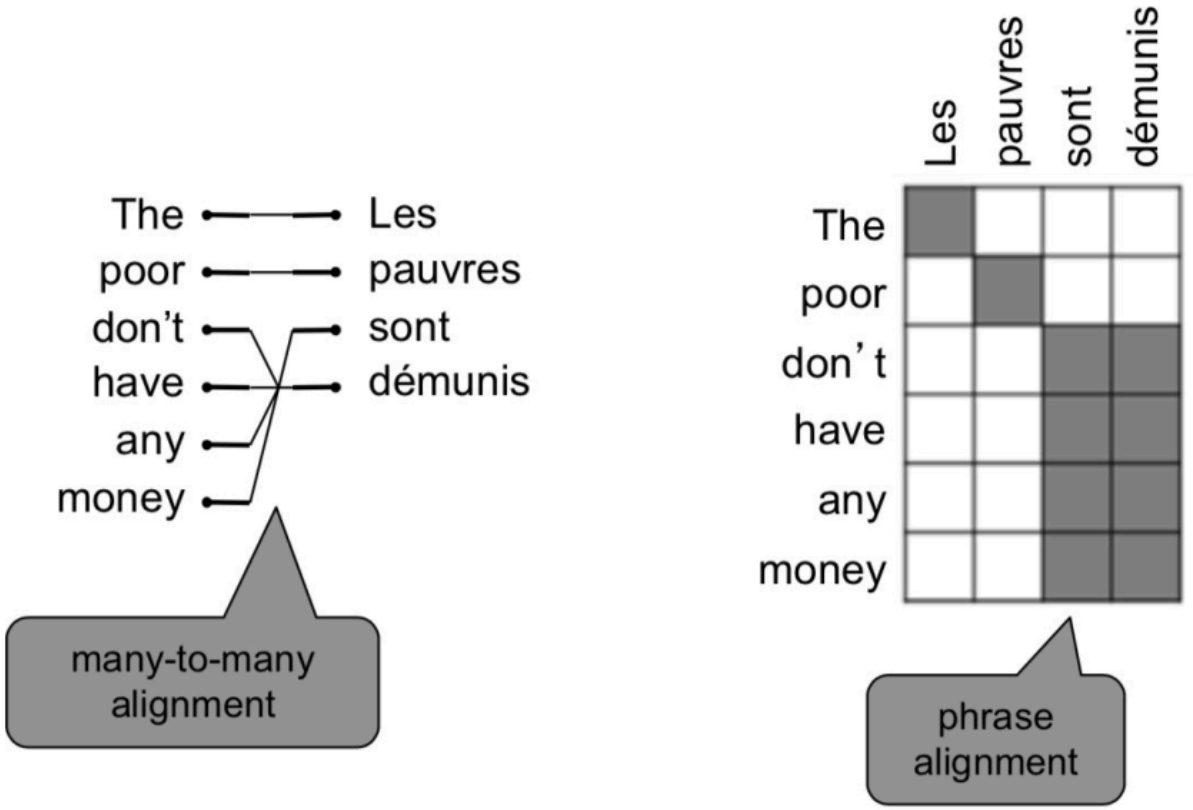
Alignment is complex

- Alignment can be one-to-many



Alignment is complex

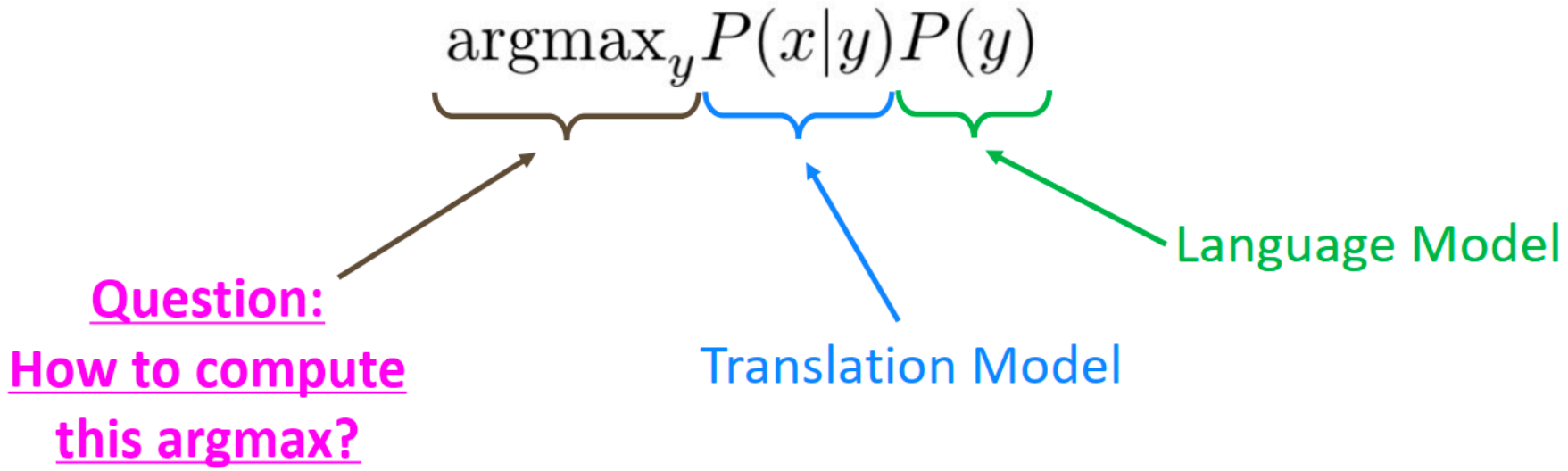
- Alignment can be many-to-many (phrase-level)



Learning alignment for SMT

- 在学习 $P(x, a|y)$ 的过程中，需要学习如下几个方面：
  - 为特定的词对齐方案的概率进行建模
  - 为特定的词拥有多个对应的目标语言词的概率进行建模
- 对齐方案a属于隐变量。在数据标注中没有体现
  - 带有隐变量的模型需要我们用一些特殊的学习算法来计算参数的分布
  - 比如：EM算法

Decoding for SMT



- 枚举每一个可能的y并计算概率？--过于耗时耗力
- 答案：在模型中加入强独立性假设，用动态规划来寻找全局最优。（Viterbi算法）
- 这个过程叫做解码（decoding）

1990s-2010s: Statistical Machine Translation

- SMT 曾是一个很大的研究领域
- 能做到很高的性能的系统极其复杂
  - 很多重要的细节我们没有提及
- SMT系统里包括很多单独设计的子模块
  - 需要大量特征工程
    - 为了捕捉大量语言现象来设计特征
  - 需要维持外部资源
    - 同义的词组表
  - 需要大量人类的工作量

神经机器翻译方法

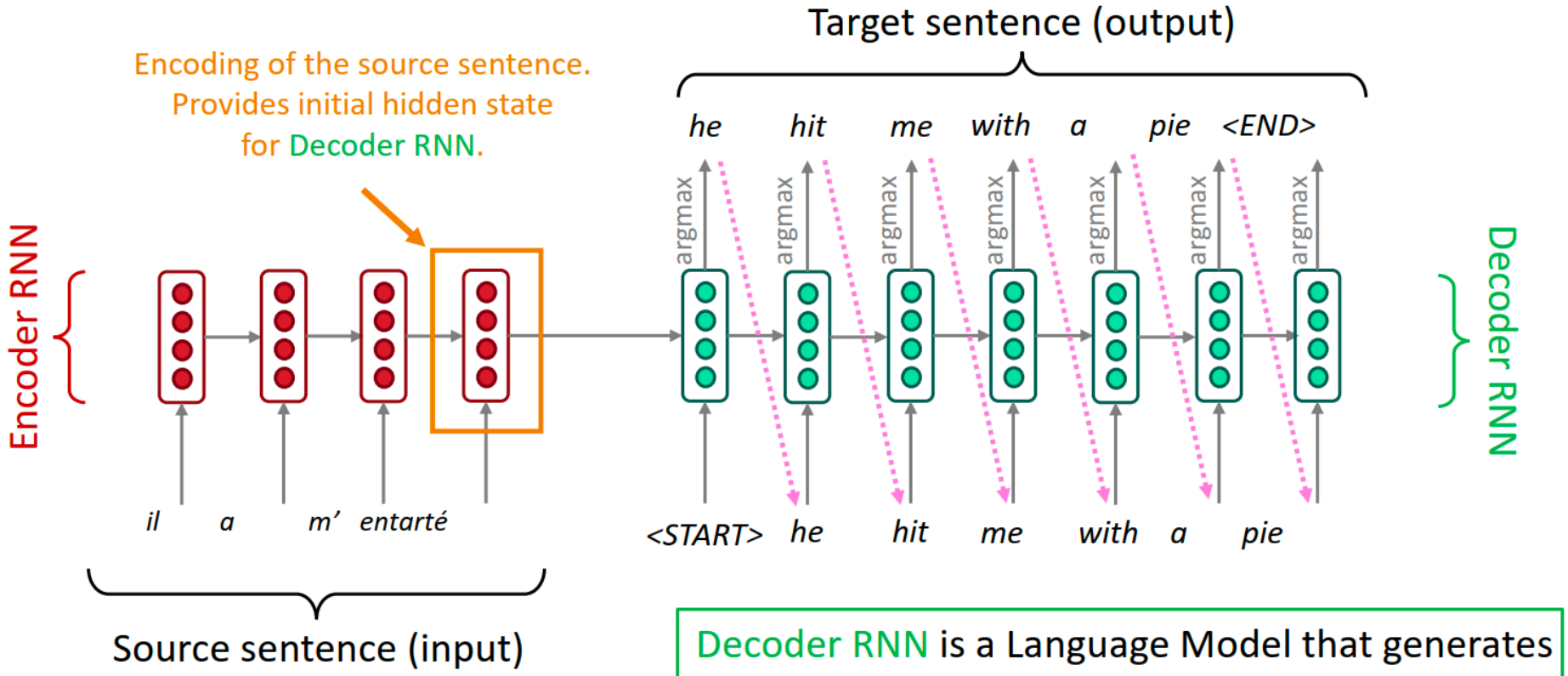
What is Neural Machine Translation?

- Neural Machine Translation (NMT) is a way to do Machine Translation with a single end-to-end neural network

- The neural network architecture is called a sequence-to-sequence model (aka seq2seq) and it involves two RNNs

Neural Machine Translation (NMT)

The sequence-to-sequence model



Encoder RNN produces an encoding of the source sentence.

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

Note: This diagram shows **test time** behavior: decoder output is fed in ⋯→ as next step's input

Sequence-to-sequence is versatile!

- The general notion here is an encoder-decoder model
  - One neural network takes input and produces a neural representation
  - Another network produces output based on that neural representation
  - If the input and output are sequences, we call it a seq2seq mode
- 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)

Neural Machine Translation (NMT)

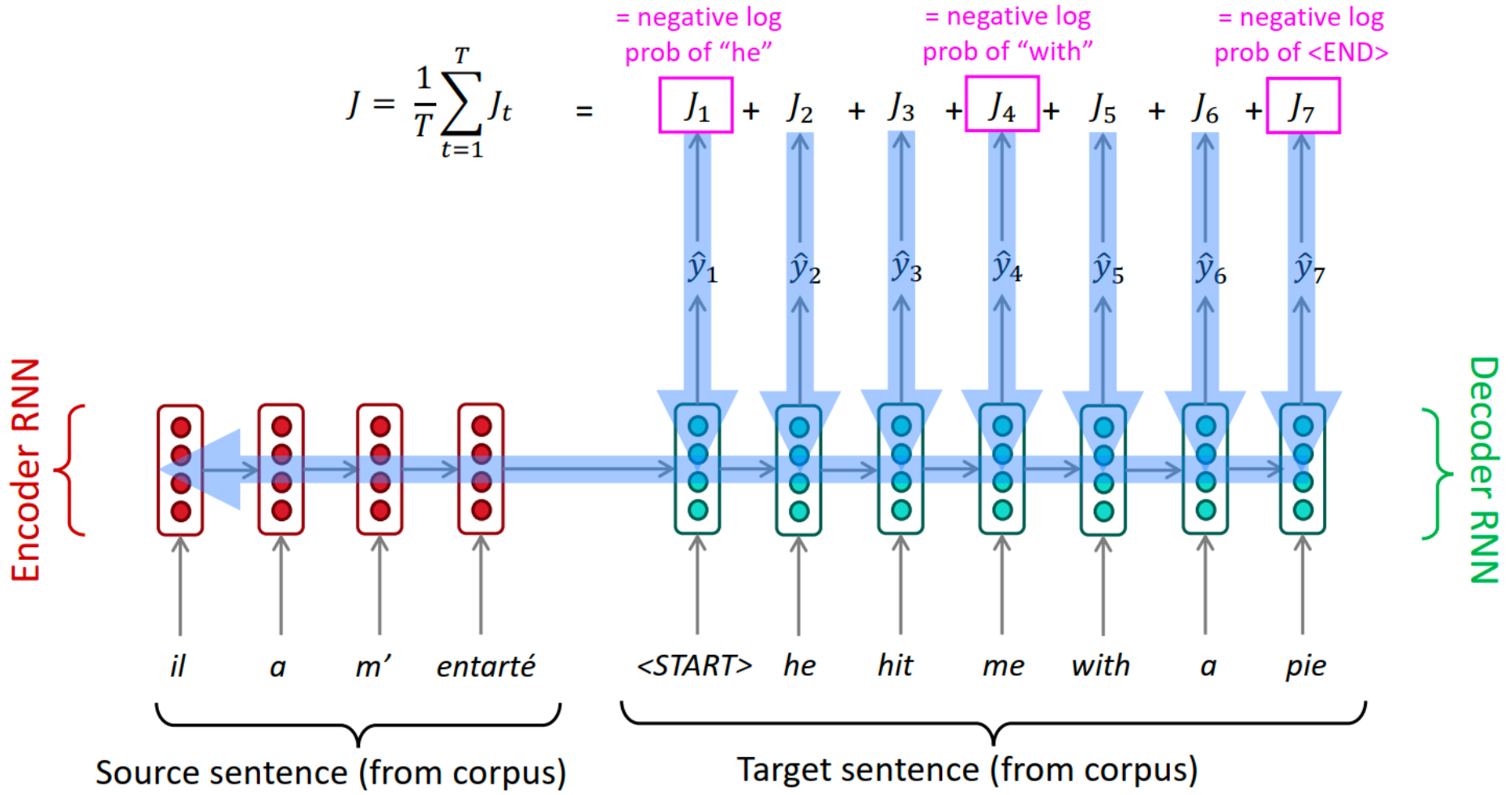
- The sequence-to-sequence model is an example of a Conditional Language Model
  - Language Model because the decoder is predicting the next word of the target sentence y
  - Conditional because its predictions are also conditioned on the source sentence x
- NMT directly calculates $P(y|x)$

