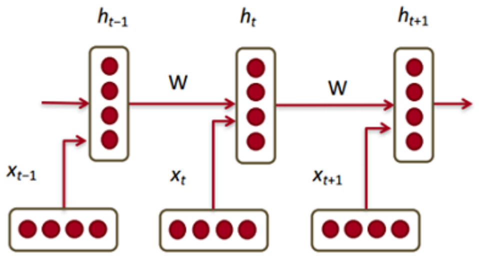
Natural Language Processing



Google Translate

$$x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture}$$

$$x_{king} - x_{man} \approx x_{queen} - x_{woman}$$

$$\log p(o|c) = \log \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}$$

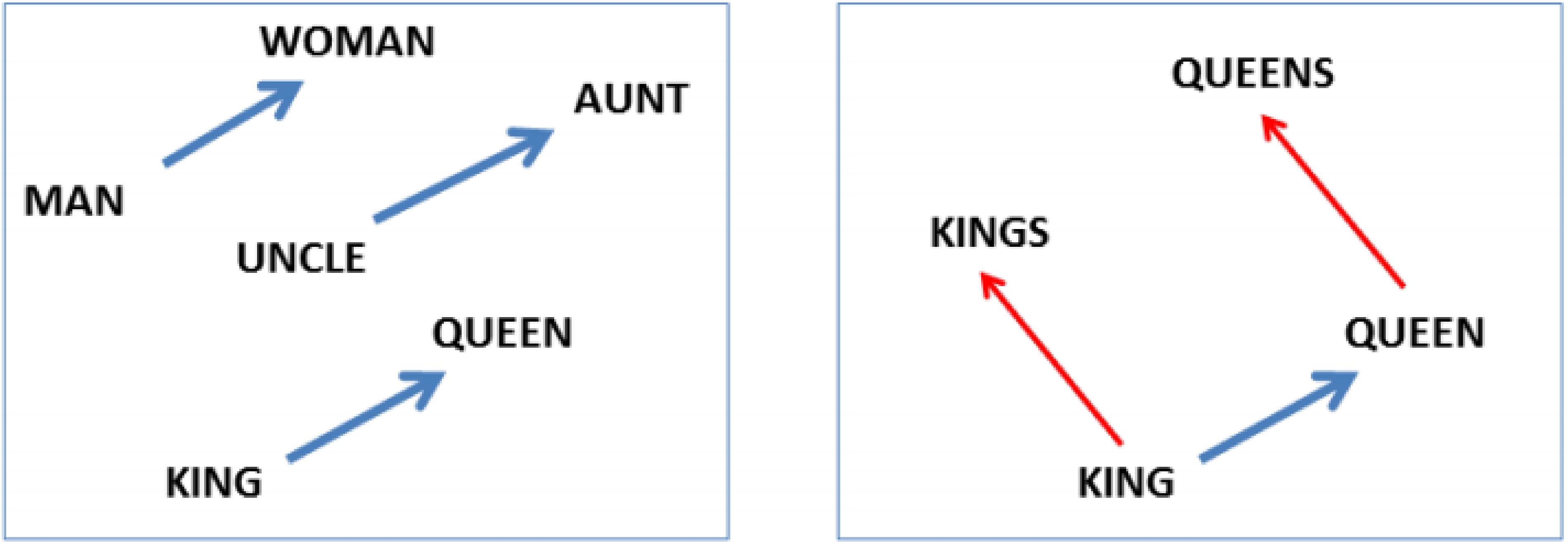
Natural Language Processing

- Token Representation – Word2Vec
- Sentence Representation – CBOW, RN, CNN, Self Attention, RNN
- Language Model – N-Gram LM, NNLM
- Neural Machine Translation

How to Represent a Token

- Intuitive embedding – one hot encoding
  - Apple, Strawberry, Dog 세 단어가 있을 때,
  - Apple → [1, 0, 0]
  - Strawberry → [0, 1, 0]
  - Dog → [0, 0, 1]
- 장점
  - Easy!
- 단점
  - 단어들 간의 의미관계를 파악할 수 없음(apple과 strawberry, apple과 dog)
  - 단어가 많아지면?

We Want…

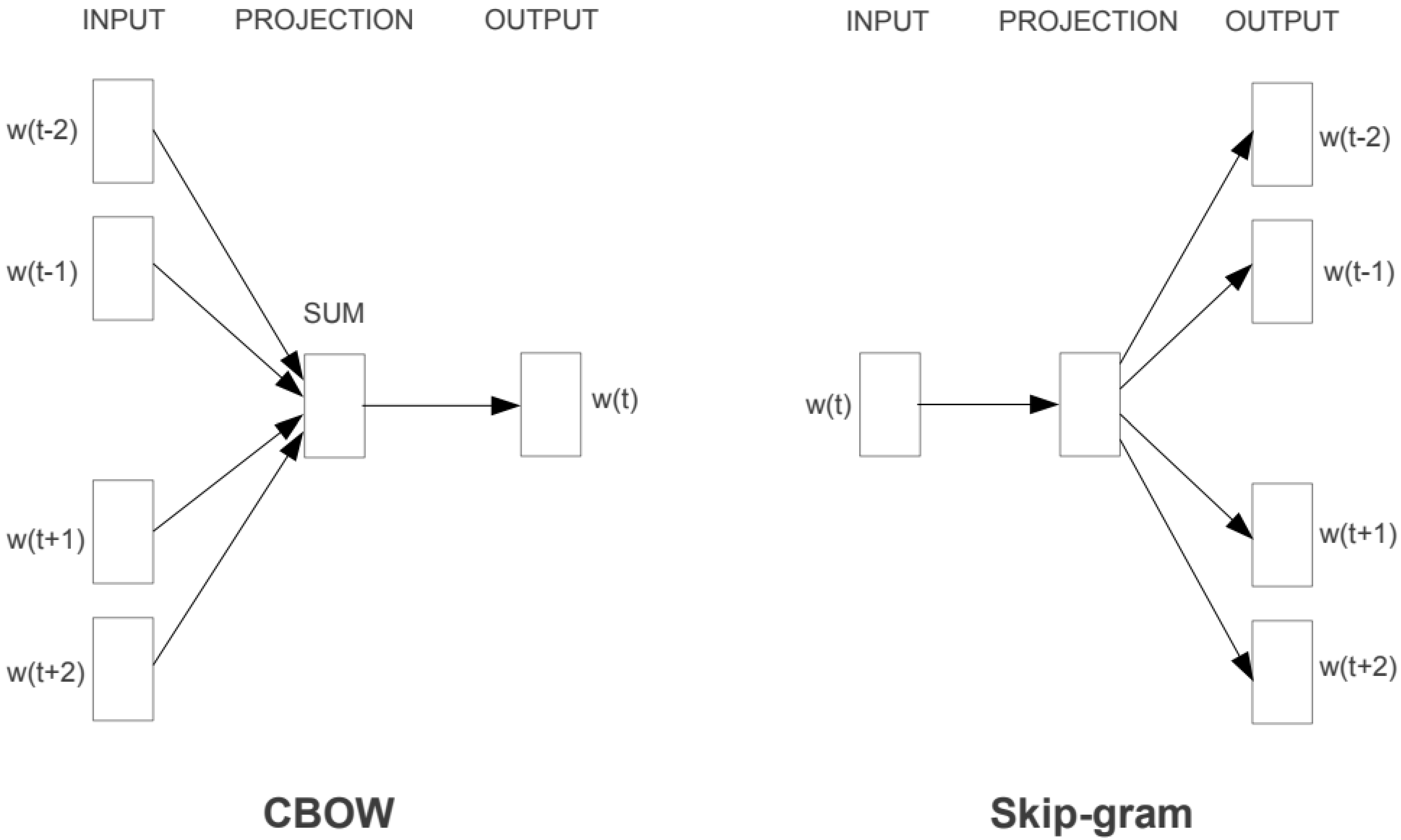


(Mikolov et al., NAACL HLT, 2013)

Let’s Try It

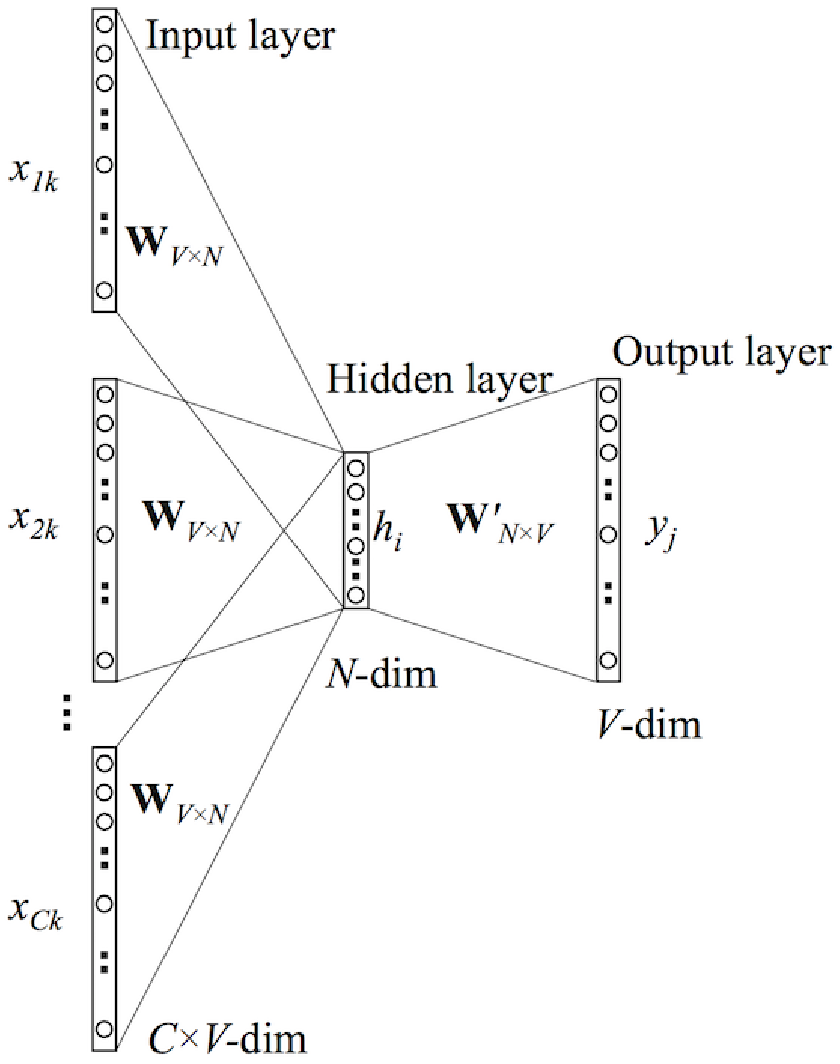
- http://w.elnn.kr/search/

CBOW & Skip-gram



CBOW – Continuous Bag of Words

- Fill the blank
  - 아이스크림을 사 먹었는데, ___ 시려서 먹기가 힘들었다.
- 앞 뒤로 C/2개의 단어를 input으로 하여 center 단어를 맞추도록 학습
- Input은 one-hot encoding
- Input → Hidden layer는 linear mapping(avg(Wx_{ik}))
- Hidden → Output layer는 Softmax($W'h_i$)



Skip-gram

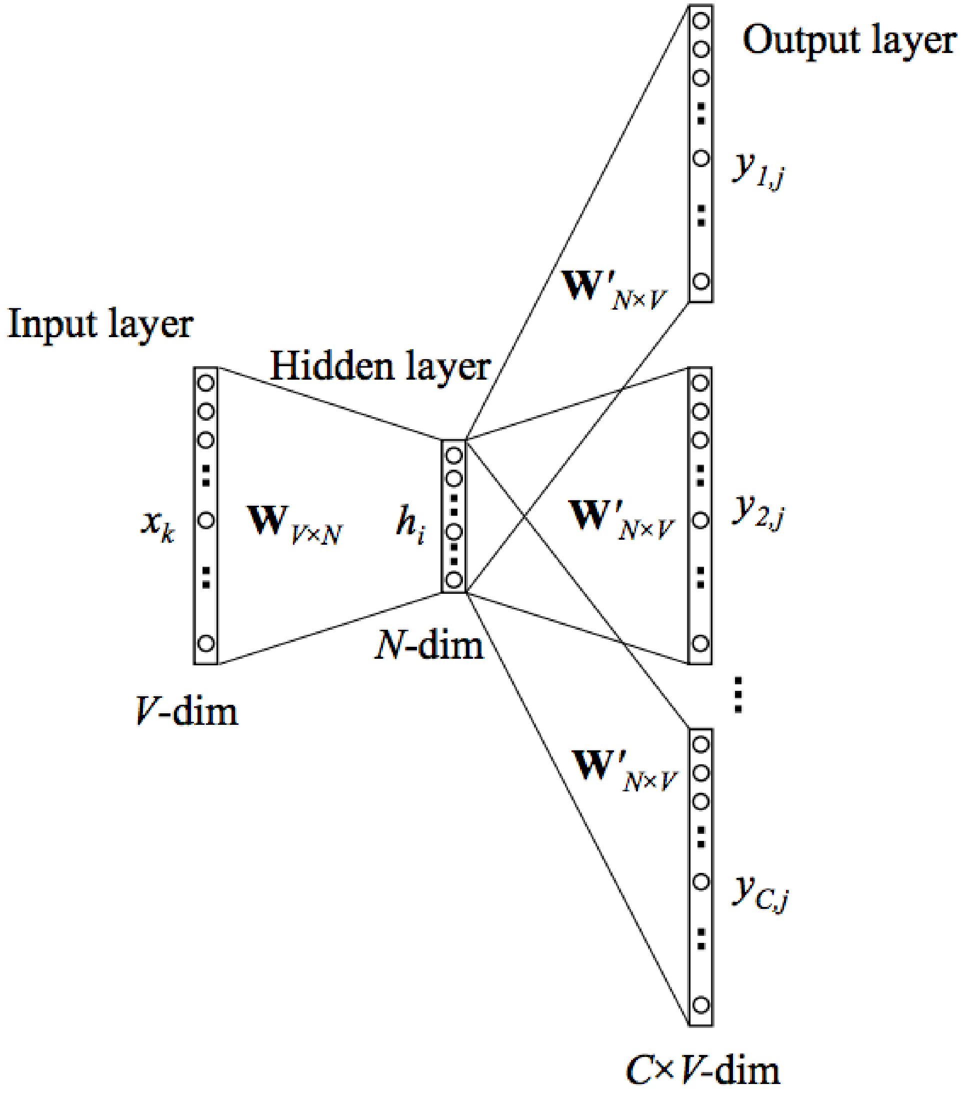
- CBOW와 반대로 중심 단어를 주고 주변 단어들에 대한 확률 값을 출력함
- Window 내에 있는 단어의 확률이 최대가 되도록 학습
- Objective function
  - Maximize

$$J'(\theta) = \prod_{t=1}^{T} \prod_{-C/2 \le j \le C/2, j \ne 0} P(x_{t+j} | x_t ; \theta)$$

→ Negative log likelihood

  - Minimize

$$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-C/2 \le j \le C/2, j \ne 0} log P(x_{t+j} | x_t ; \theta)$$



How to Represent a Token

- How do should we represent a token so that it reflects its "meaning"?
- First, we assume nothing is known: use an one-hot encoding.
- Second, the neural network capture the token's meaning as a vector.
- This is done by a simple matrix multiplication:
  $Wx = W[\hat{x}]$, if x is one-hot,
  where $\hat{x} = \arg\max_j x_j$ is the token's index in the vocabulary.