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

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

- **Question**: How to train an NMT system?
- **(Easy) Answer**: Get a big parallel corpus…
  - But there is now exciting work on "unsupervised NMT", data augmentation, etc

Training a Neural Machine Translation system

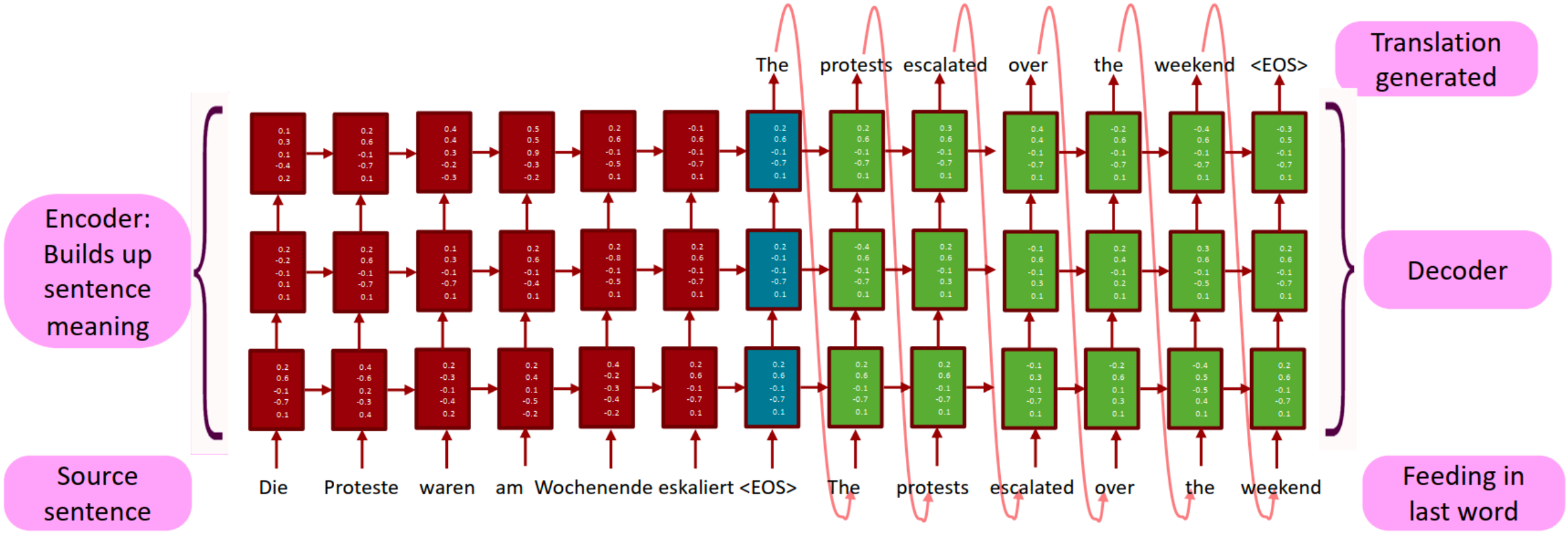


Seq2seq is optimized as a **single system.** Backpropagation operates *"end-to-end"*.

Multi-layer deep encoder-decoder machine translation net

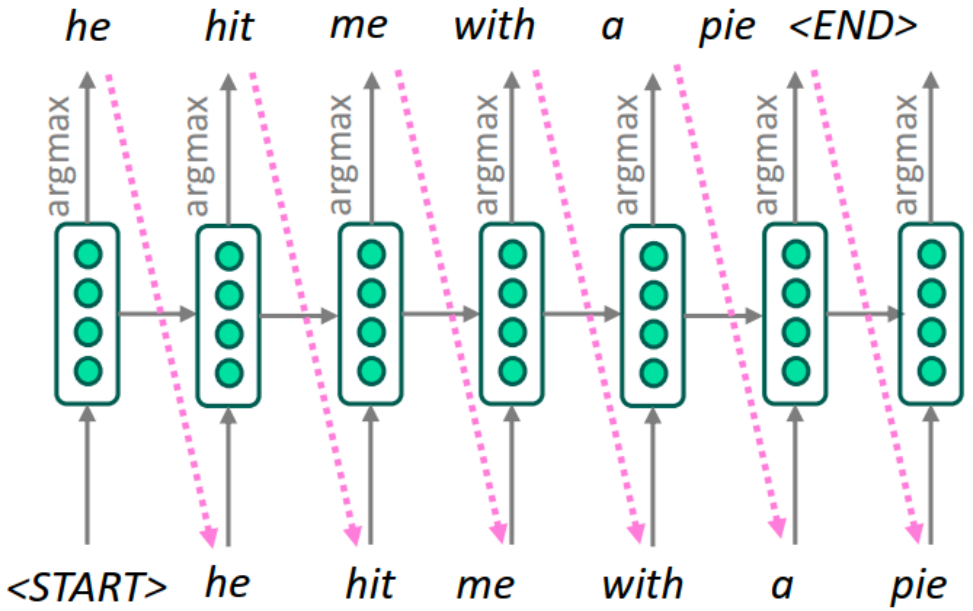
[Sutskever et al. 2014; Luong et al. 2015]

The hidden states from RNN layer *i* are the inputs to RNN layer *i*+1



Greedy decoding

- We saw how to generate (or "decode") the target sentence by taking argmax on each step of the decoder



- This is greedy decoding (take most probable word on each step)
- Problems with this method?

Problems with greedy decoding

- Greedy decoding has no way to undo decisions!
  - Input: il a m'entarté (he hit me with a pie)
  - → he ____
  - → he hit ____
  - → he hit a ____ (whoops! no going back now…)

- How to fix this?

Exhaustive search decoding

- Ideally, we want to find a (length T) translation y that maximizes

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

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

- We could try computing all possible sequences y
  - This means that on each step t of the decoder, we're tracking V^T possible partial translations, where V is vocab size
  - This $O(V^T)$ complexity is far too expensive!

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, in NMT)
- 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!

Beam search decoding: example

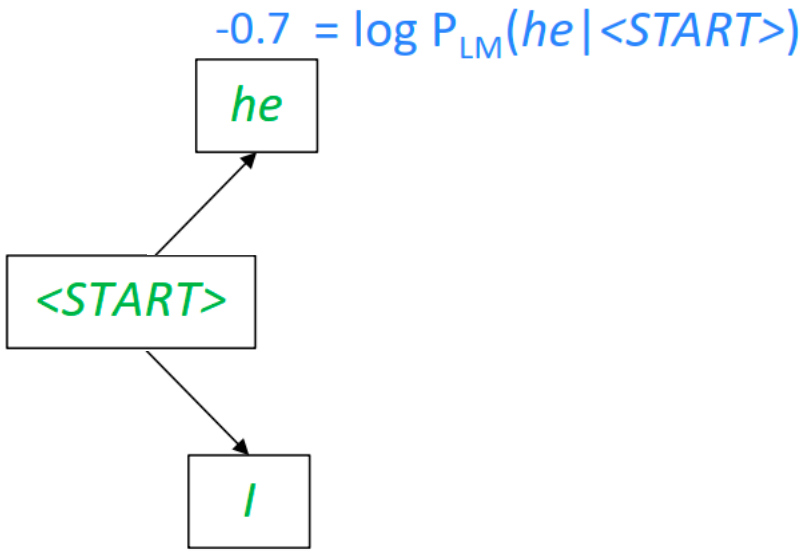
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$

<START>

Calculate prob dist of next word

Beam search decoding: example

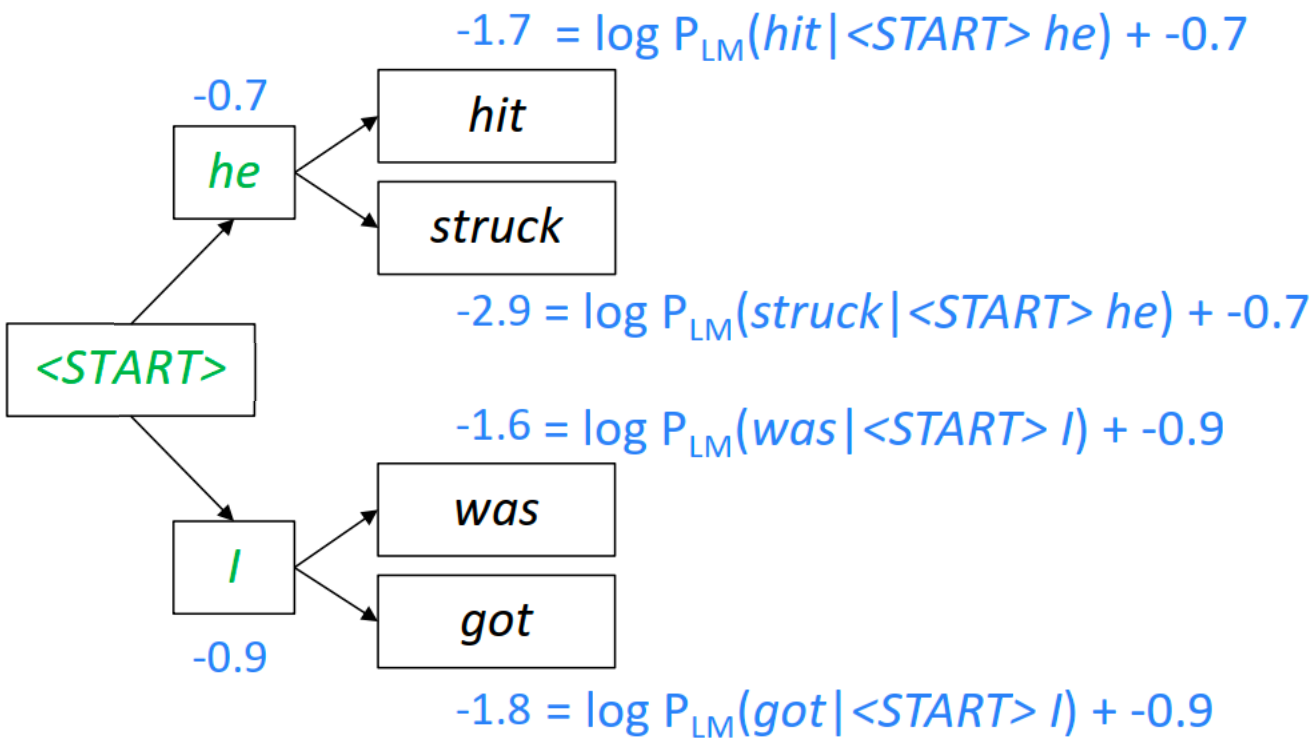
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Take top *k* words and compute scores

Beam search decoding: example

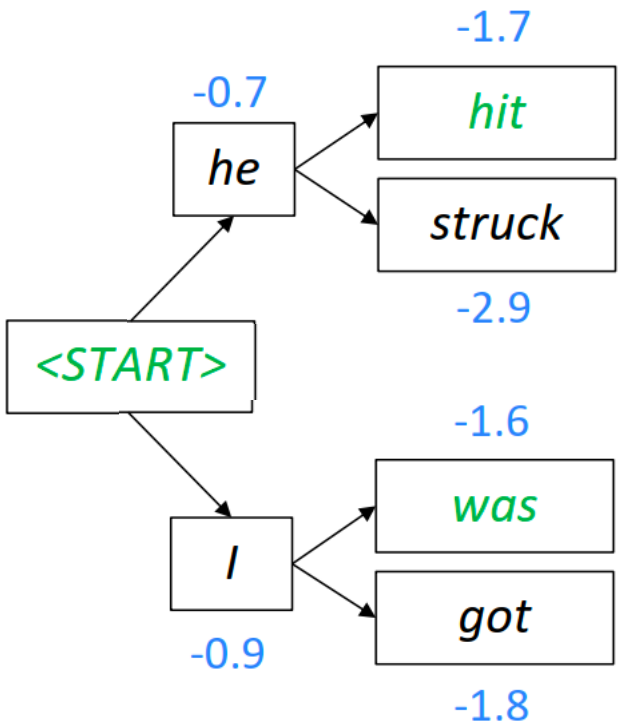
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam search decoding: example

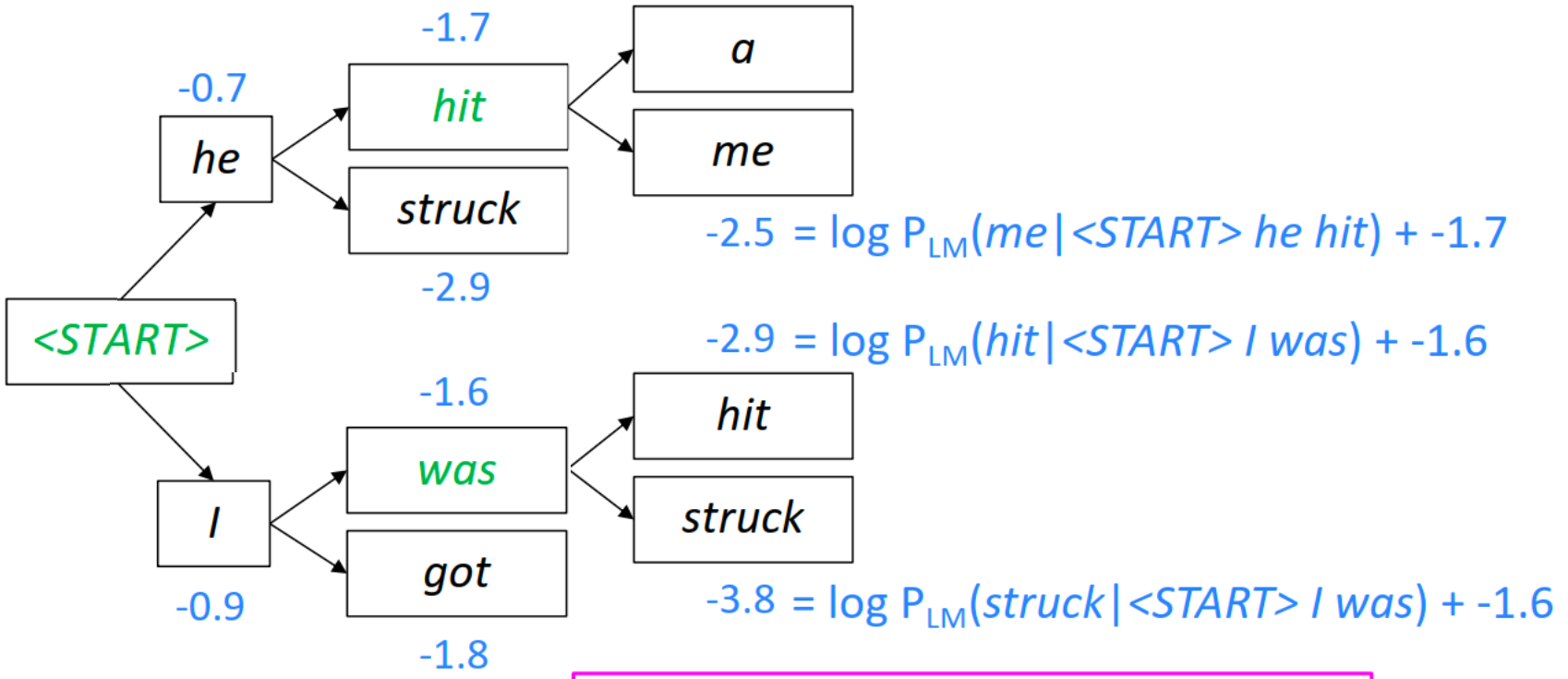
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam search decoding: example