Table Lookup

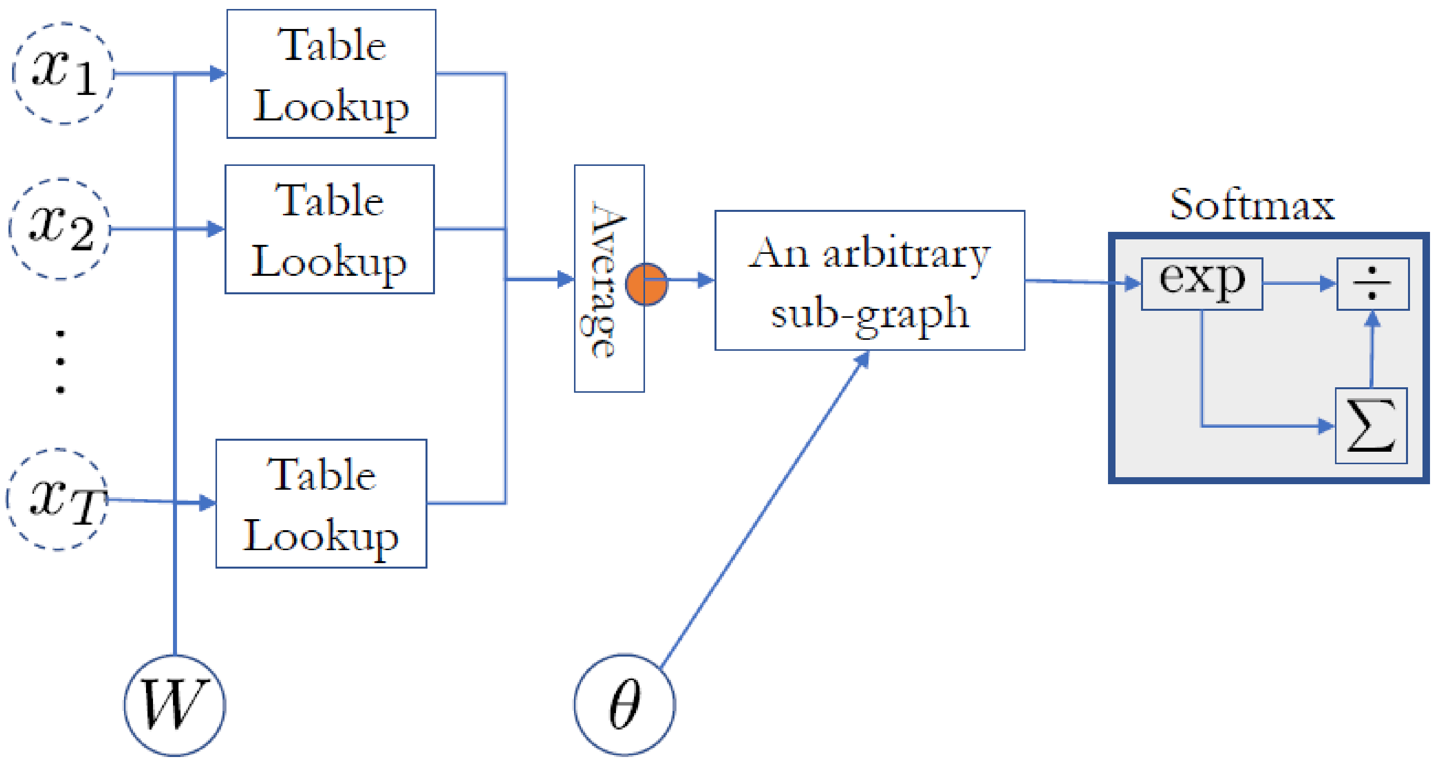
$$W^\top = \begin{bmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{\hat{x}1} & \cdots & w_{\hat{x}d} \\ \vdots & \ddots & \vdots \\ w_{|V|1} & \cdots & w_{|V|d} \end{bmatrix}$$

How to Represent a Sentence

- 단어는 Word2Vec을 사용하면 될텐데, 문장은 어떻게 해야할까?
- 문장 = 단어들의 sequence
- Sequence의 길이가 문장마다 모두 다름 → fixed length vector로 표현하는 방법을 찾아야 함

How to Represent a Sentence - CBOW

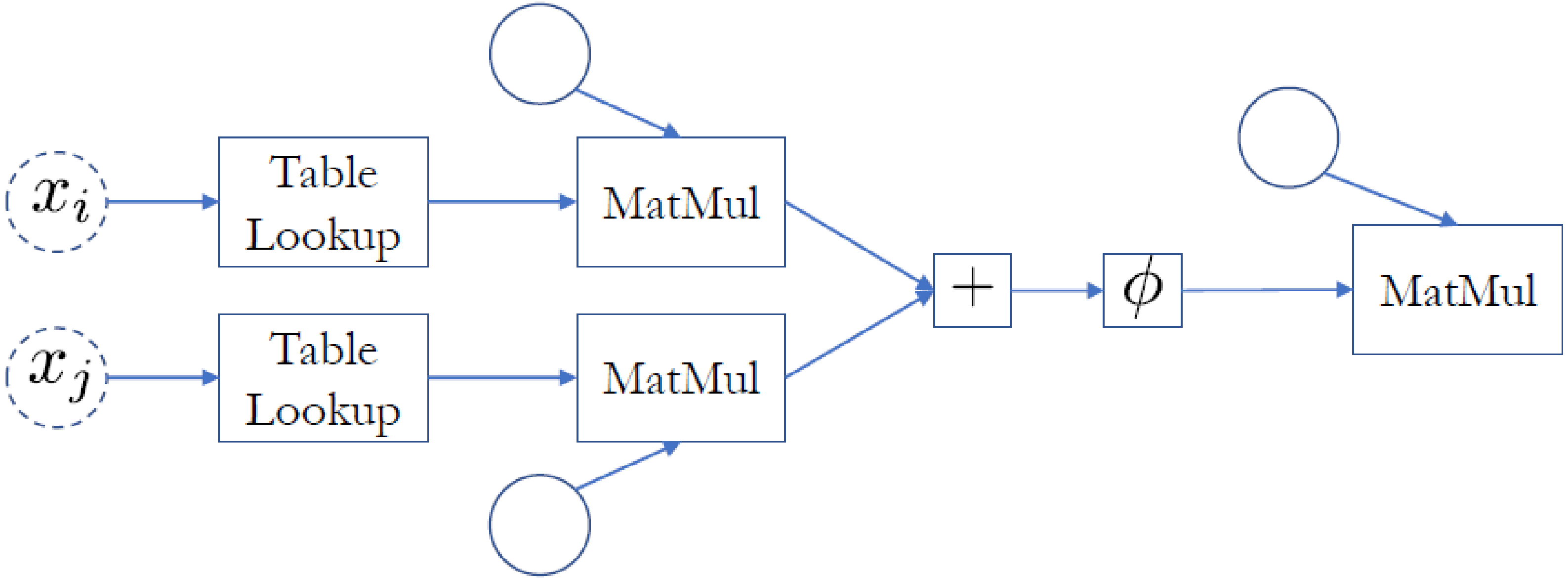
- Continuous bag-of-words based multi-class text classifier



- With this DAG, you use automatic backpropagation and stochastic gradient descent to train the classifier.

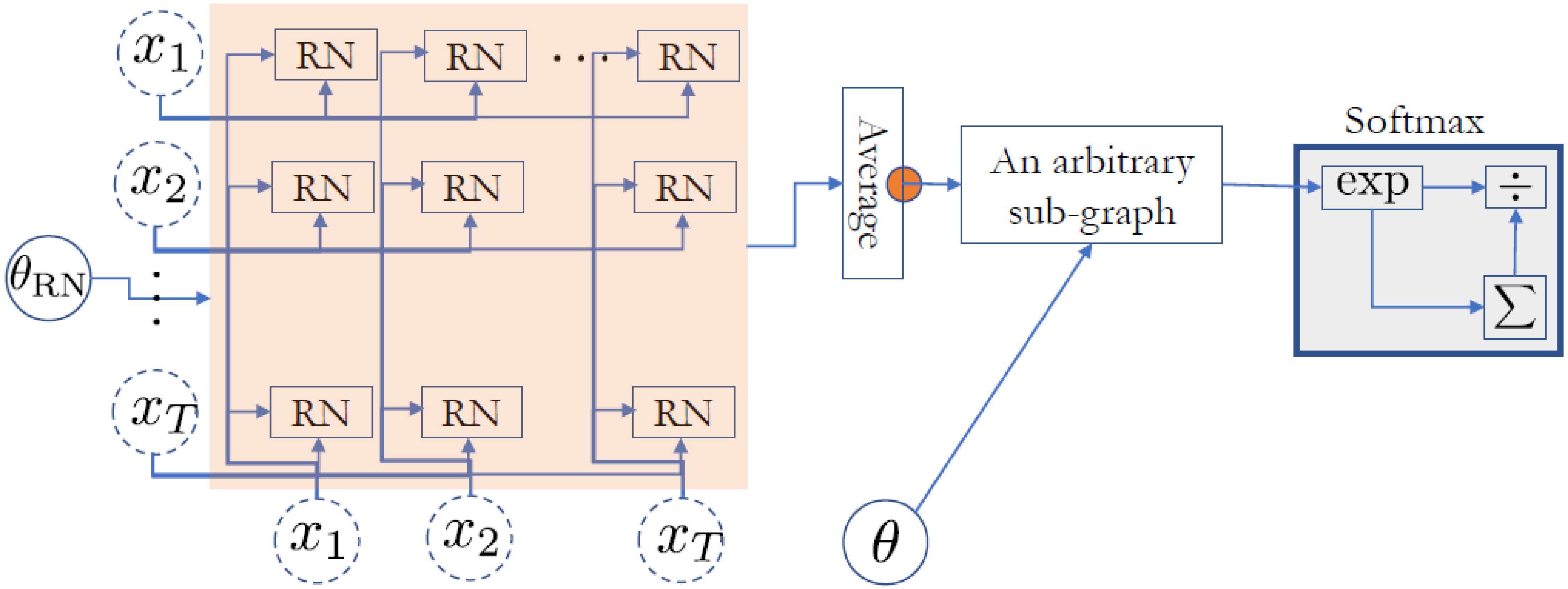
How to Represent a Sentence - RN

- Relation Network [Santoro et al., 2017]: Skip Bigrams
  - Consider all possible pairs of tokens: $(x_i, x_j), \forall i \neq j$
  - Combine two token vectors with a neural network for each pair
    $$f(x_i, x_j) = W\phi(U_{\text{left}} e_i + U_{\text{right}} e_j)$$
    - ϕ is a element-wise nonlinear function, such as tanh or ReLU ($\max(0, a)$)
    - One subgraph in the DAG.



How to Represent a Sentence - RN

- Relation Network: Skip Bigrams
  - Considers all possible pairs of tokens: $(x_i, x_j), \forall i \neq j$
    $f(x_i, x_j) = W\phi(U_{\text{left}} e_i + U_{\text{right}} e_j)$
  - Considers the pair-wise "relation"ship $\text{RN}(X) = \frac{1}{2N(N-1)} \sum_{i=1}^{T-1} \sum_{j=i+1}^{T} f(x_i, x_j)$
  - Averages all these relationship vectors

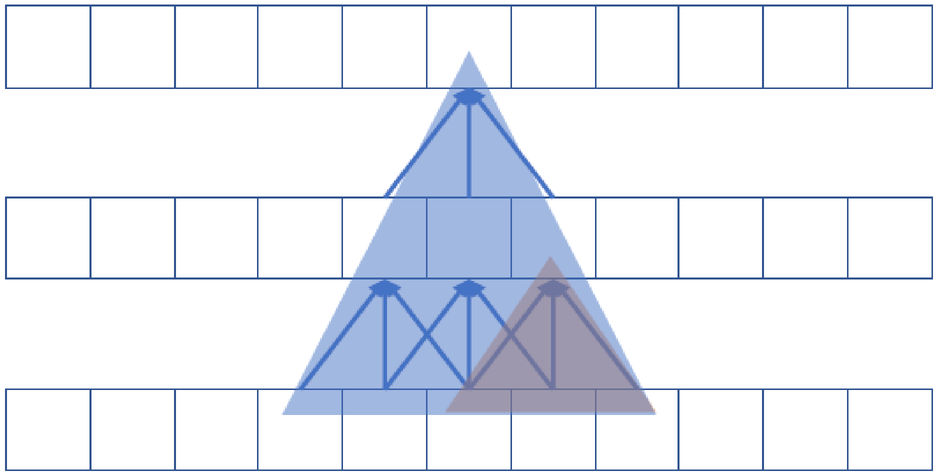


How to Represent a Sentence - CNN

- Convolutional Networks [Kim, 2014; Kalchbrenner et al., 2015]
  - Captures k-grams hierarchically
  - One 1-D convolutional layer: considers all k-grams

$$h_t = \phi\left(\sum_{\tau=-k/2}^{k/2} W_\tau e_{t+\tau}\right), \text{ resulting in } H = (h_1, h_2, \ldots, h_T).$$

  - Stack more than one convolutional layers: progressively-growing window
  - Fits our intuition of how sentence is understood: **tokens→multi-word expressions→phrases→sentence**



How to Represent a Sentence – Self Attention

- Can we combine and generalize the relation network and the CNN?
- Relation Network:
  - Each token's representation is computed against all the other tokens

$$h_t = f(x_t, x_1) + \cdots + f(x_t, x_{t-1}) + f(x_t, x_{t+1}) + \cdots + f(x_t, x_T)$$

- CNN:
  - Each token's representation is computed against neighbouring tokens

$$h_t = f(x_t, x_{t-k}) + \cdots + f(x_t, x_t) + \cdots + f(x_t, x_{t+k})$$

- RN considers the entire sentence vs. CNN focuses on the local context.

How to Represent a Sentence – Self Attention

- Can we combine and generalize the relation network and the CNN?
- CNN as a weighted relation network:
  - Original: $h_t = f(x_t, x_{t-k}) + \cdots + f(x_t, x_t) + \cdots + f(x_t, x_{t+k})$
  - Weighted:

$$h_t = \sum_{t'=1}^{T} \mathbb{I}(|t' - t| \leq k) f(x_t, x_{t'})$$

where $\mathbb{I}(S) = 1, \text{ if } S \text{ is true, and } 0, \text{otherwise}$.

- Can we compute those weights instead of fixing them to 0 or 1?

How to Represent a Sentence – Self Attention

- Can we compute those weights instead of fixing them to 0 or 1?
- That is, compute the weight of each pair $(x_t, x_{t'})$

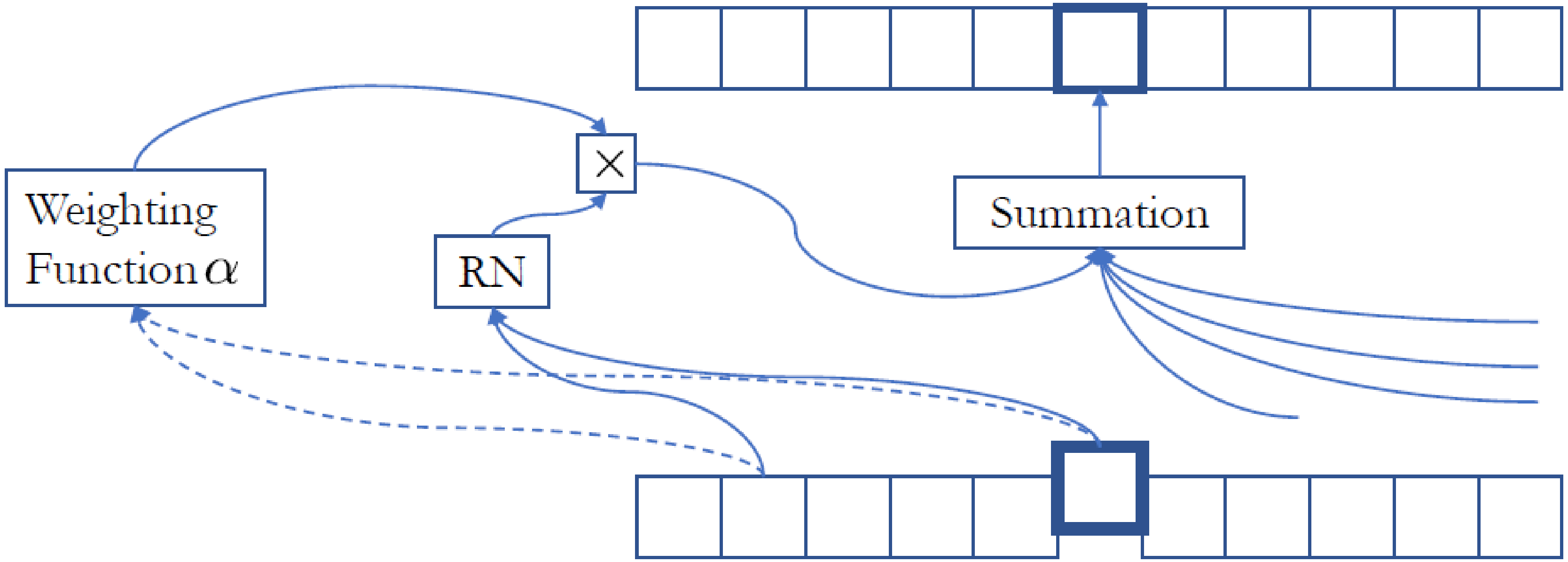
$$h_t = \sum_{t'=1}^{T} \alpha(x_t, x_{t'}) f(x_t, x_{t'})$$

- The weighting function could be yet another neural network
  - Just another subgraph in a DAG: easy to use!
    $\alpha(x_t, x_{t'}) = \sigma(\mathrm{RN}(x_t, x_{t'})) \in [0, 1]$
  - Perhaps we want to normalize them so that the weights sum to one

$$\alpha(x_t, x_{t'}) = \frac{\exp(\beta(x_t, x_{t'}))}{\sum_{t''=1}^{T} \exp(\beta(x_t, x_{t''}))}, \text{where } \beta(x_t, x_{t'}) = \mathrm{RN}(x_t, x_{t'}))$$

How to Represent a Sentence – Self Attention

- Self-Attention: a generalization of CNN and RN.
- Able to capture long-range dependencies within a single layer.
- Able to ignore irrelevant long-range dependencies.

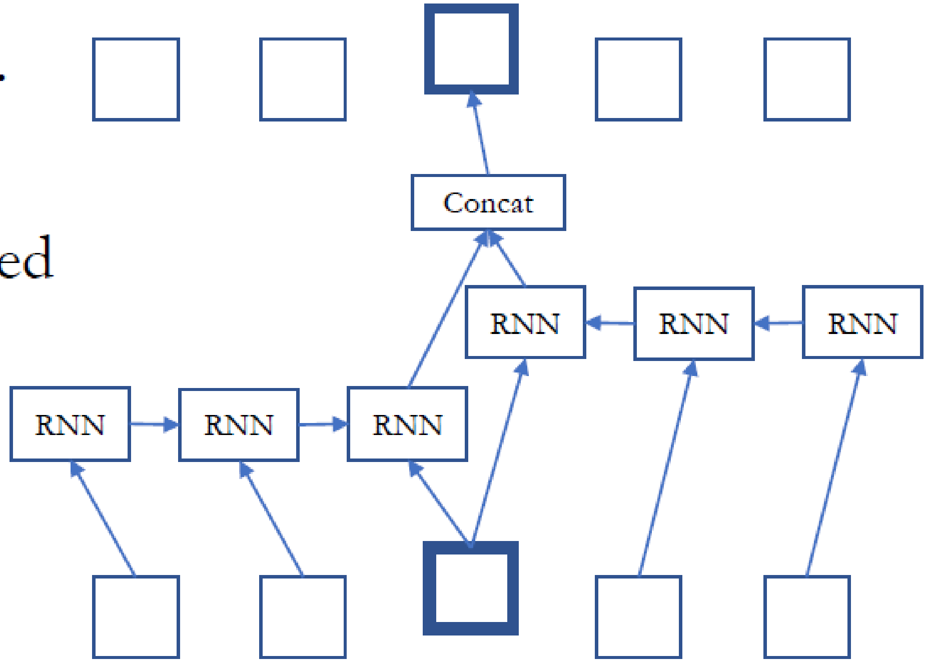


How to Represent a Sentence – RNN

- Recurrent neural network: online compression of a sequence $O(T)$

  $h_t = \text{RNN}(h_{t-1}, x_t)$, where $h_0 = 0$.

- Bidirectional RNN to account for both sides.
- Inherently sequential processing
  - Less desirable for modern, parallelized, distributed computing infrastructure.
- LSTM [Hochreiter&Schmidhuber, 1999] and GRU [Cho et al., 2014] have become de facto standard
  - All standard frameworks implement them.
  - Efficient GPU kernels are available.



Language Model

- Input: a sentence
- Output: the probability of the input sentence
- A language model captures the distribution over all possible sentences.

  $$p(X) = p((x_1, x_2, \ldots, x_T))$$

- Unlike text classification, it is *unsupervised learning*.
  - We will however turn the problem into a *sequence of supervised learning*.

Autoregressive Language Model

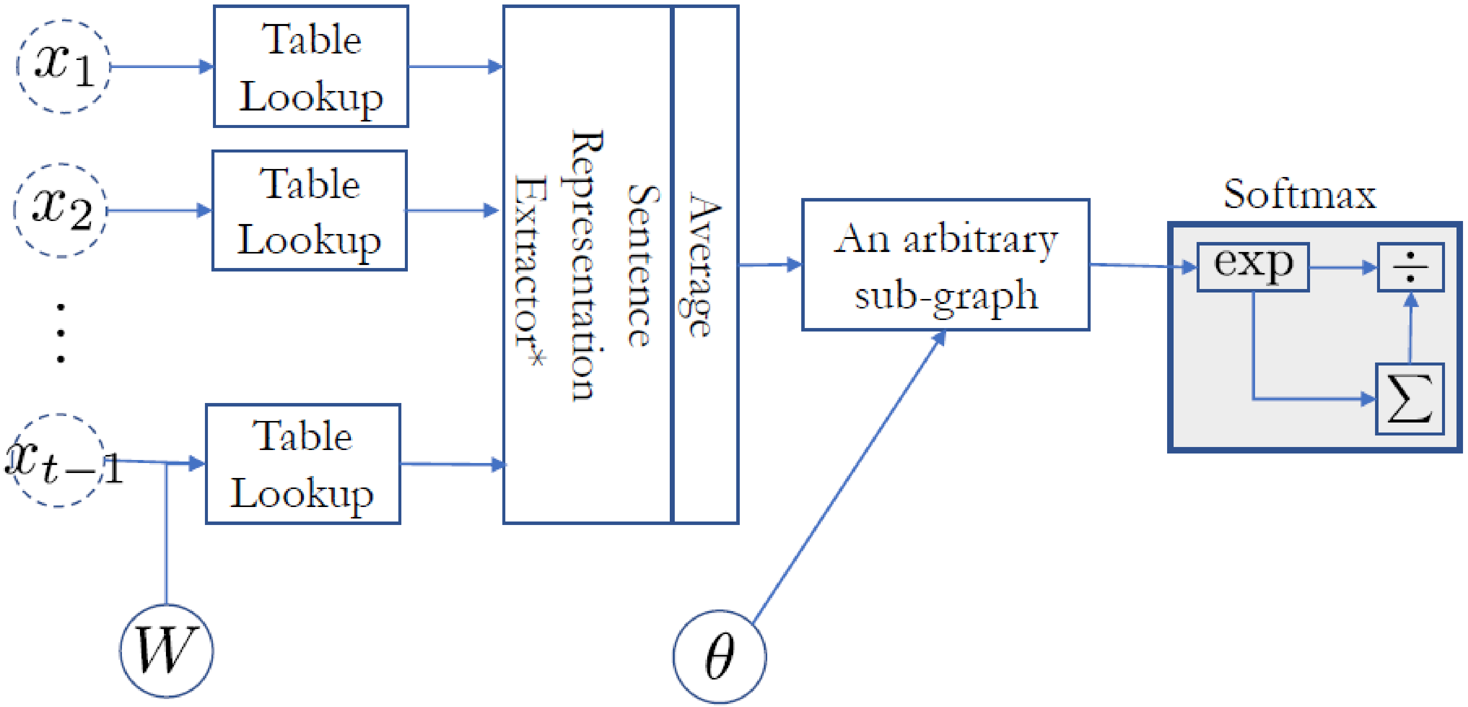
- Autoregressive sequence modelling
  - The distribution over the next token is based on all the previous tokens.
    $p(X) = p(x_1)p(x_2|x_1) \cdots p(x_T|x_1, \ldots, x_{T-1})$
  - This equality holds exactly due to the def. of conditional distribution*
- Unsupervised learning becomes a set of supervised problems.
  - Each conditional is a neural network classifier.
  - Input is all the previous tokens (a partial sentence).
  - Output is the distribution over all possible next tokens (classes).
  - It is a **text classification** problem.

Autoregressive Language Model

- Autoregressive sequence modelling
  - The distribution over the next token is based on all the previous tokens.

$$p(X) = p(x_1)p(x_2|x_1)\cdots p(x_T|x_1,\ldots,x_{T-1})$$

- Each conditional is a sentence classifier:



Machine Translation

- Input: a sentence written in a source language L_S
- Output: a corresponding sentence in a target language L_T
- Problem statement:
  - Supervised learning: given the input sentence, output its translation
  - Compute the conditional distribution over all possible translation given the input

$$p(Y = (y_1, \ldots, y_T) | X = (x_1, \ldots, x_{T'}))$$

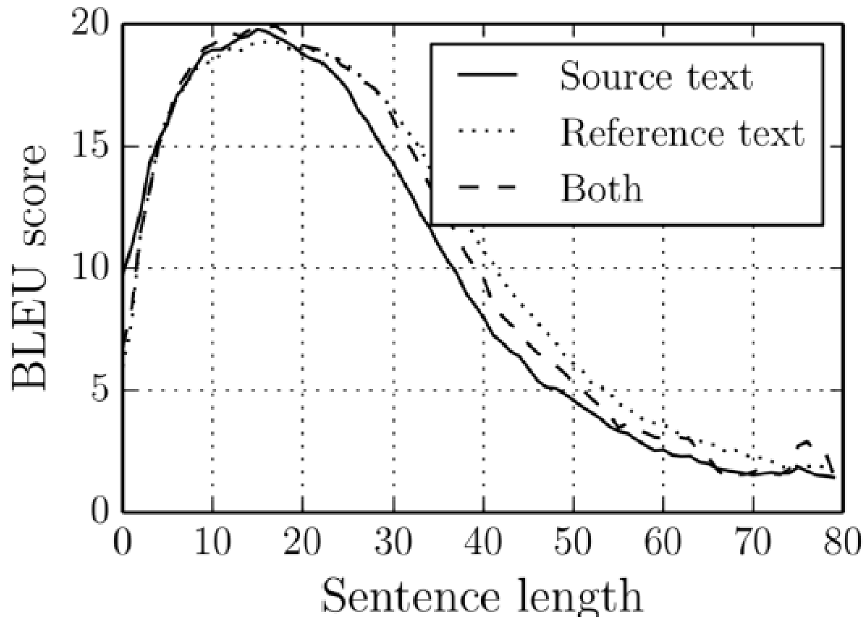
- *We have already learned every necessary ingredient for building a full neural machine translation system.*

Encoder – Source Sentence Representation

- Encode the source sentence into a set of sentence representation vectors
  - # of encoded vectors is proportional to the source sentence length: often same.
    $H = (h_1, \ldots, h_{T'})$
  - Recurrent networks have been widely used [Cho et al., 2014; Sutskever et al., 2014], but CNN [Gehring et al., 2017; Kalchbrenner&Blunsom, 2013] and self-attention [Vaswani et al., 2017] are used increasingly more often. See Lecture 2 for details.
- We do not want to collapse them into a single vector.
  - Collapsing often corresponds to information loss.
  - Increasingly more difficult to encode the entire source sentence into a single vector, as the sentence length increases [Cho et al., 2014b].
  - We didn't know initially until [Bahdanau et al., 2015].