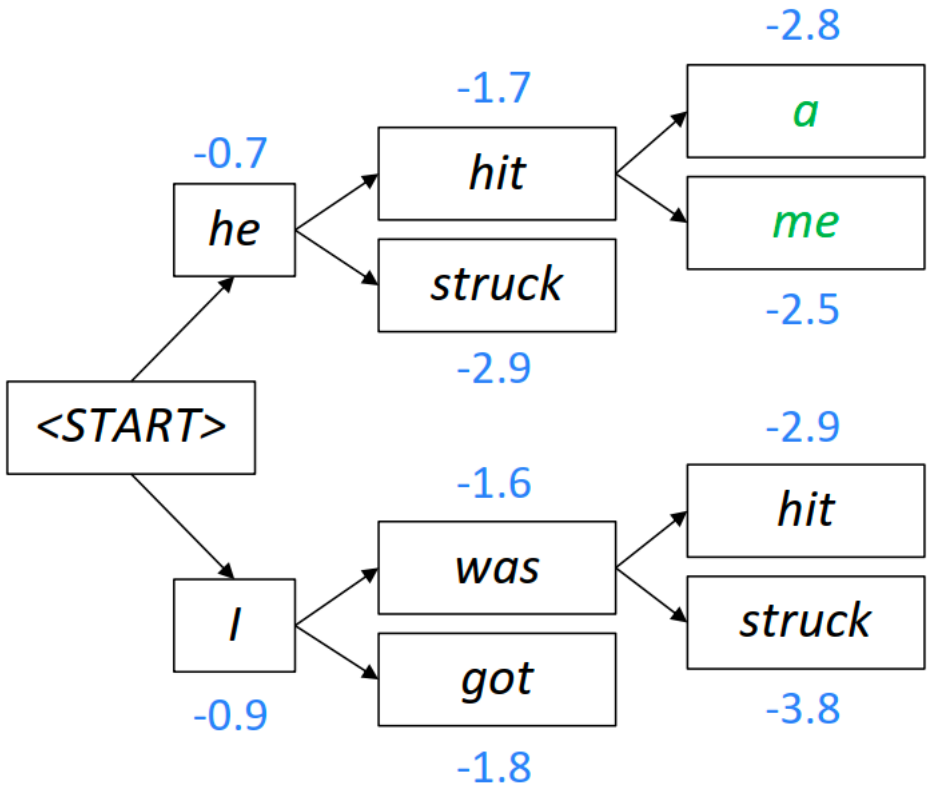
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam search decoding: example

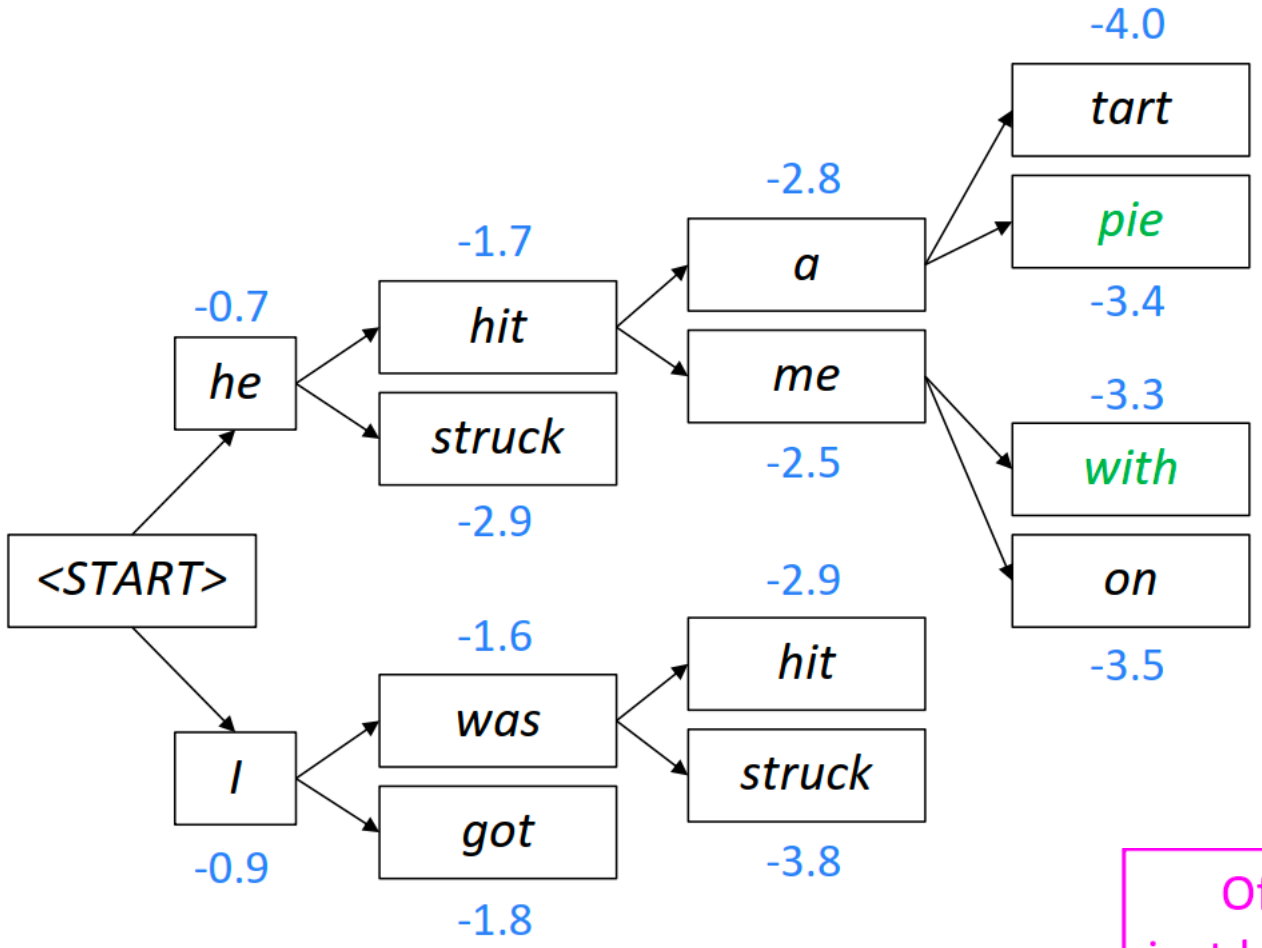
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam search decoding: example

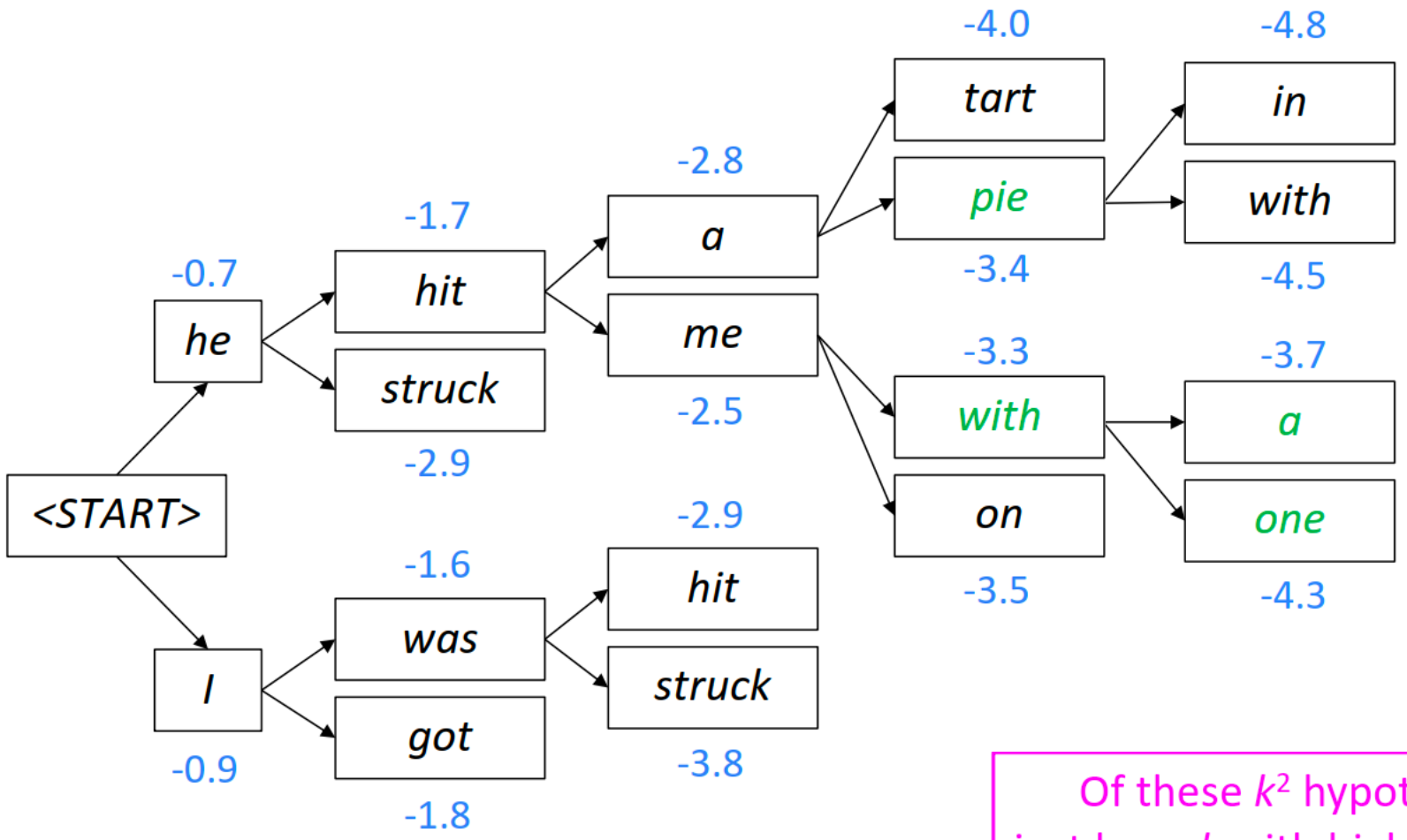
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam search decoding: example

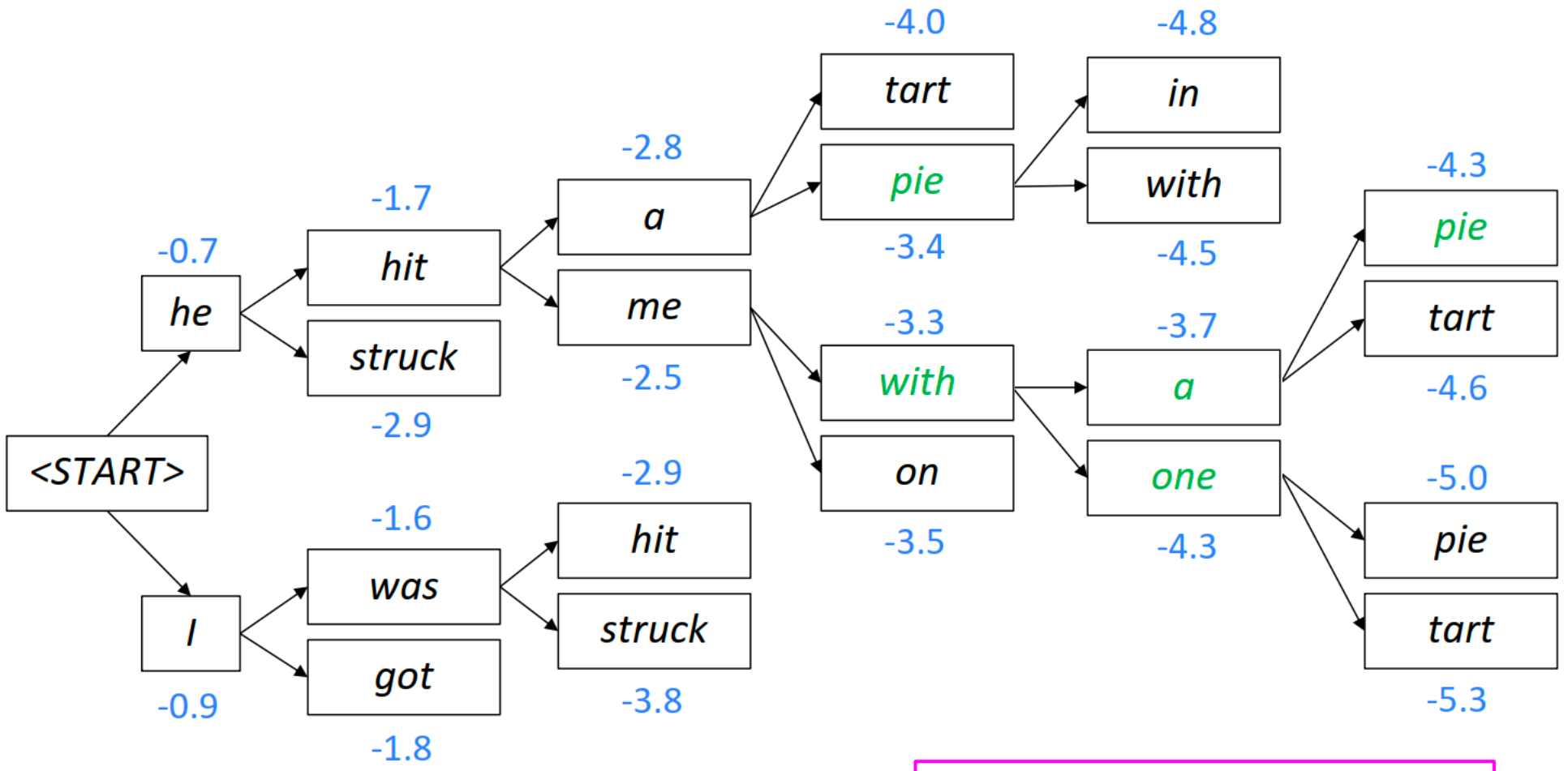
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