Encoder – Source Sentence Representation

- Encode the source sentence into a set of sentence representation vectors
- We do not want to collapse them into a single vector.
  - Increasingly more difficult to encode the entire source sentence into a single vector, as the sentence length increases [Cho et al., 2014b].



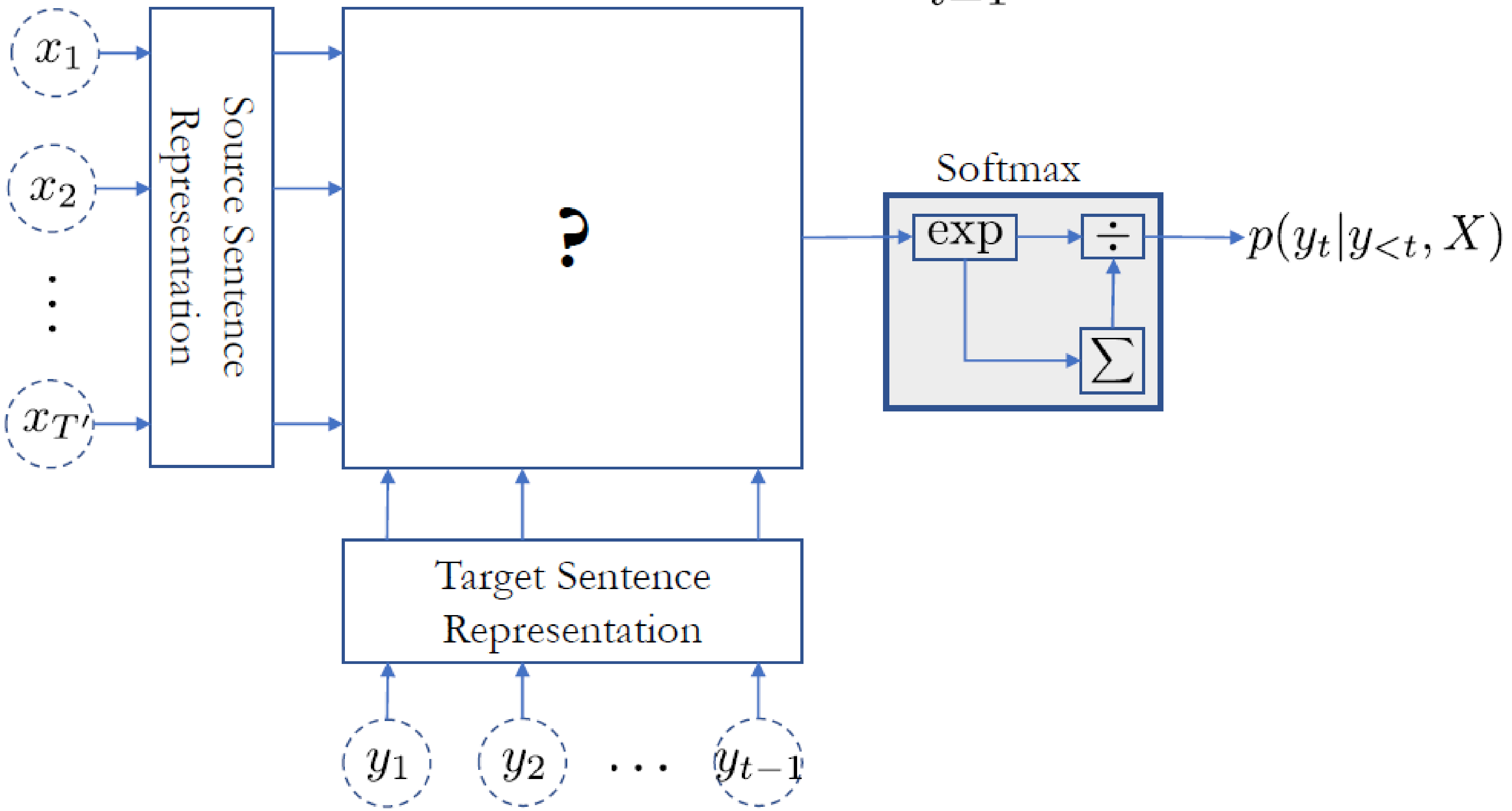
Decoder – Language Modelling

- Autoregressive Language modelling with an infinite context $n \to \infty$
  - Larger context is necessary to generate a coherent sentence.
    - Semantics could be largely provided by the source sentence, but syntactic properties need to be handled by the language model directly.
  - Recurrent networks, self-attention and (dilated) convolutional networks
    - Causal structure must be followed.
    - See Lecture 3.
- **Conditional** Language modelling
  - The context based on which the next token is predicted is **two-fold**

$$p(Y|X) = \prod_{t=1}^{T} p(y_t | y_{<t}, X)$$

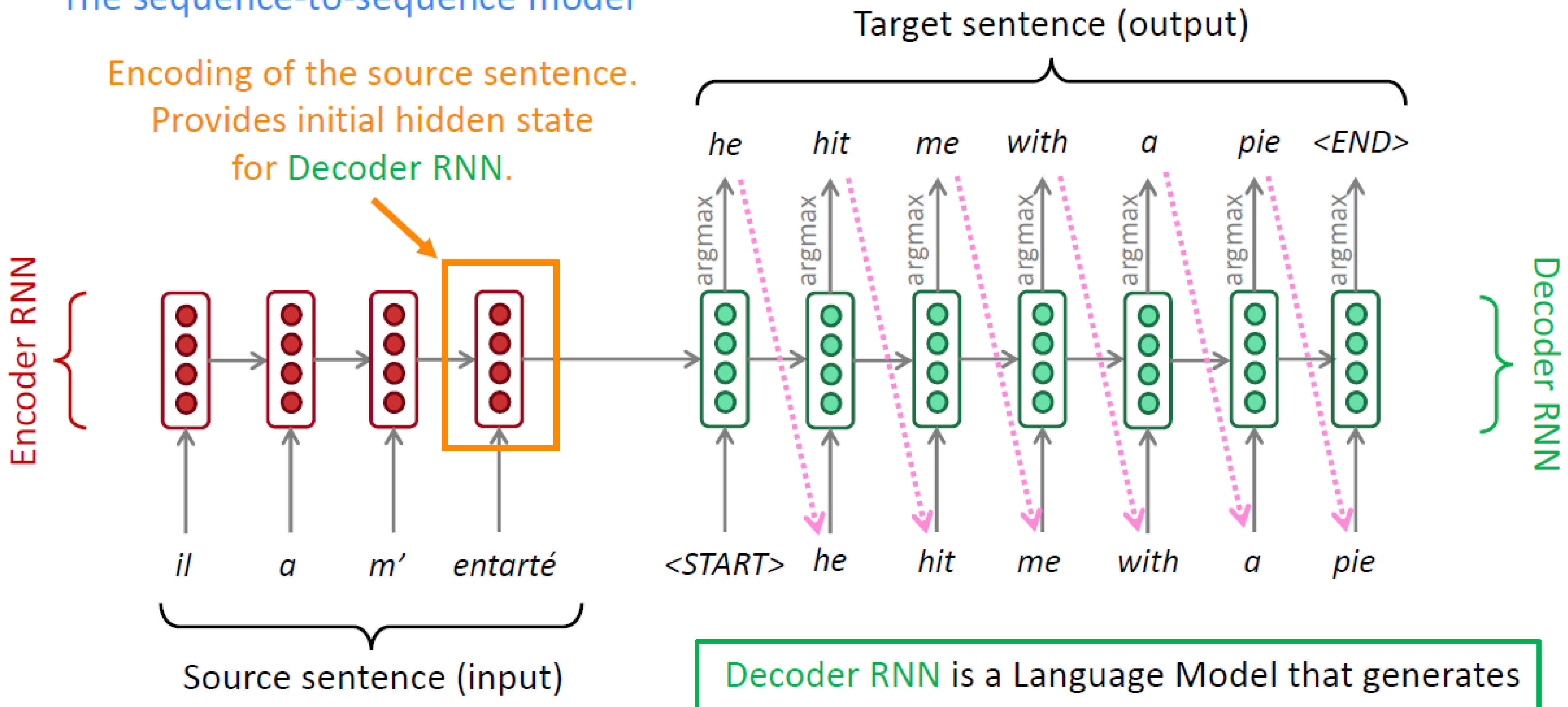
Decoder – Language Modelling

- Conditional Language modelling $p(Y|X) = \prod_{t=1}^{T} p(y_t|y_{<t}, X)$



Neural Machine Translation (NMT)

The sequence-to-sequence model



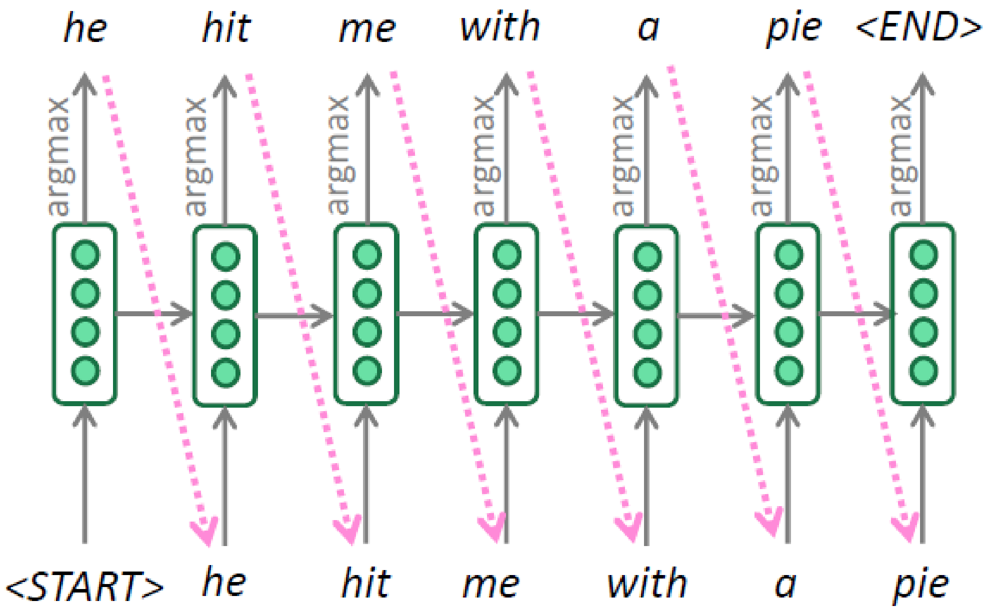
Decoder RNN is a Language Model that generates target sentence, *conditioned on* encoding.

Encoder RNN produces an encoding of the source sentence.

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

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?**

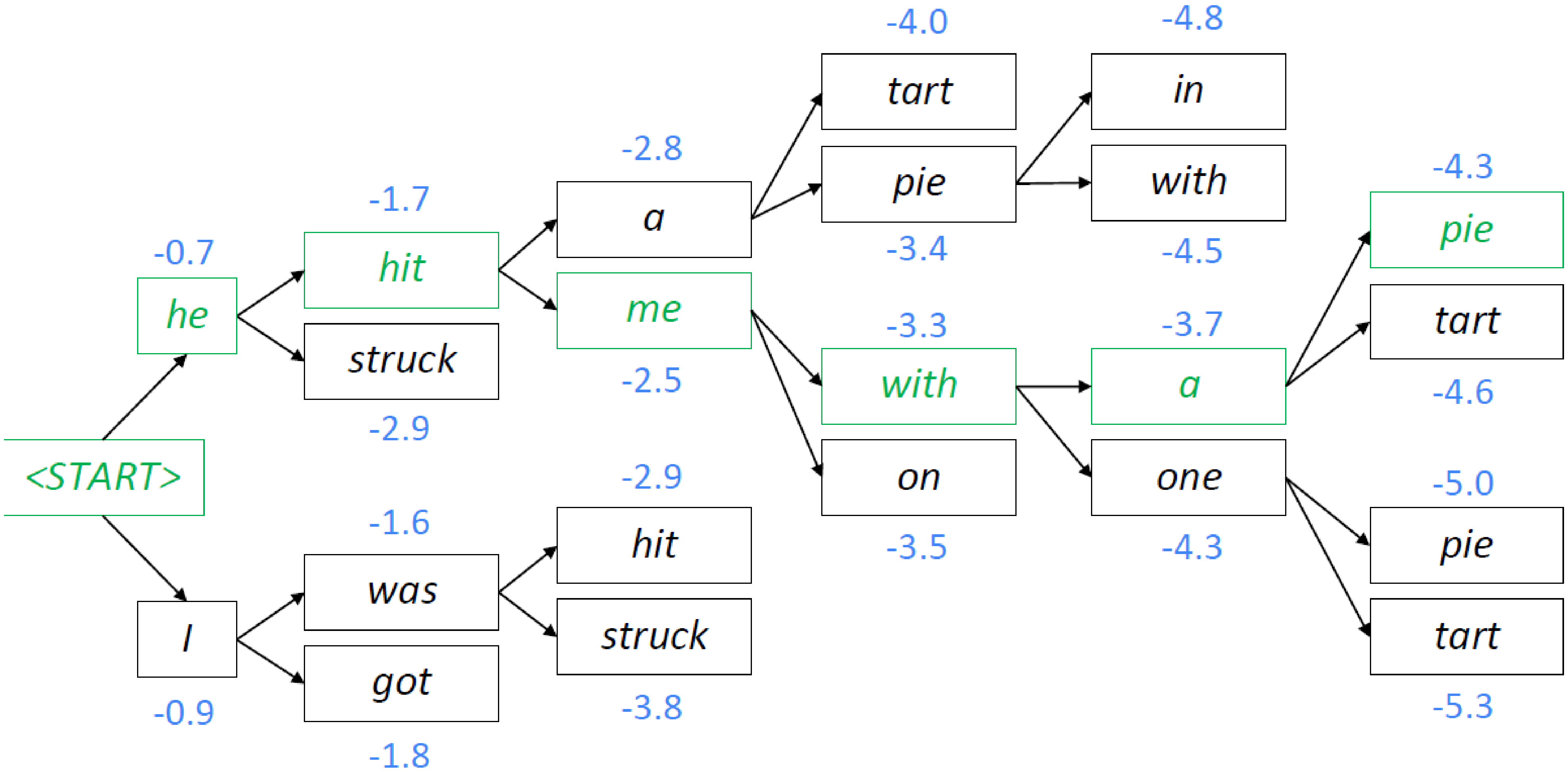
Problems with Greedy Decoding

- 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…)

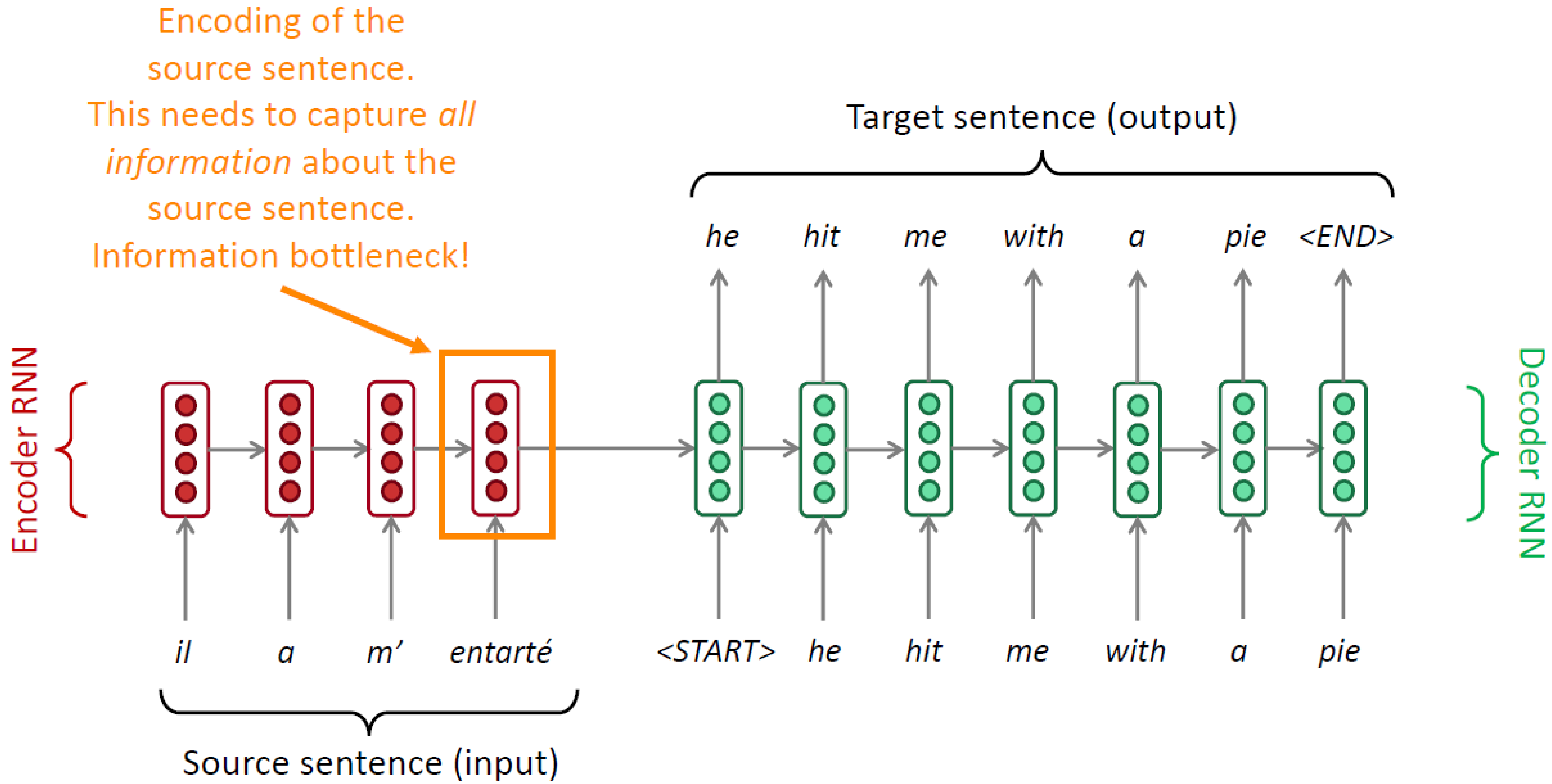
- How to fix this?

Beam Search Decoding

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



Sequence-to-Sequence: the Bottleneck Problem



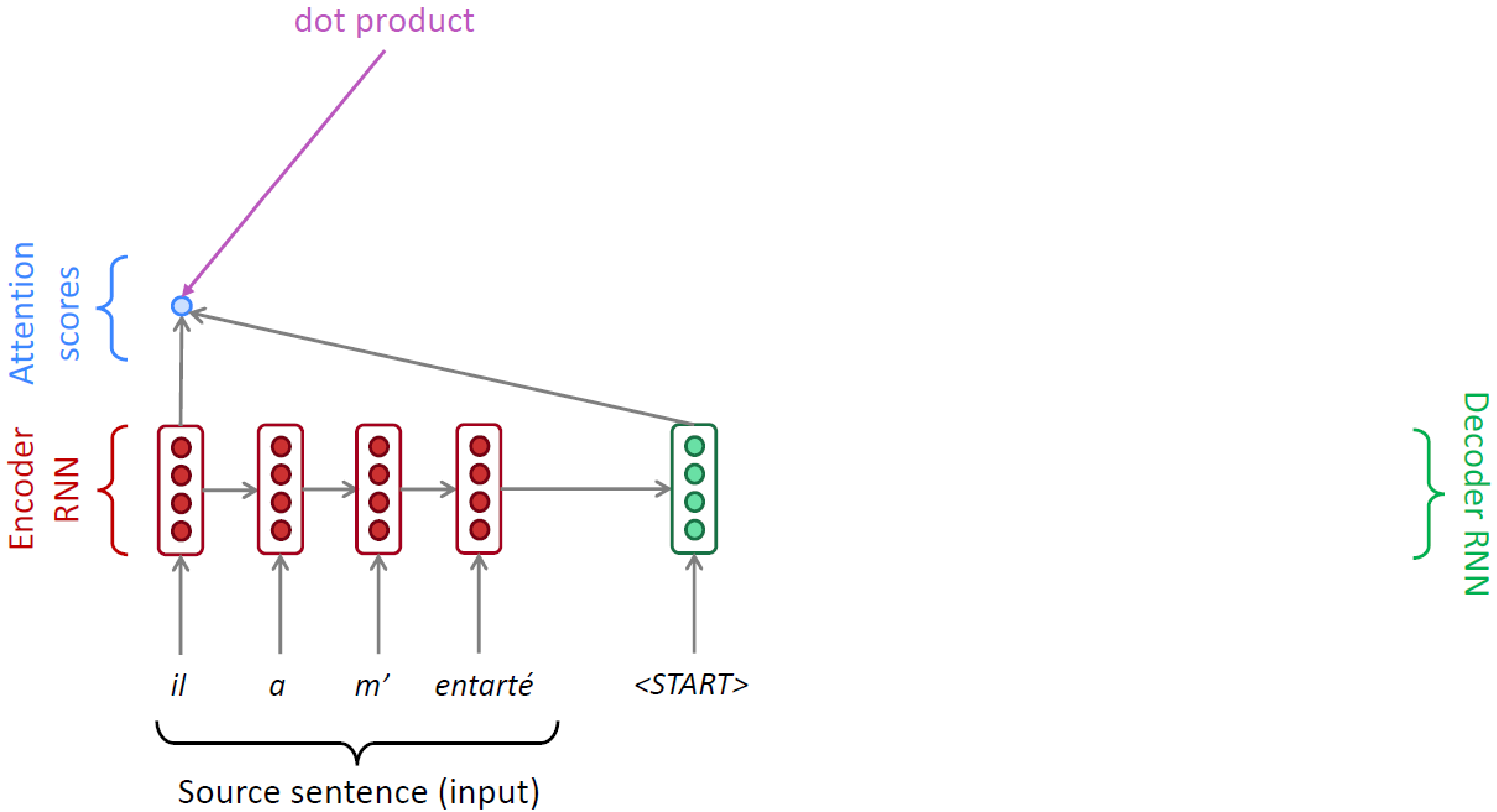
Attention

- **Attention** provides a solution to the bottleneck problem.

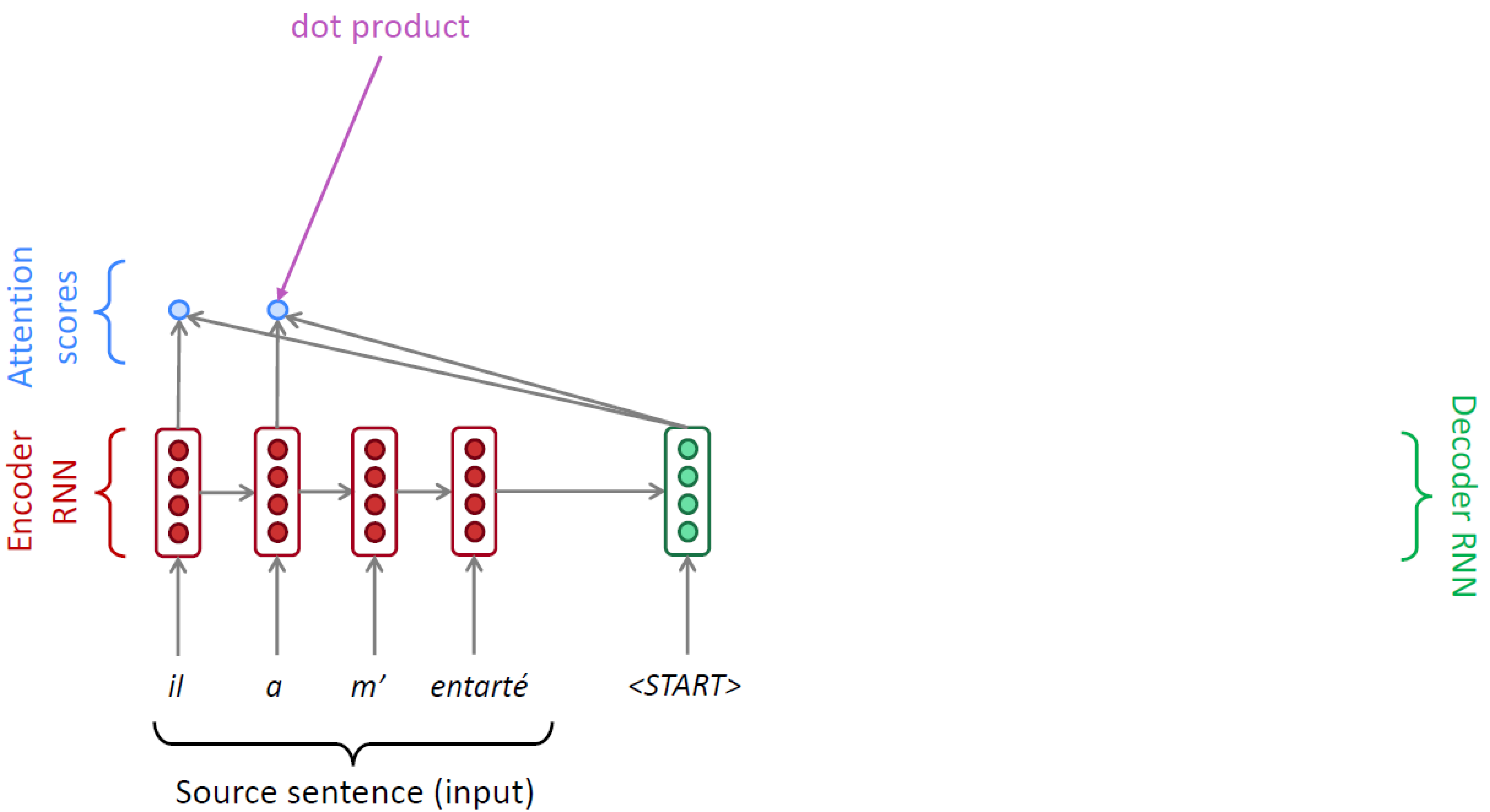
- Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence

- First we will show via diagram (no equations), then we will show with equations