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

Beam search decoding: example

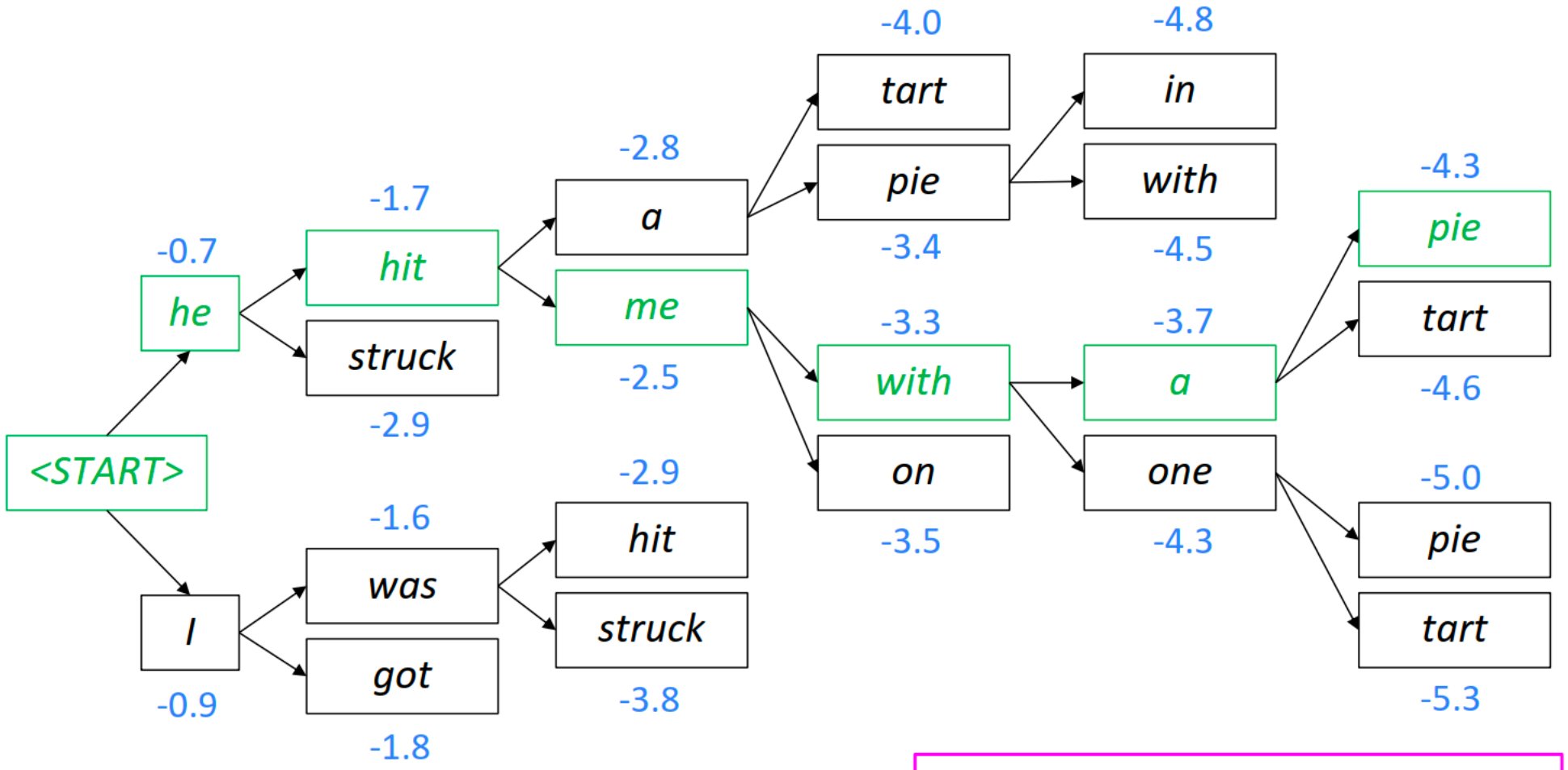
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



This is the top-scoring hypothesis!

Beam search decoding: example

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



Backtrack to obtain the full hypothesis

Beam search decoding: stopping criterion

- In greedy decoding, usually we decode until the model produces an <END> token
  - **For example**: <START> he hit me with a pie <END>

- In beam search decoding, different hypotheses may produce <END> tokens on different timesteps
  - When a hypothesis produces <END>, that hypothesis is complete.
  - Place it aside and continue exploring other hypotheses via beam search.

- Usually we continue beam search until:
  - We reach timestep T (where T is some pre-defined cutoff), or
  - We have at least n completed hypotheses (where n is pre-defined cutoff)

Beam search decoding: finishing up

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

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

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

- **Problem with this**: longer hypotheses have lower scores
- **Fix**: Normalize by length. Use this to select top one instead:

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

Advantages of NMT

- 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

Disadvantages of NMT?

- 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!

How do we evaluate Machine Translation?

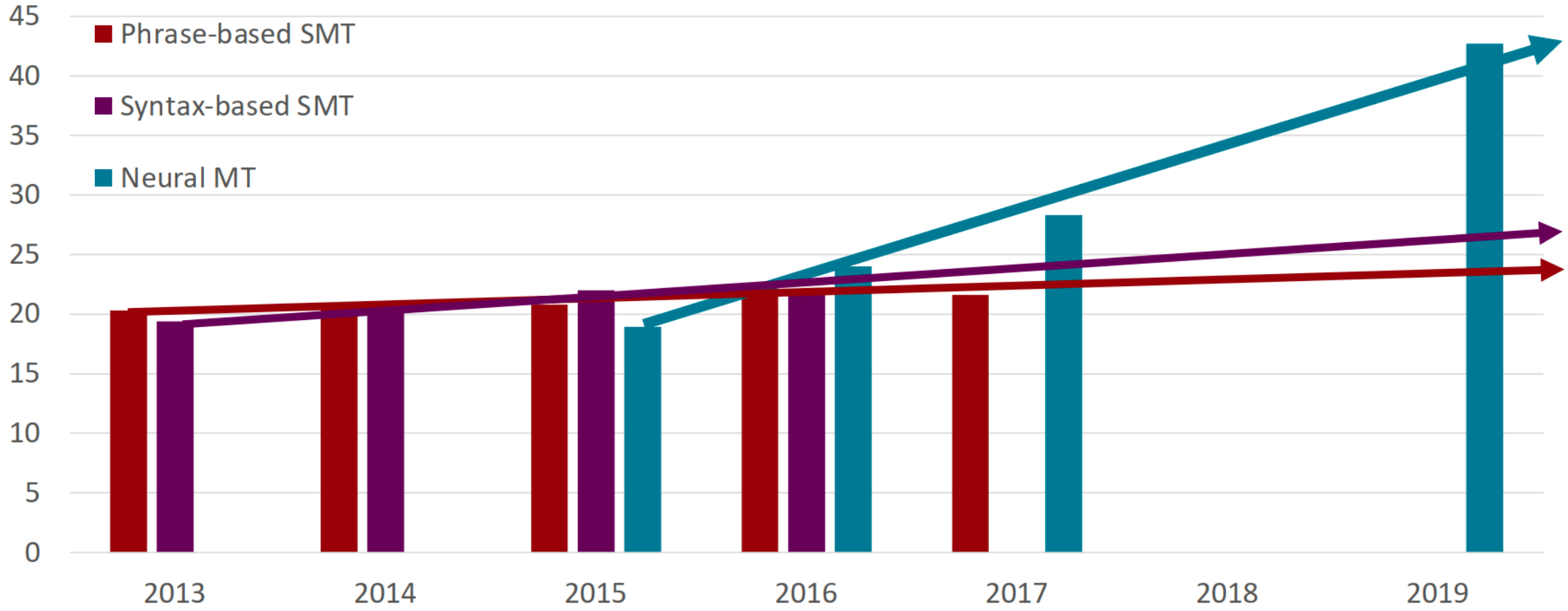
- **BLEU (Bilingual Evaluation Understudy)**

- 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 🥵

Source: ”BLEU: a Method for Automatic Evaluation of Machine Translation“, Papineni et al, 2002. http://aclweb.org/anthology/P02-1040

MT progress over time



Sources: http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf & http://matrix.statmt.org/

NMT: perhaps the biggest success story of NLP Deep Learning?

- Neural Machine Translation went from a fringe research attempt in 2014 to the leading standard method in 2016

- 2014: First seq2seq paper published
- 2016: Google Translate switches from SMT to NMT – and by 2018 everyone has

- This is amazing!
  - SMT systems, built by hundreds of engineers over many years, outperformed by
  - NMT systems trained by a small group of engineers in a few months

So, is Machine Translation solved?

- Nope!
- Many difficulties remain:
  - Out-of-vocabulary words
  - Domain mismatch between train and test data
  - Maintaining context over longer text
  - Low-resource language pairs
  - Failures to accurately capture sentence meaning
  - Pronoun (or zero pronoun) resolution errors
  - Morphological agreement errors

Further reading: “Has AI surpassed humans at translation? Not even close!”
https://www.skynettoday.com/editorials/state_of_nmt

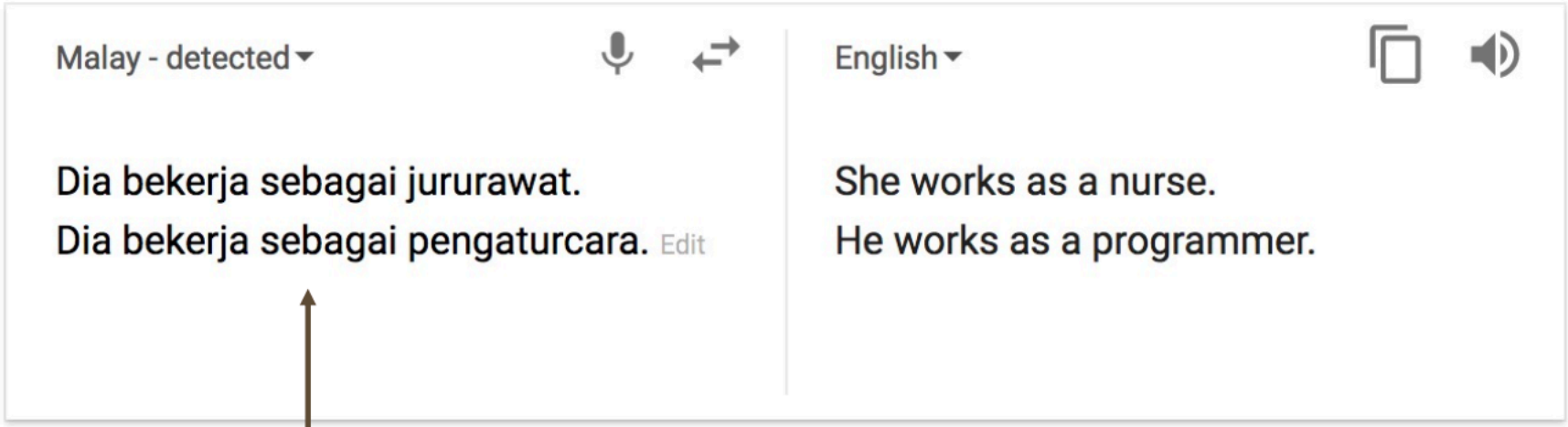
So is Machine Translation solved?

- Nope!
- Using common sense is still hard

?

So is Machine Translation solved?

- Nope!
- NMT picks up biases in training data

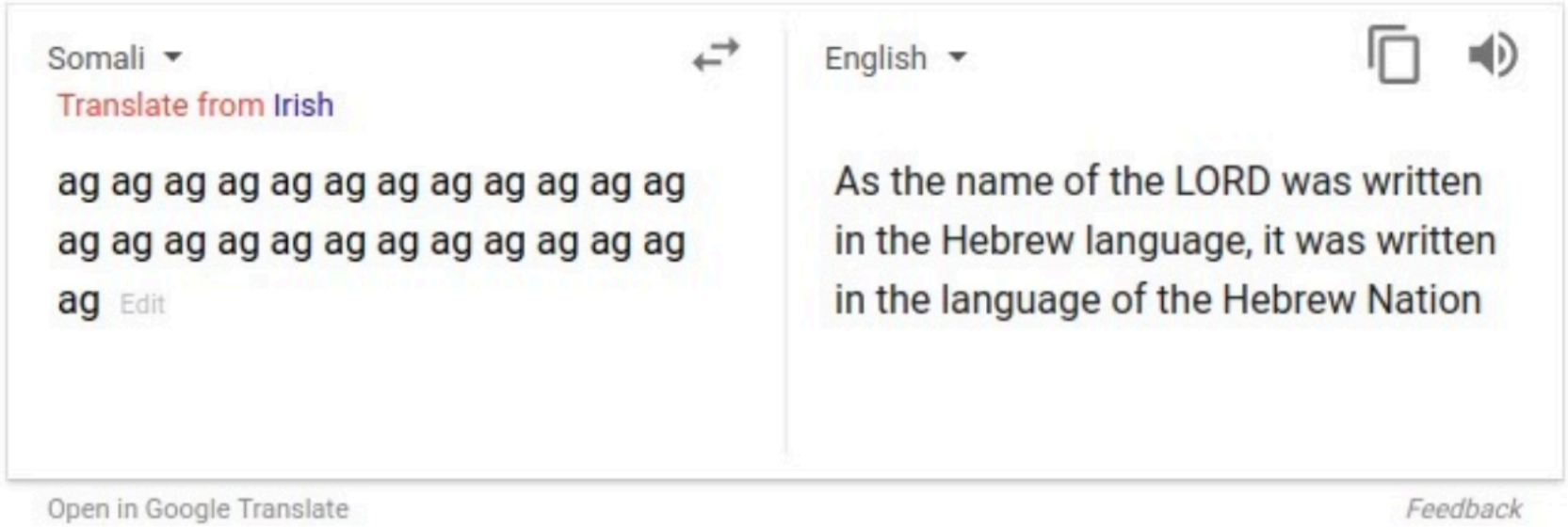


Didn't specify gender

Source: https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c

So is Machine Translation solved?

- Nope!
- Uninterpretable systems do strange things
- (But I think this problem has been fixed in Google Translate by 2021.)



Picture source: https://www.vice.com/en_uk/article/j5npeg/why-is-google-translate-spitting-out-sinister-religious-prophecies
Explanation: https://www.skynettoday.com/briefs/google-nmt-prophecies

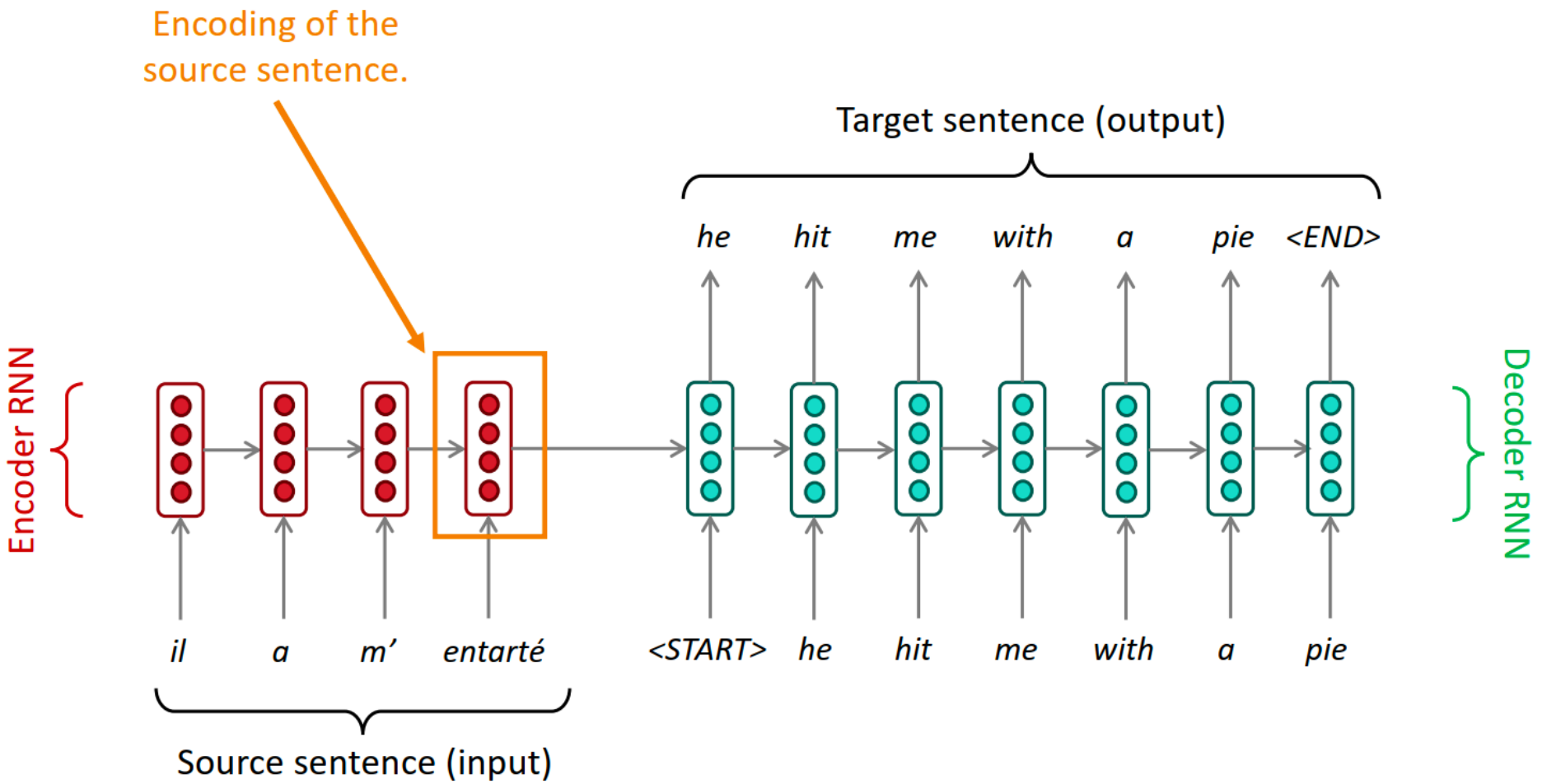
NMT research continues

- NMT is a **flagship task** for NLP Deep Learning
- NMT research has pioneered many of the recent innovations of NLP Deep Learning
- NMT research continues to thrive
  - Researchers have found many, many improvements to the "vanilla" seq2seq NMT system we've just presented
  - But we'll present next one improvement so integral that it is the new vanilla…

Attention

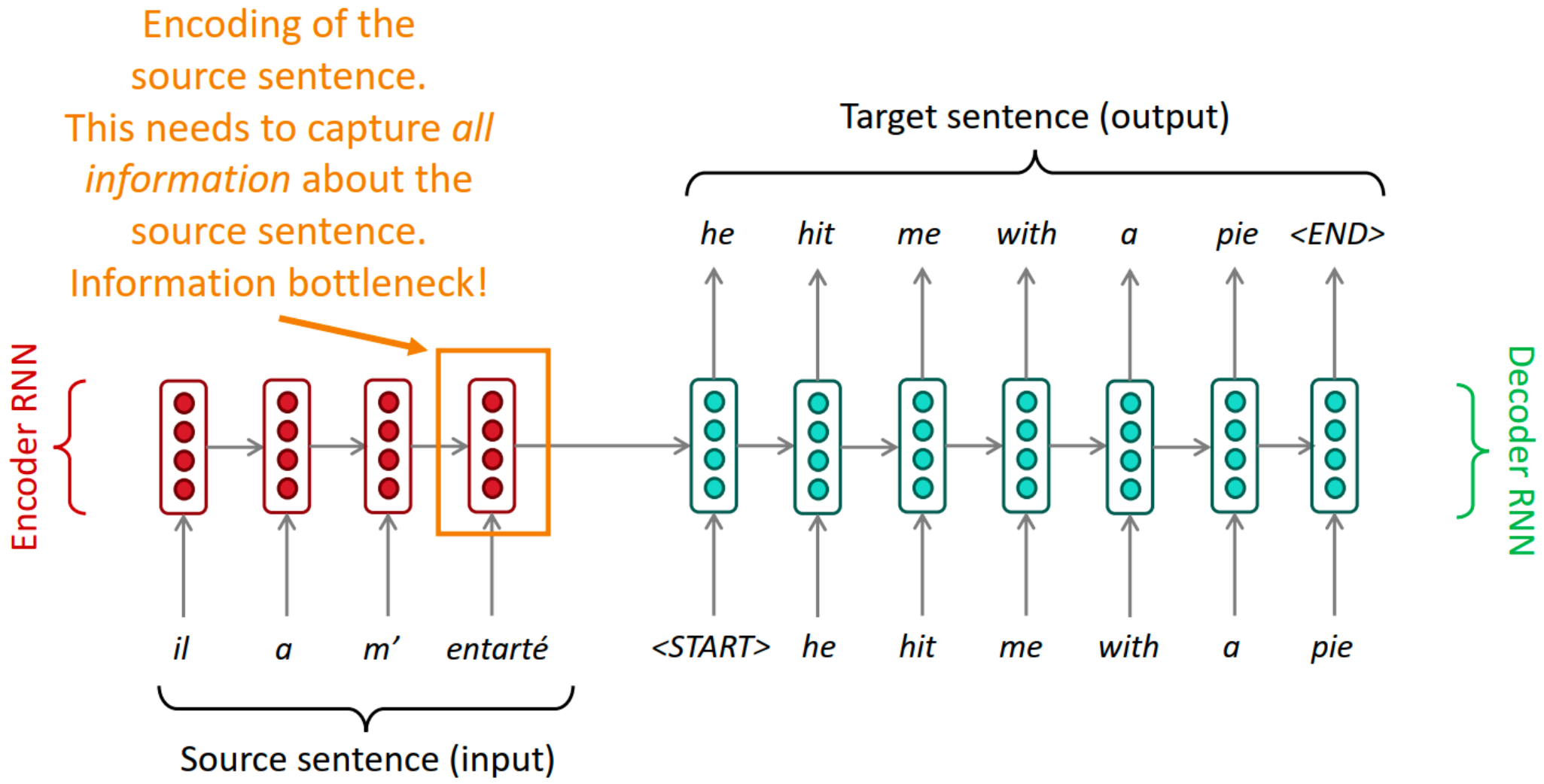
注意力模型Attention

Sequence-to-sequence: the bottleneck problem



Problems with this architecture?

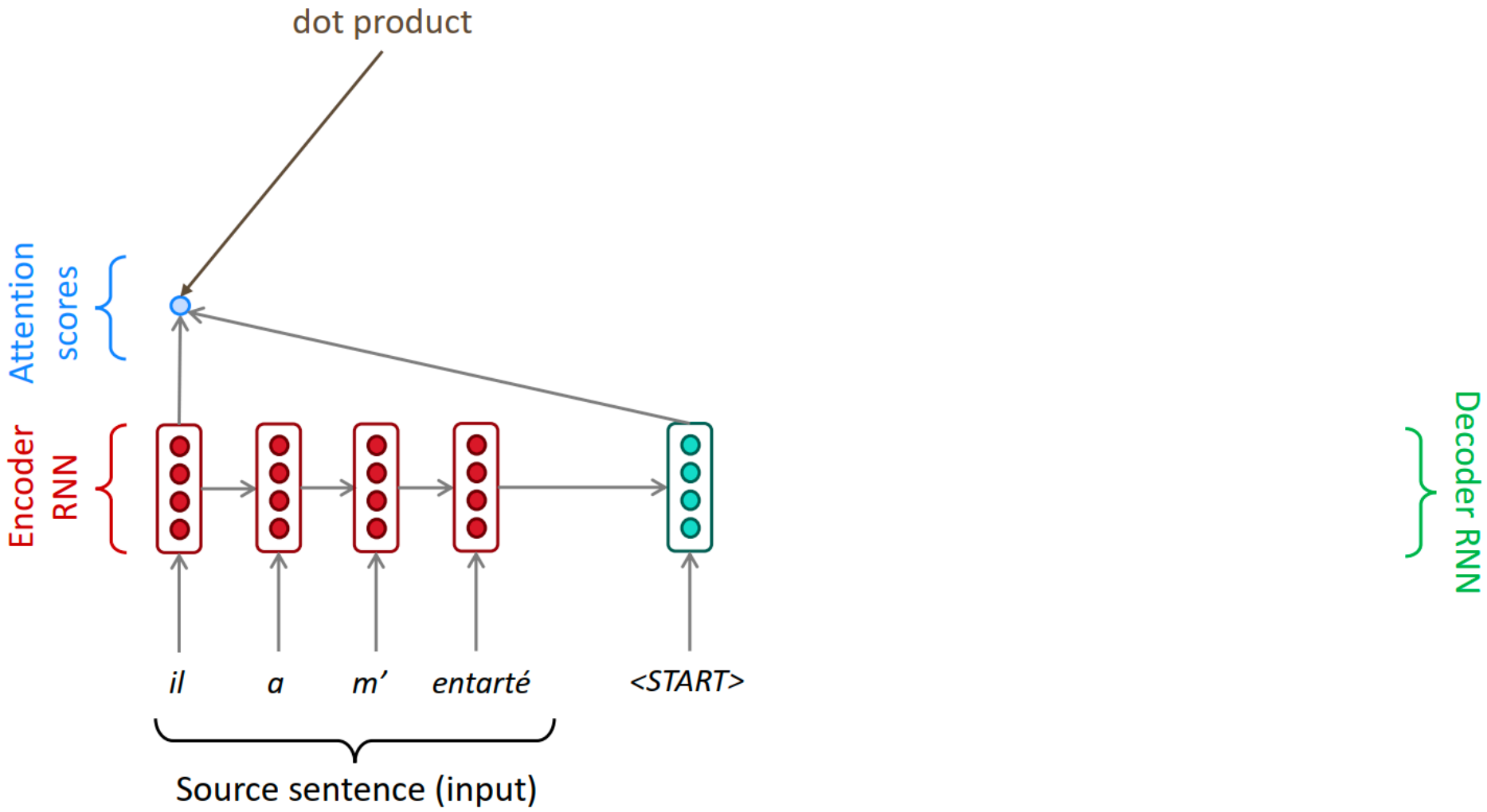
Sequence-to-sequence: the bottleneck problem