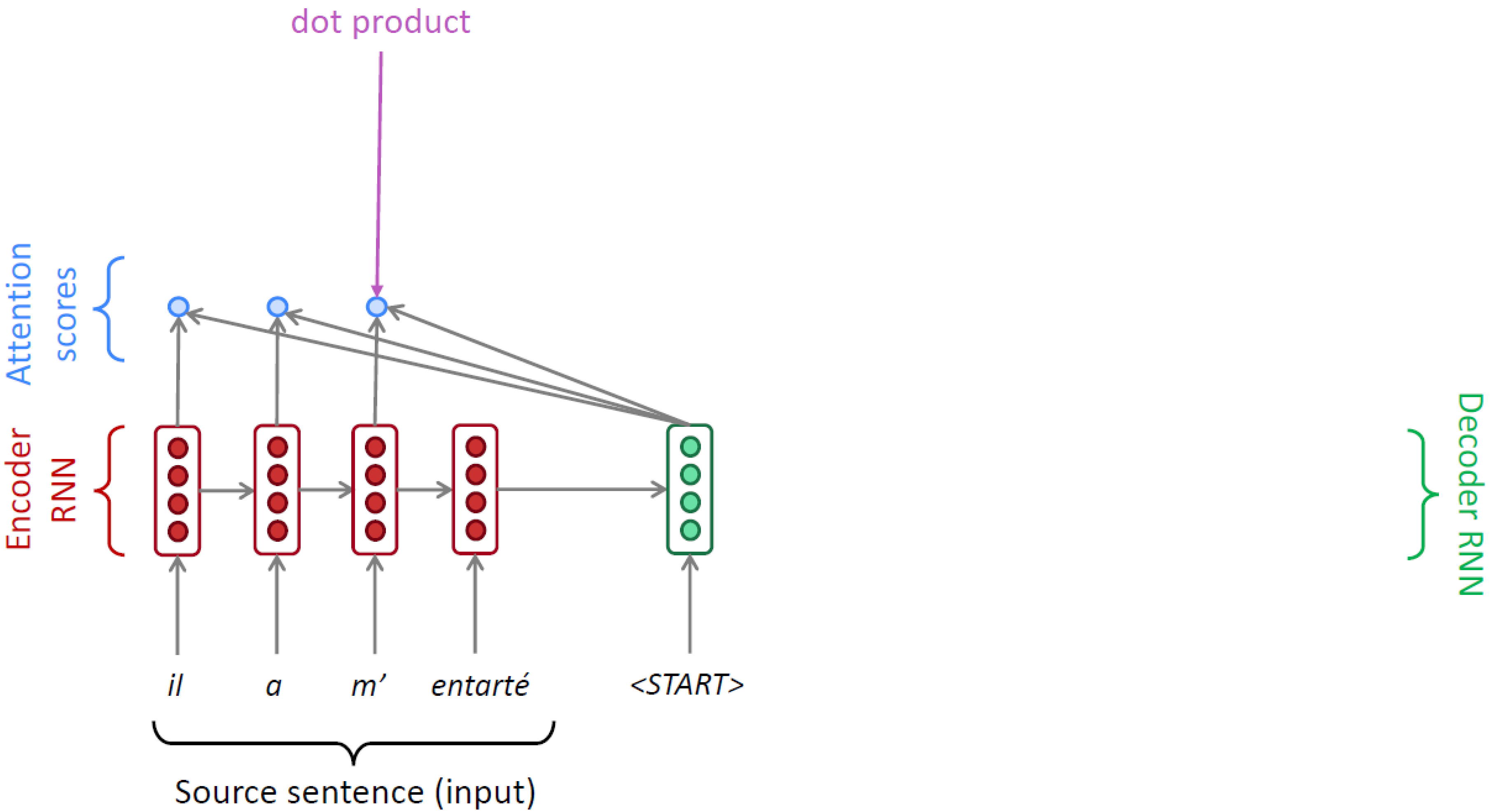
Sequence-to-Sequence with Attention



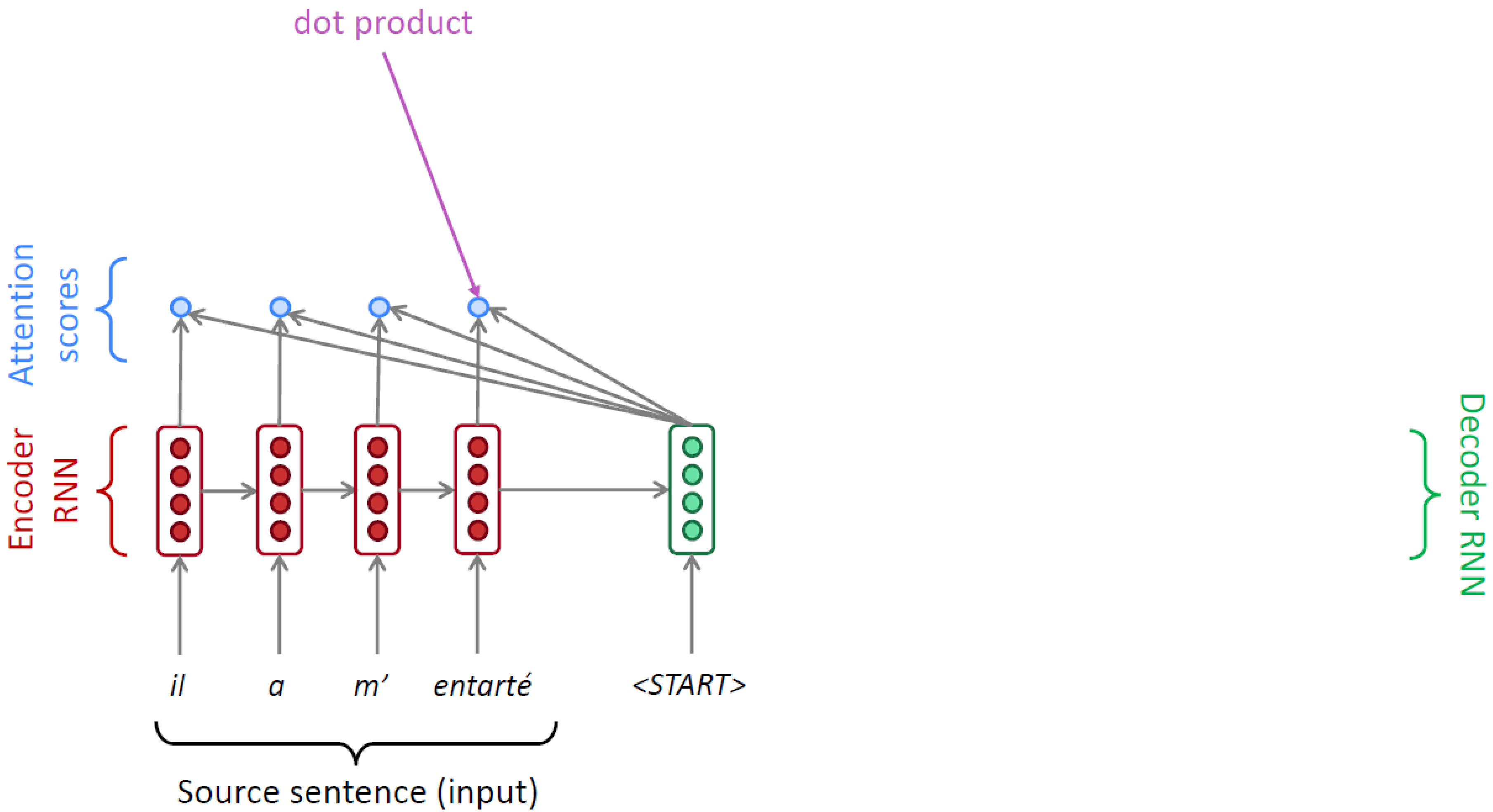
Sequence-to-Sequence with Attention



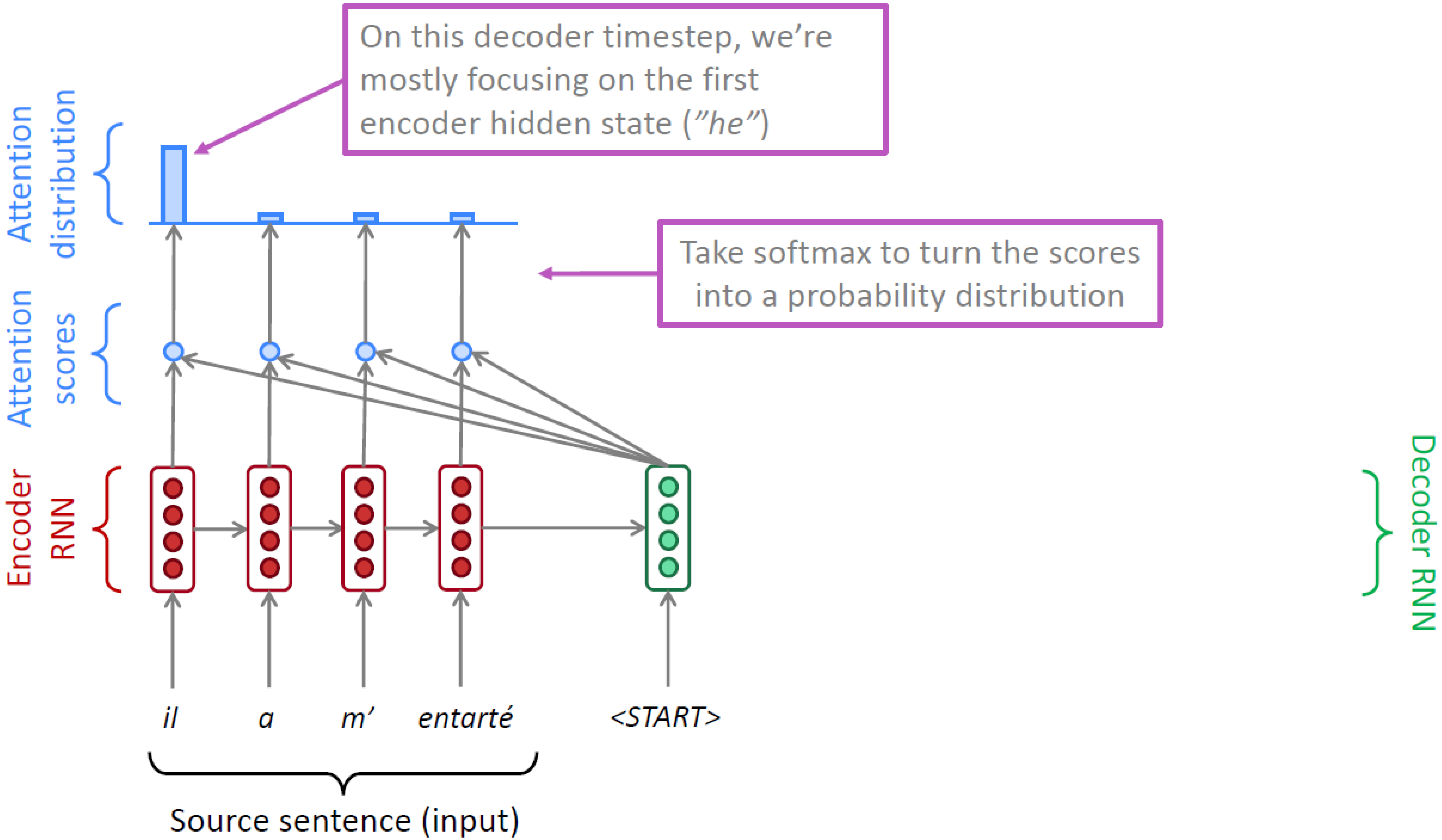
Sequence-to-Sequence with Attention



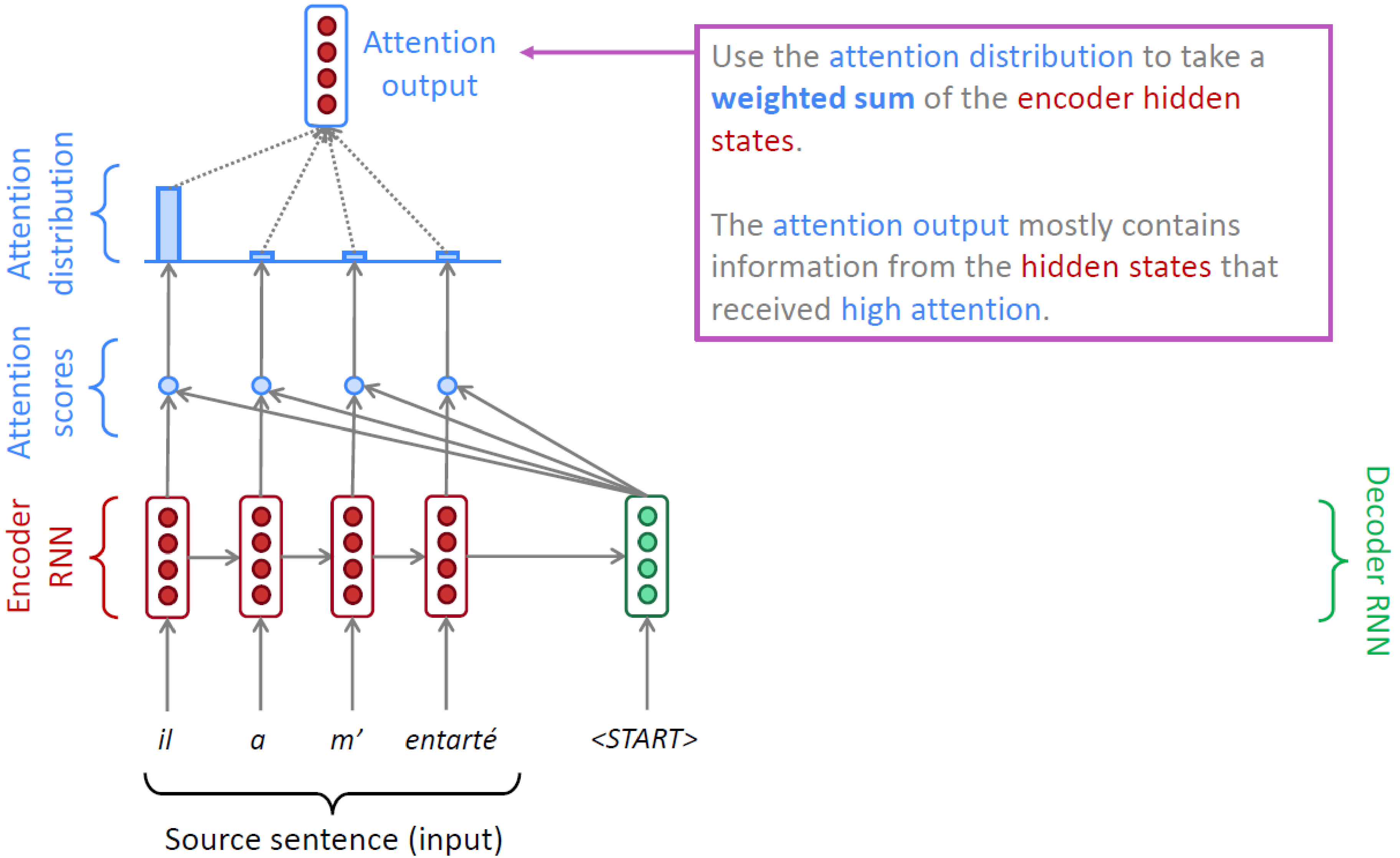
Sequence-to-Sequence with Attention



Sequence-to-Sequence with Attention



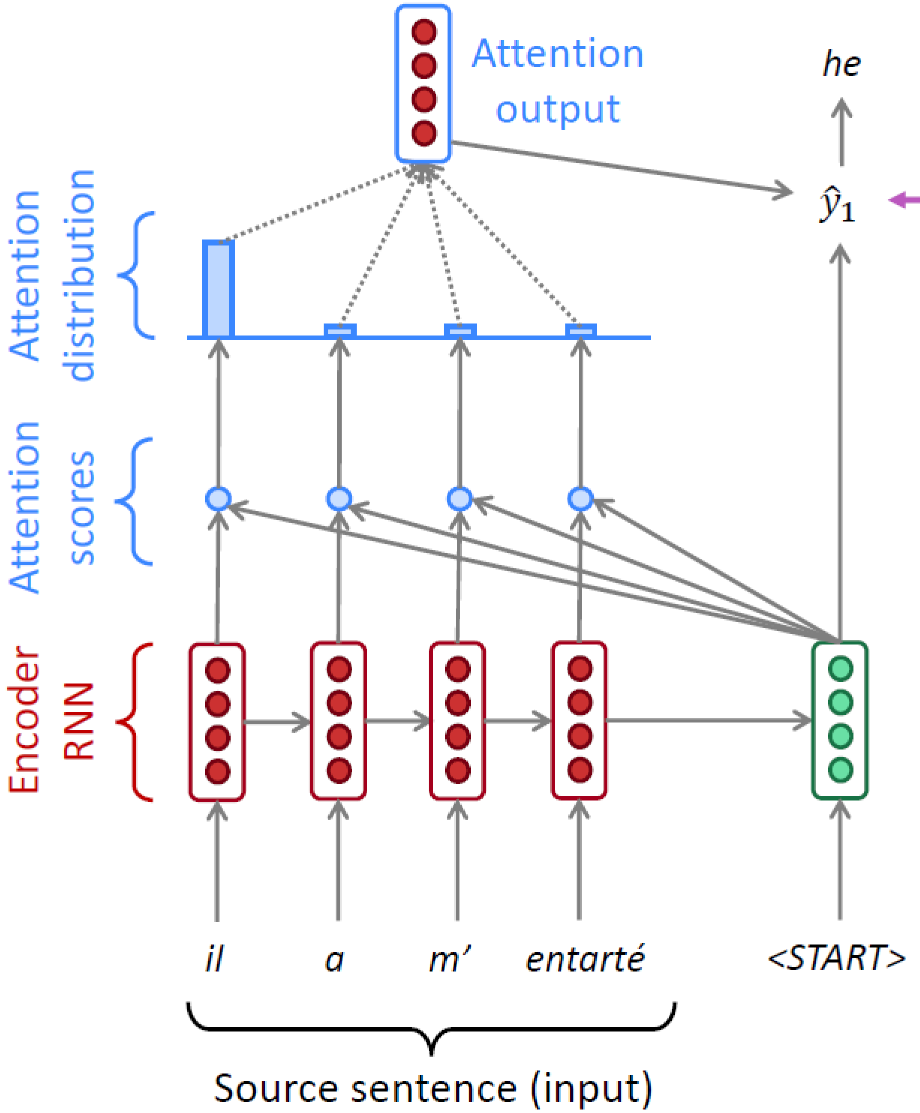
Sequence-to-Sequence with Attention



Use the attention distribution to take a **weighted sum** of the encoder hidden states.

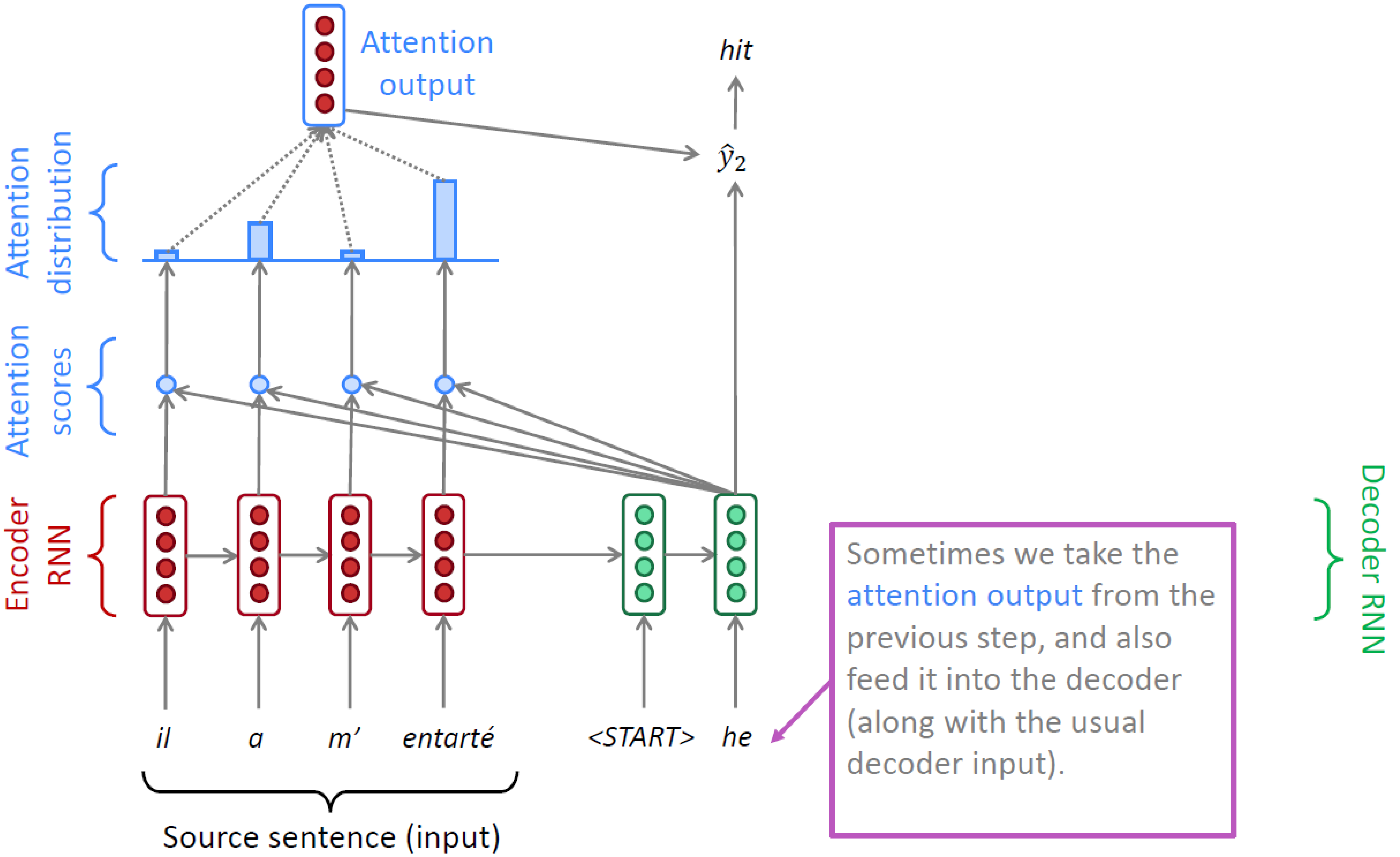
The attention output mostly contains information from the hidden states that received high attention.

Sequence-to-Sequence with Attention



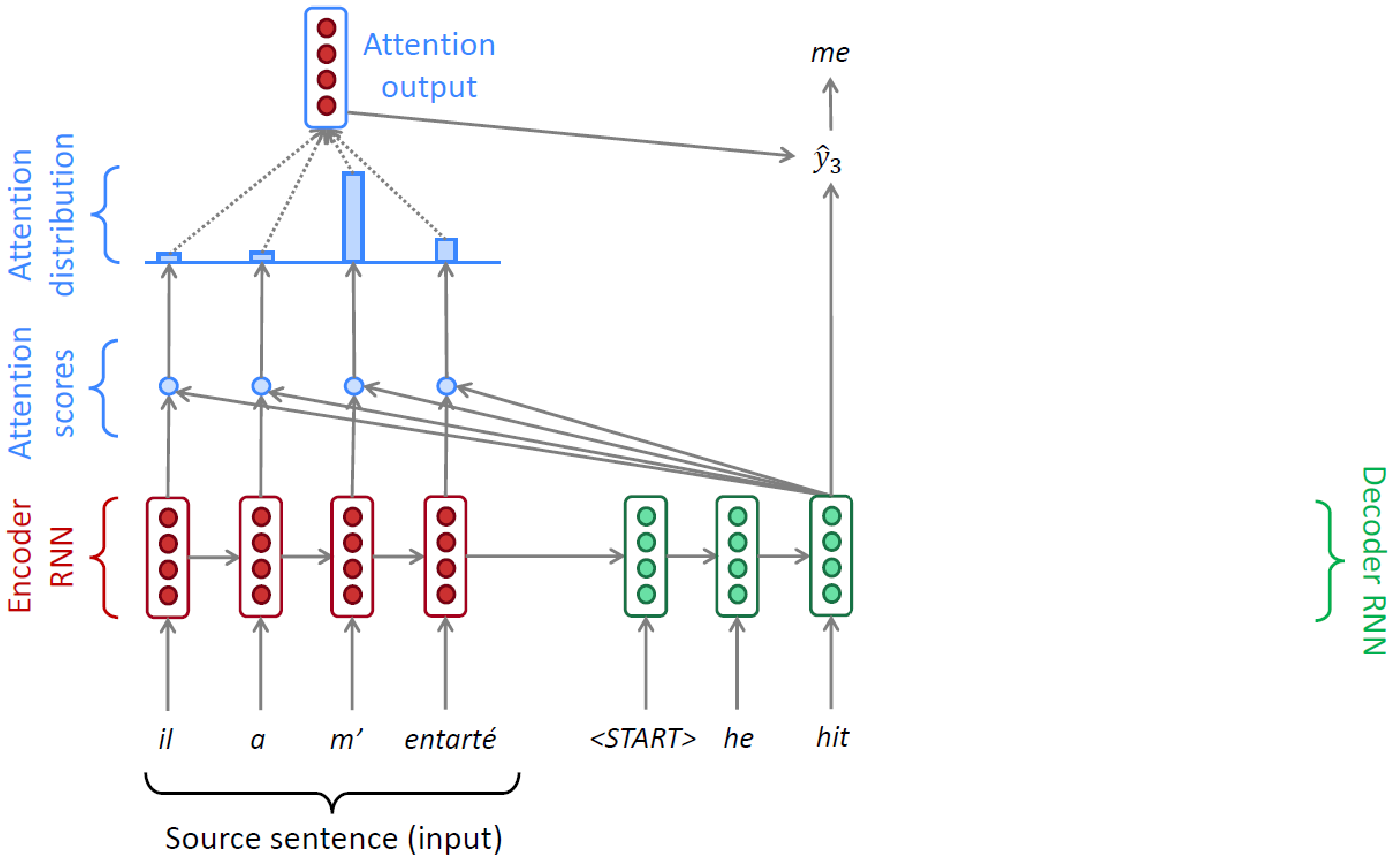
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Sequence-to-Sequence with Attention

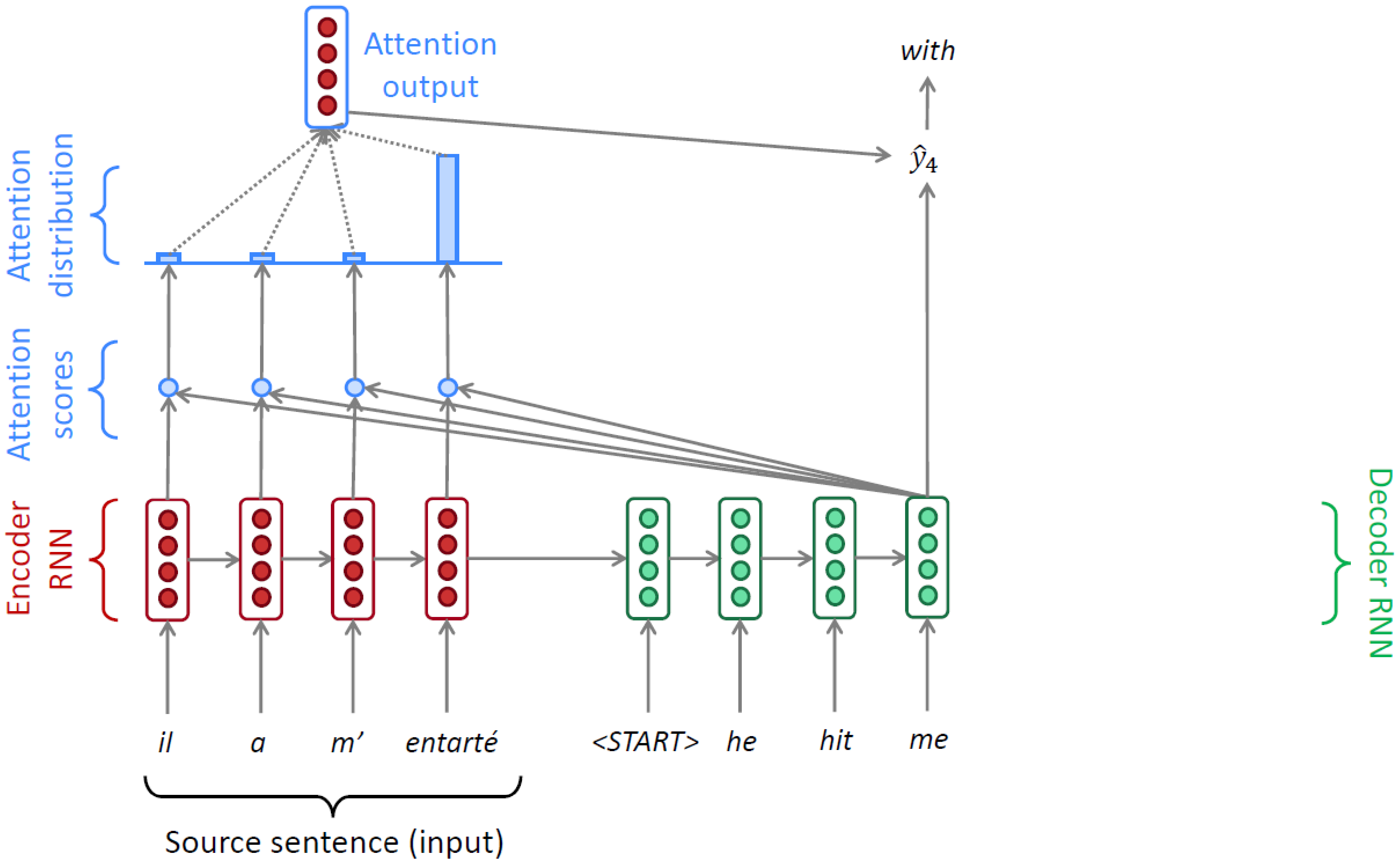


Sometimes we take the attention output from the previous step, and also feed it into the decoder (along with the usual decoder input).