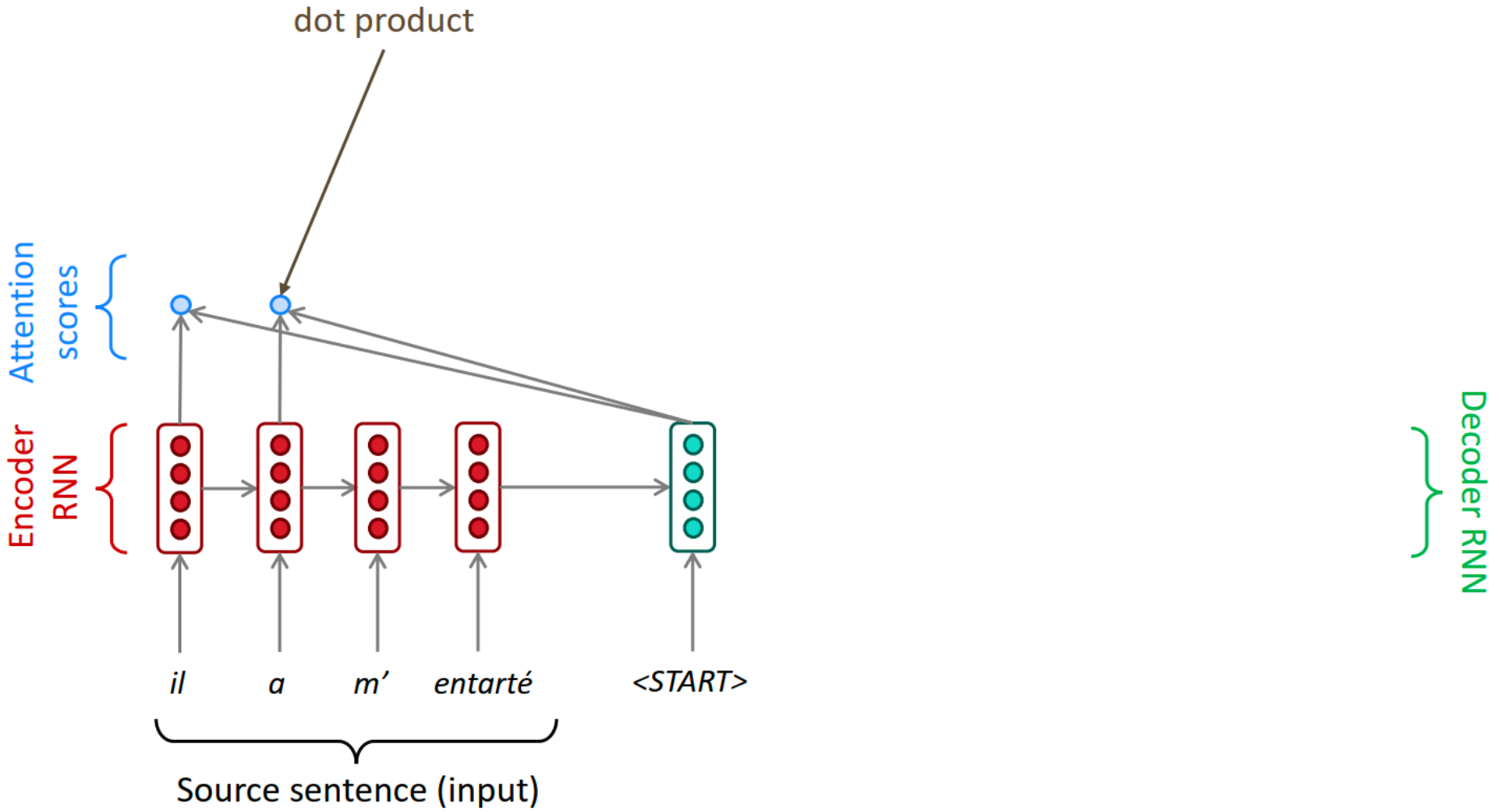
Attention

- **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

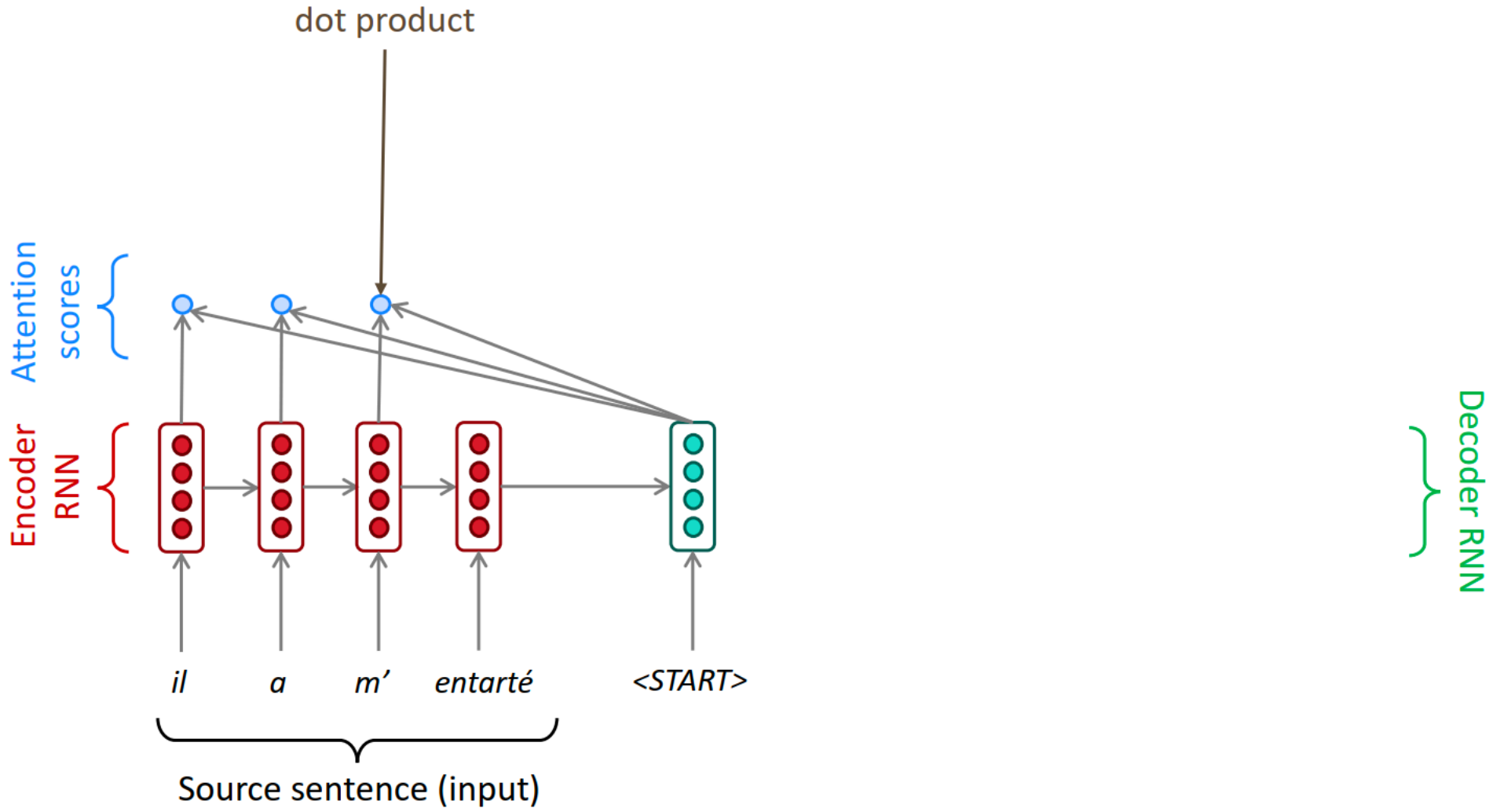
Sequence-to-sequence with attention



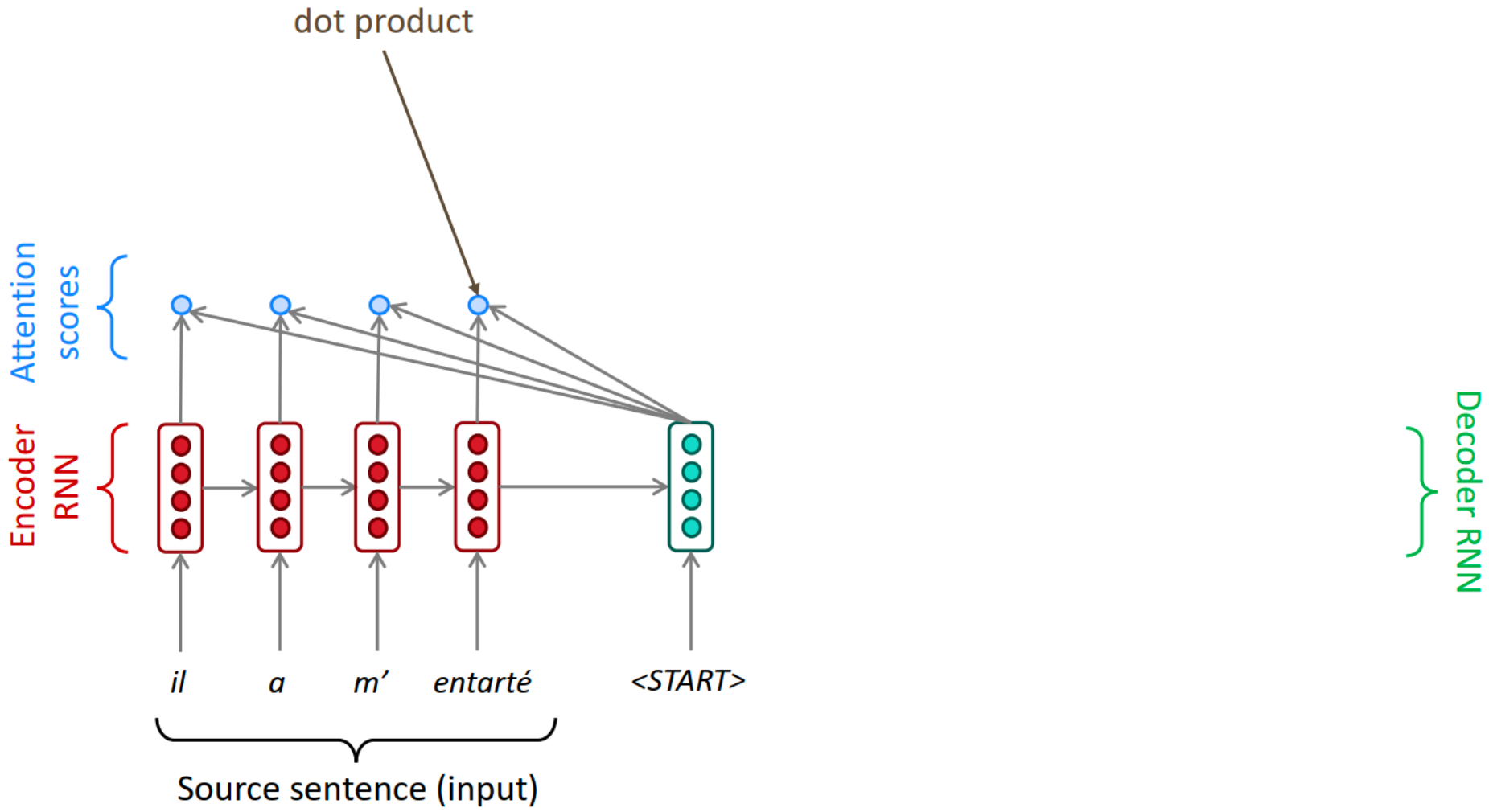
Sequence-to-sequence with attention



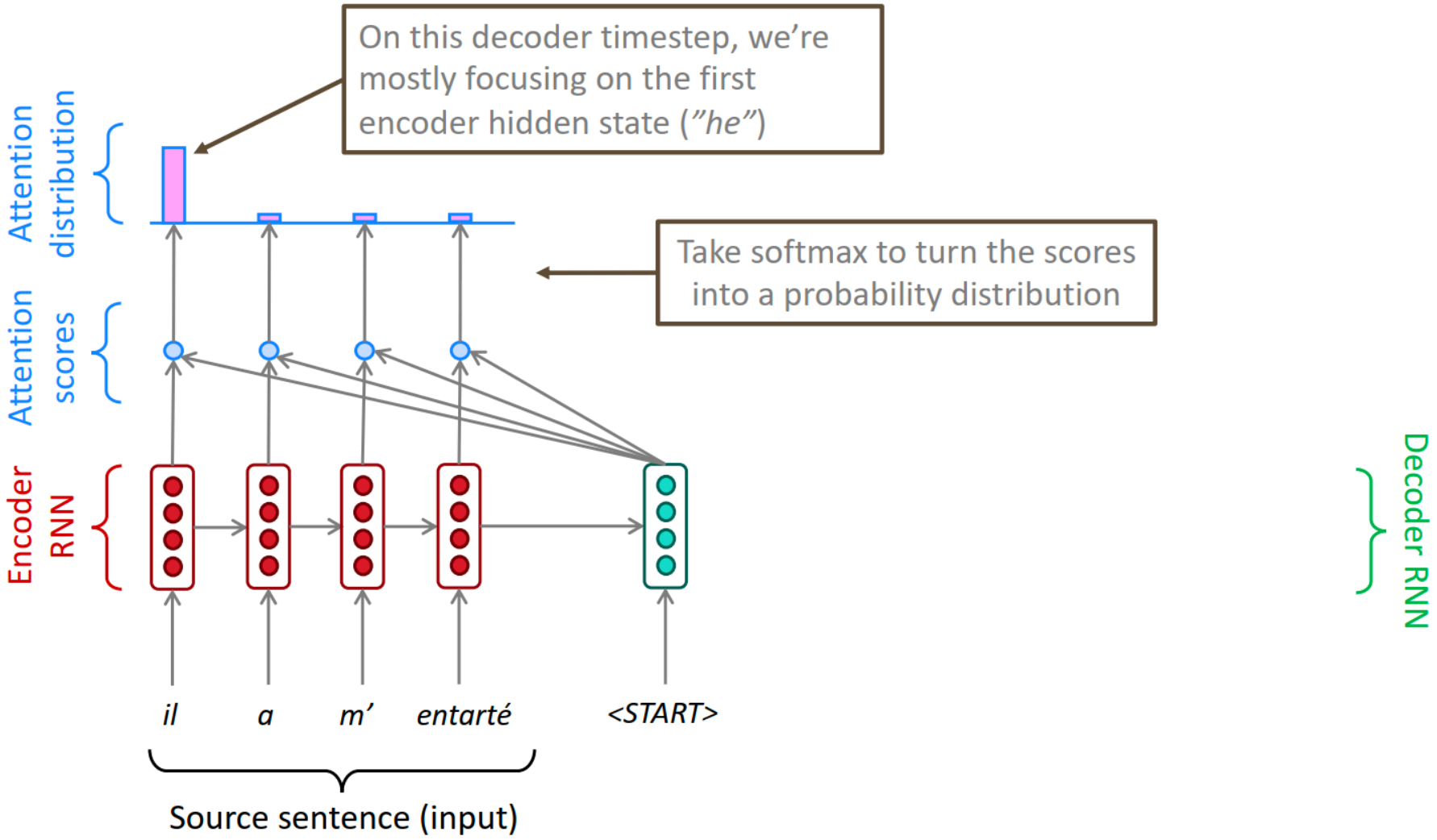
Sequence-to-sequence with attention



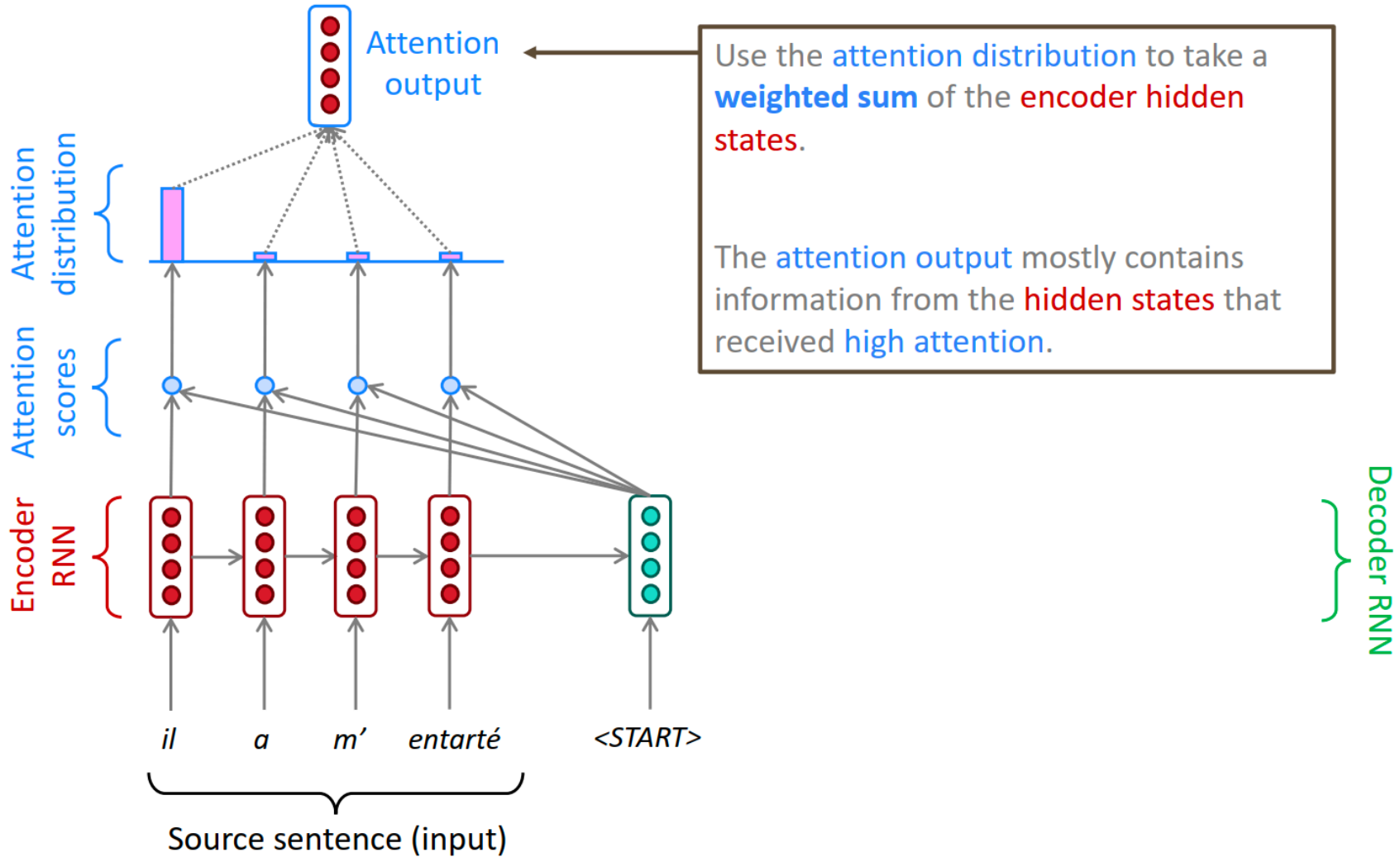
Sequence-to-sequence with attention



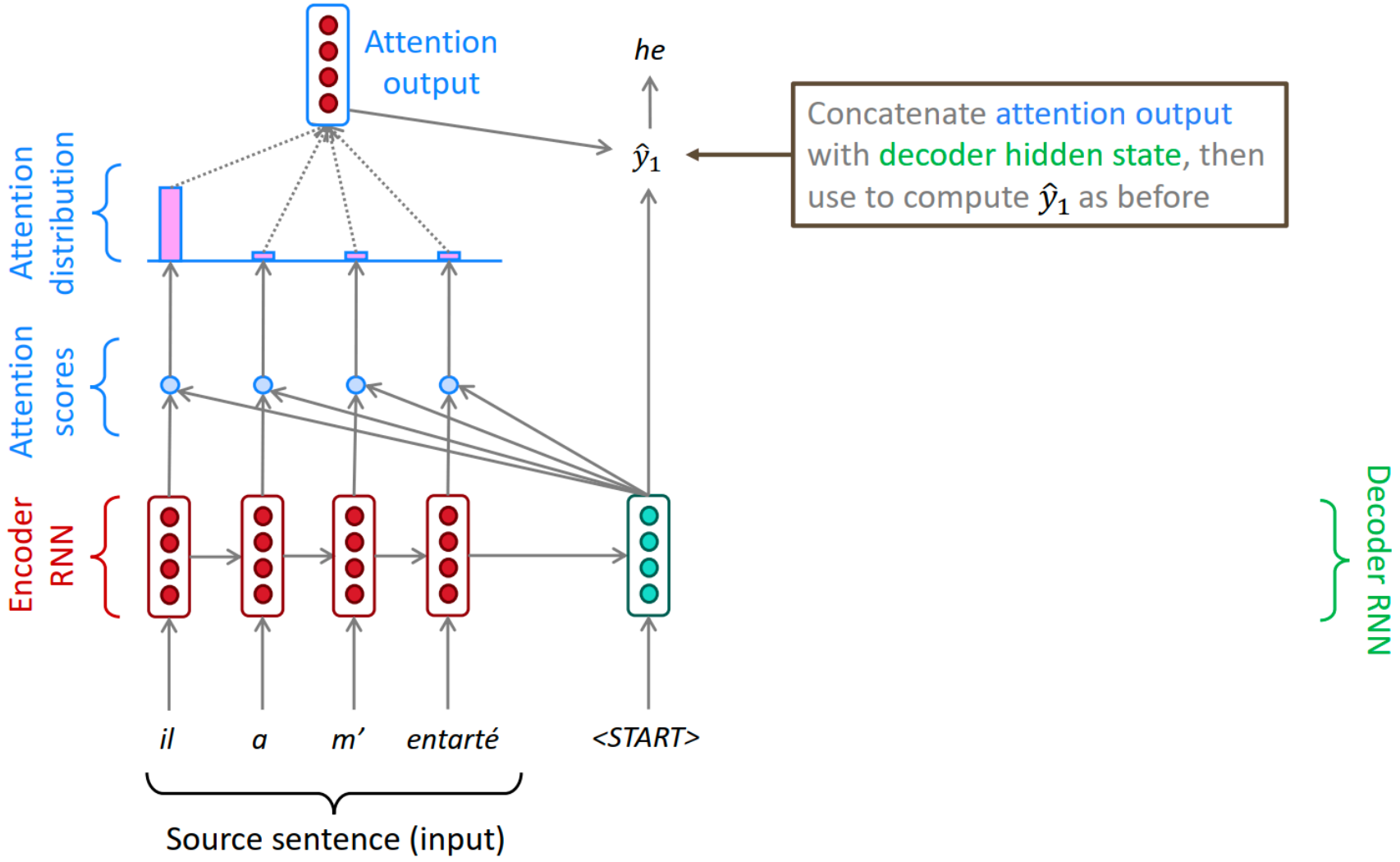
Sequence-to-sequence with attention



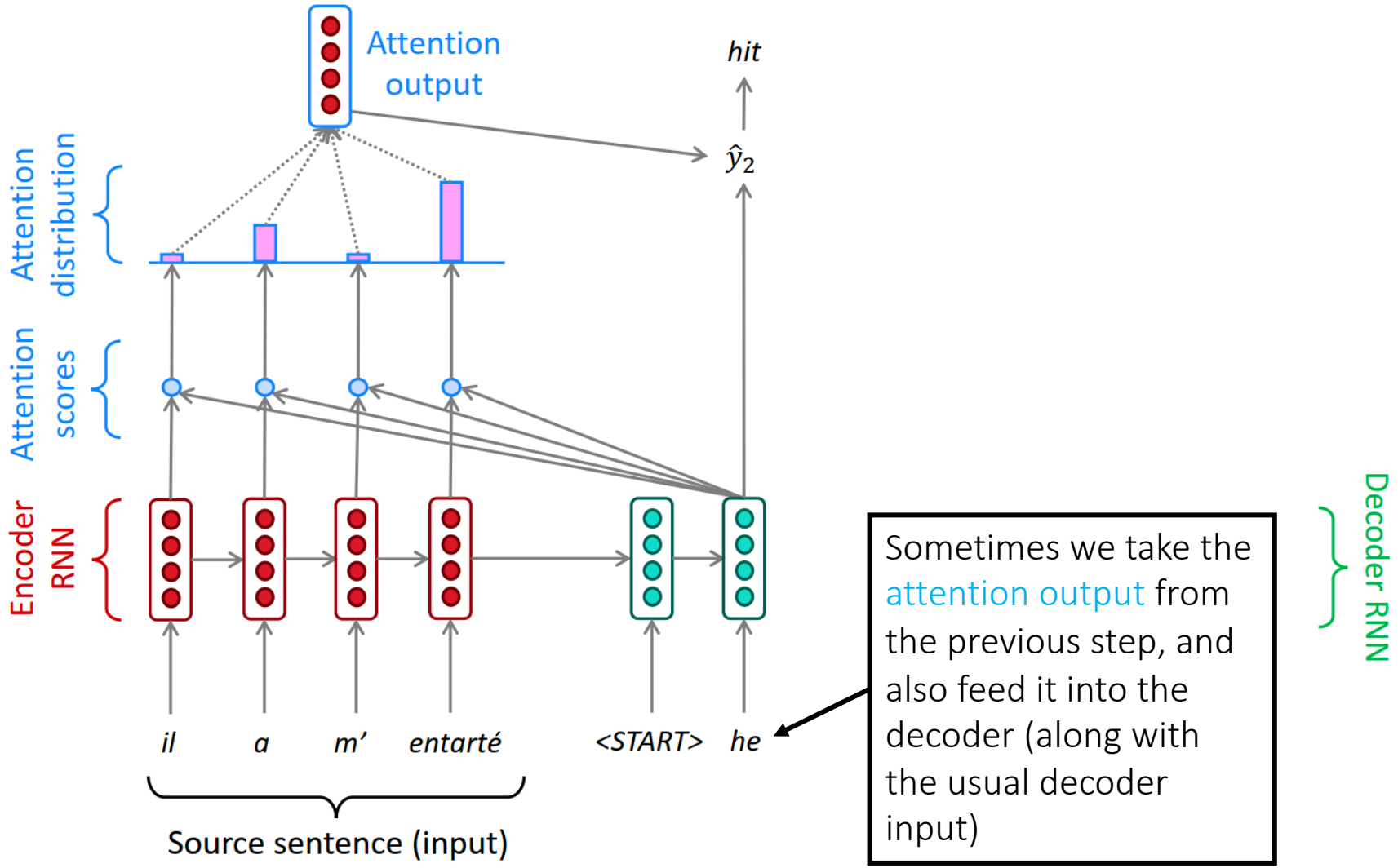
Sequence-to-sequence with attention



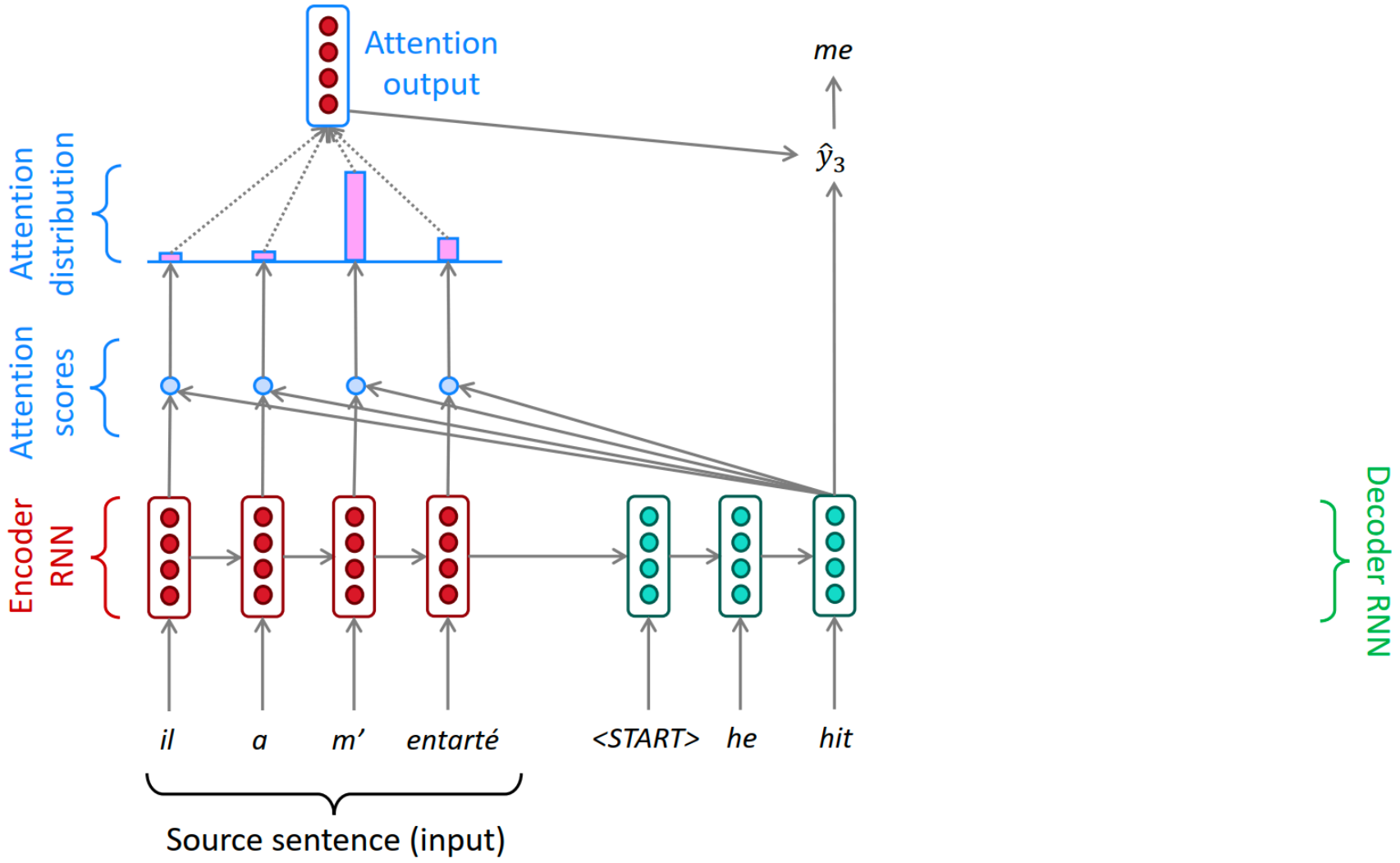
Sequence-to-sequence with attention



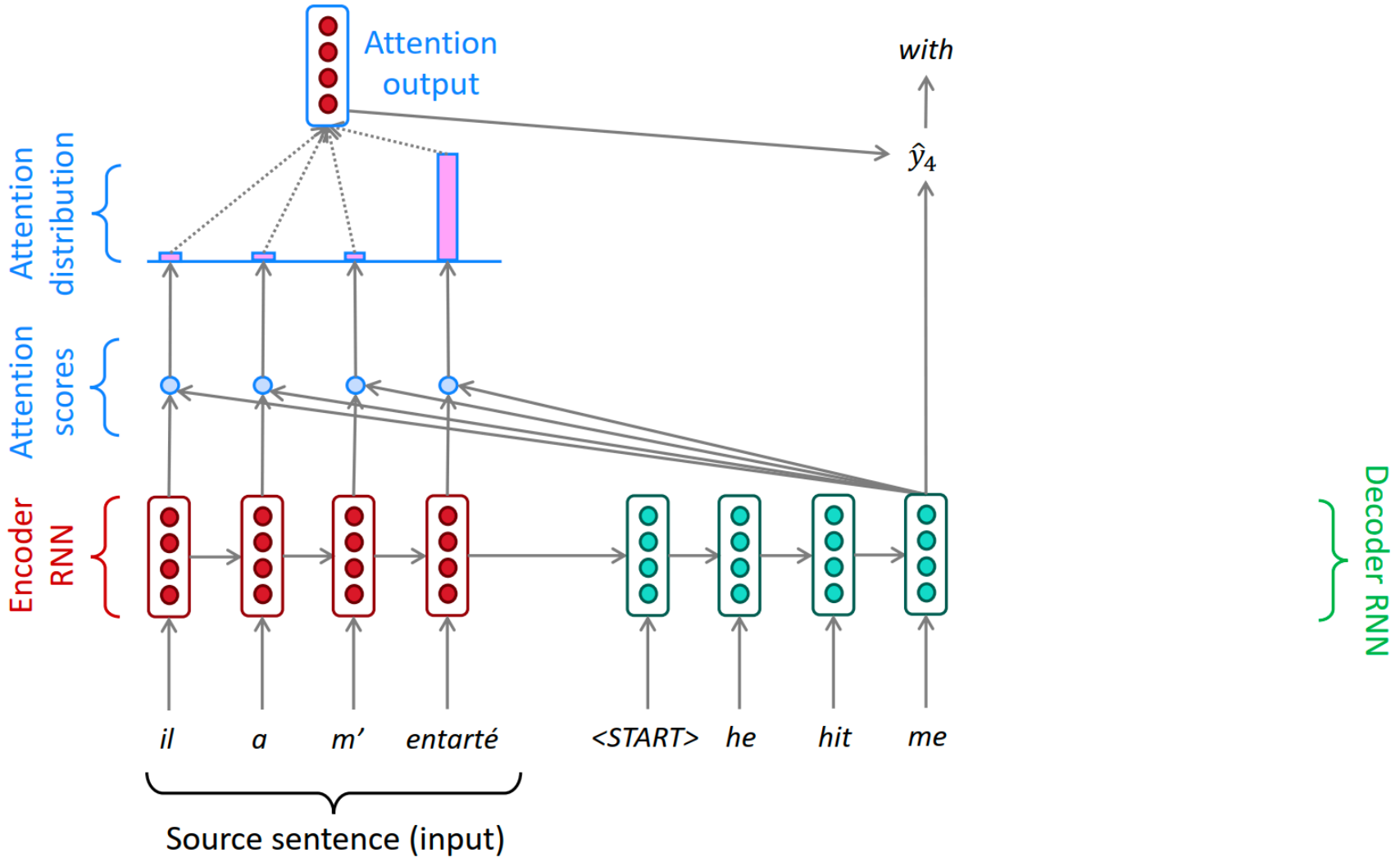
Sequence-to-sequence with attention



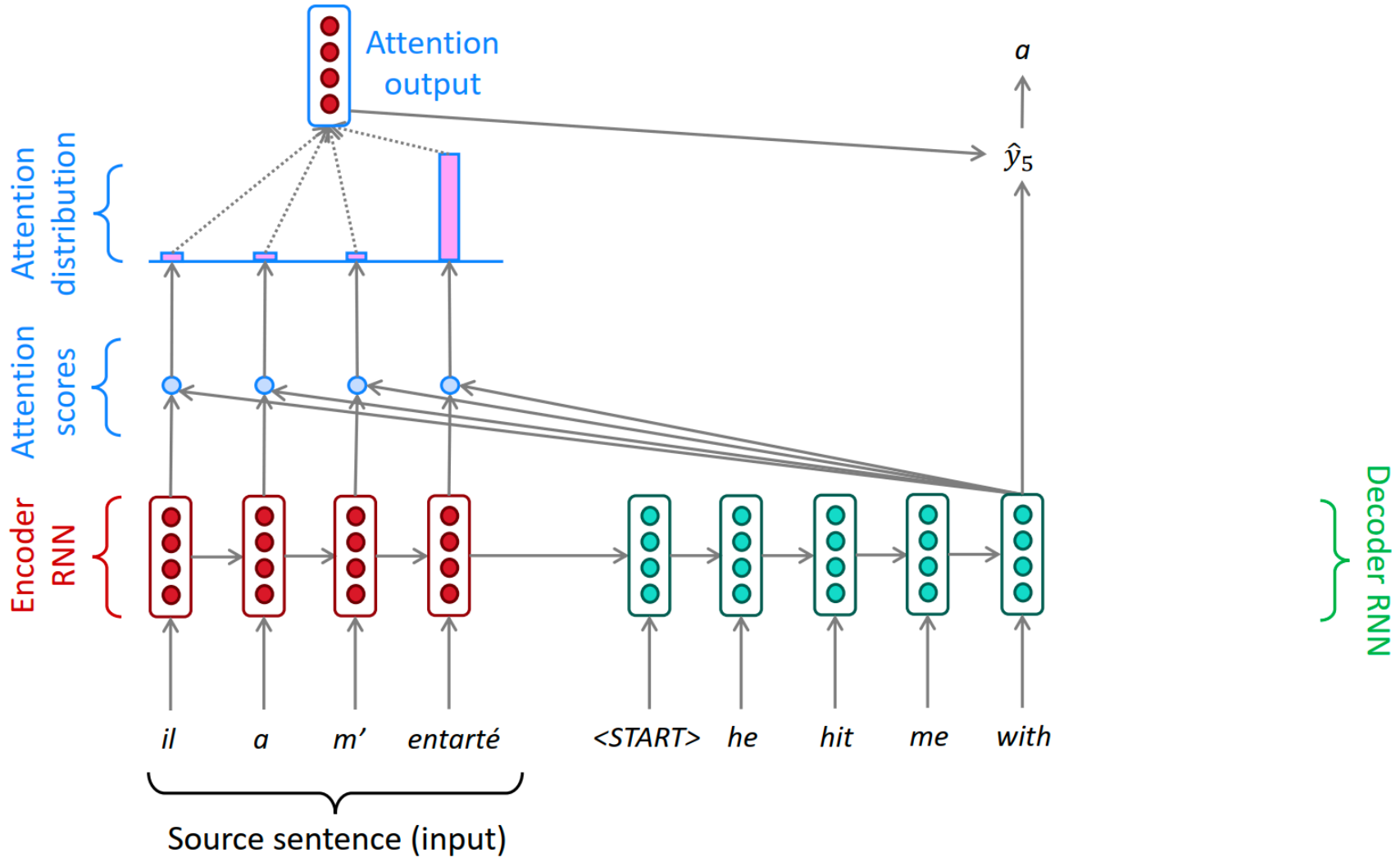
Sequence-to-sequence with attention



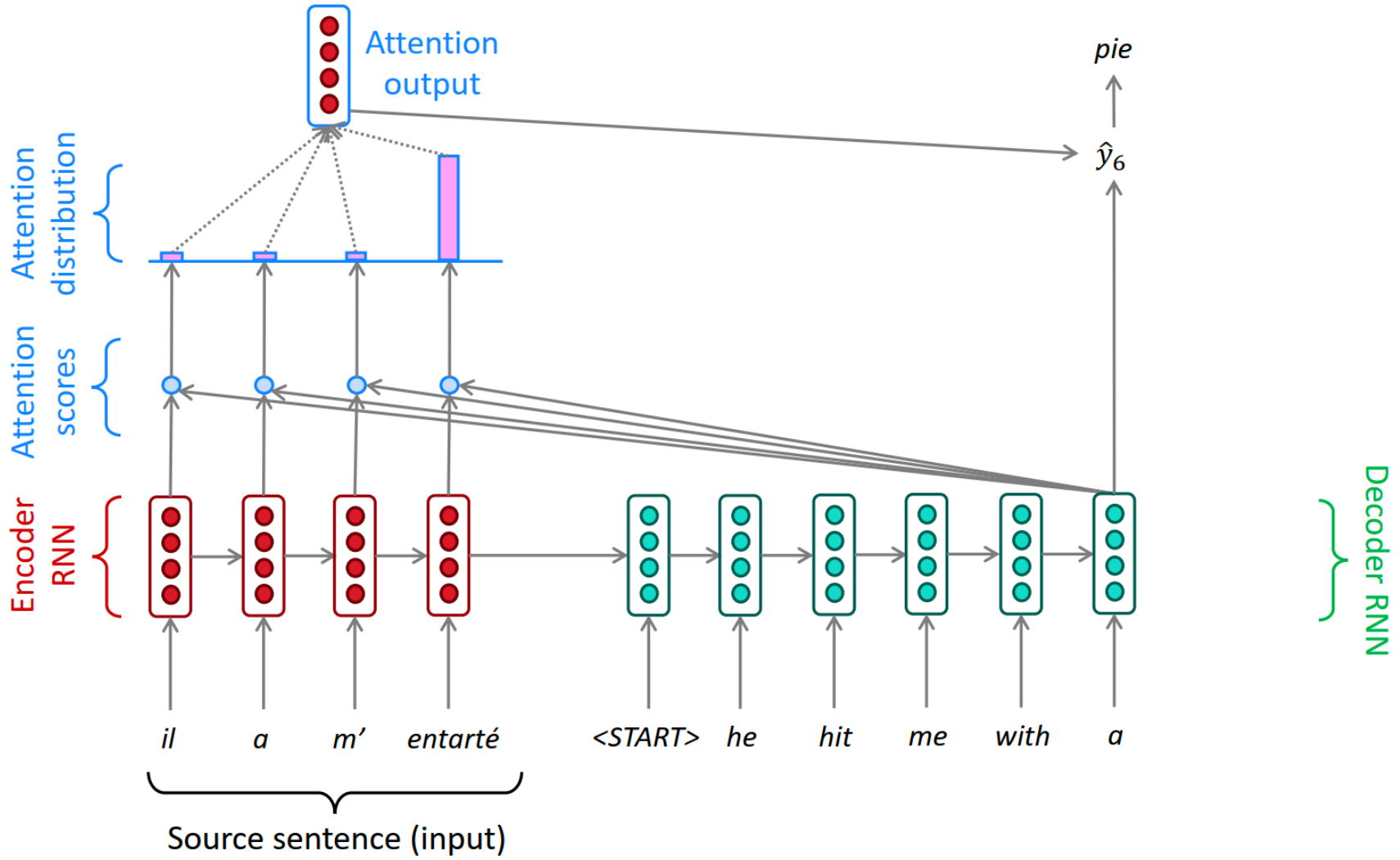
Sequence-to-sequence with attention



Sequence-to-sequence with attention



Sequence-to-sequence 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)

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

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

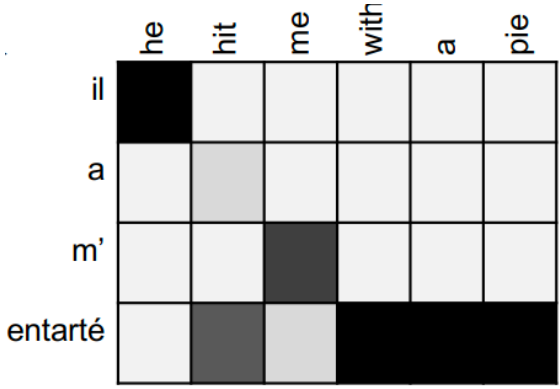
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- 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

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

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 al
- 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



There are several 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 $\boldsymbol{e} \in \mathbb{R}^N$ ← There are multiple ways to do this
  - 2. Taking softmax to get attention distribution α:

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

  - 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)

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. Perhaps better called "bilinear attention"
- Reduced-rank multiplicative attention: $e_i = s^T (\boldsymbol{U}^T \boldsymbol{V}) h_i = (\boldsymbol{U} s)^T (\boldsymbol{V} h_i)$
  - For low rank matrices $\boldsymbol{U} \in \mathbb{R}^{k \times d_2}, \boldsymbol{V} \in \mathbb{R}^{k \times d_1}, k \ll d_1, d_2$
- 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
  - "Additive" is a weird/bad name. It's really using a feed-forward neural net layer.

Attention is a general Deep Learning technique

- You can also 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).

Thank you