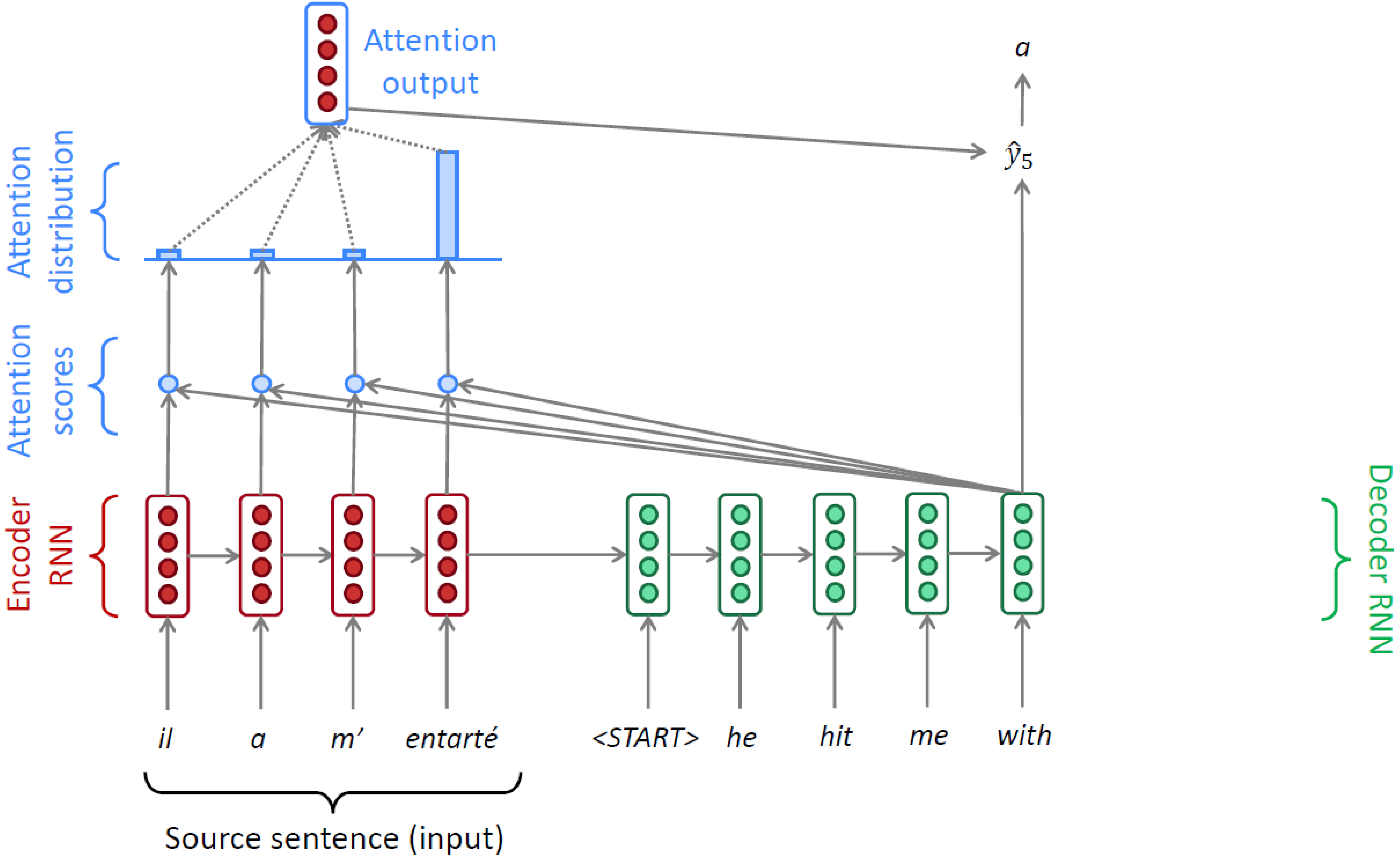
Sequence-to-Sequence with Attention



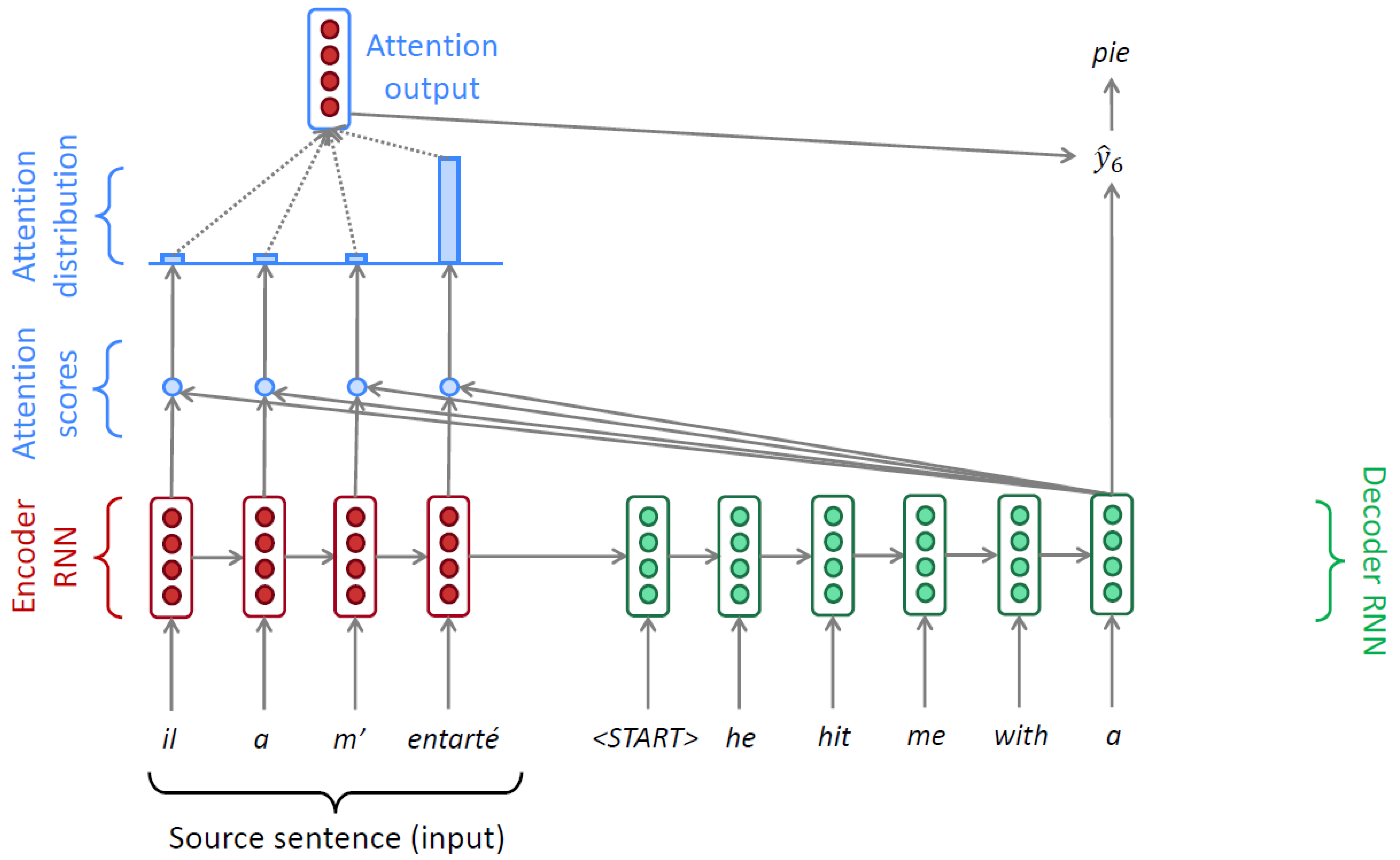
Sequence-to-Sequence with Attention



Sequence-to-Sequence with Attention



Sequence-to-Sequence with 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 $\boldsymbol{a}_t$

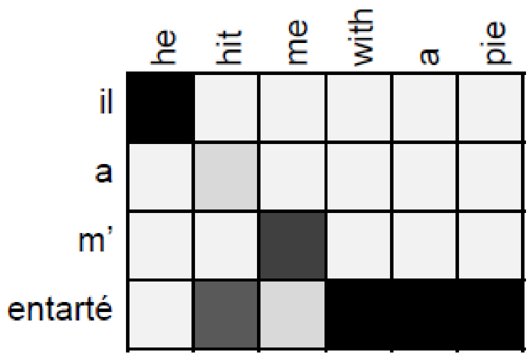
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- 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

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

Attention is Great

- Attention significantly improves NMT performance
  - It’s very useful to allow decoder to focus on certain parts of the source
- Attention 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

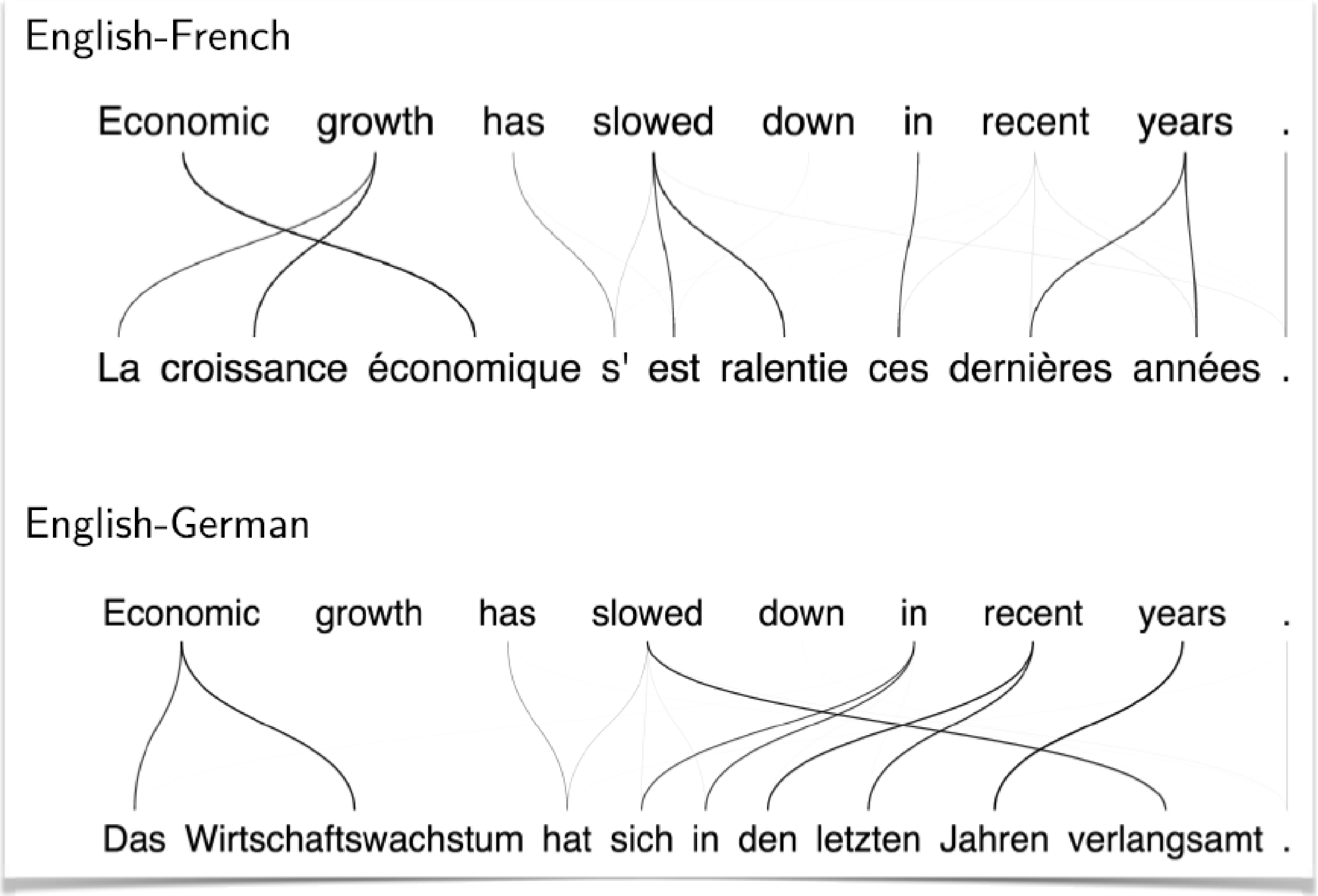


RNN Neural Machine Translation

- **Source**: *An admitting privilege is the right of a doctor to admit a patient to a hospital or a medical centre* <u>*to carry out a diagnosis or a procedure, based on his status as a health care worker at a hospital.*</u>
- **When collapsed:** *Un privilège d'admission est le droit d'un médecin de reconnaître un patient à l'hôpital ou un centre médical* <u>*d'un diagnostic ou de prendre un diagnostic en fonction de son état de santé.*</u>
- **RNNSearch**: *Un privilège d'admission est le droit d'un médecin d'admettre un patient à un hôpital ou un centre médical* <u>*pour effectuer un diagnostic ou une procédure, selon son statut de travailleur des soins de santé à l'hôpital.*</u>

RNN Neural Machine Translation

- Sensible alignment between source and target tokens
- Capture long-range reordering/dependencies
- Without strong supervision on the alignment
  - Weakly supervised learning



RNN Neural Machine Translation

- Input: **arbitrary as long as encoded into a set of continuous vectors**
- Output: a corresponding sentence in a target language

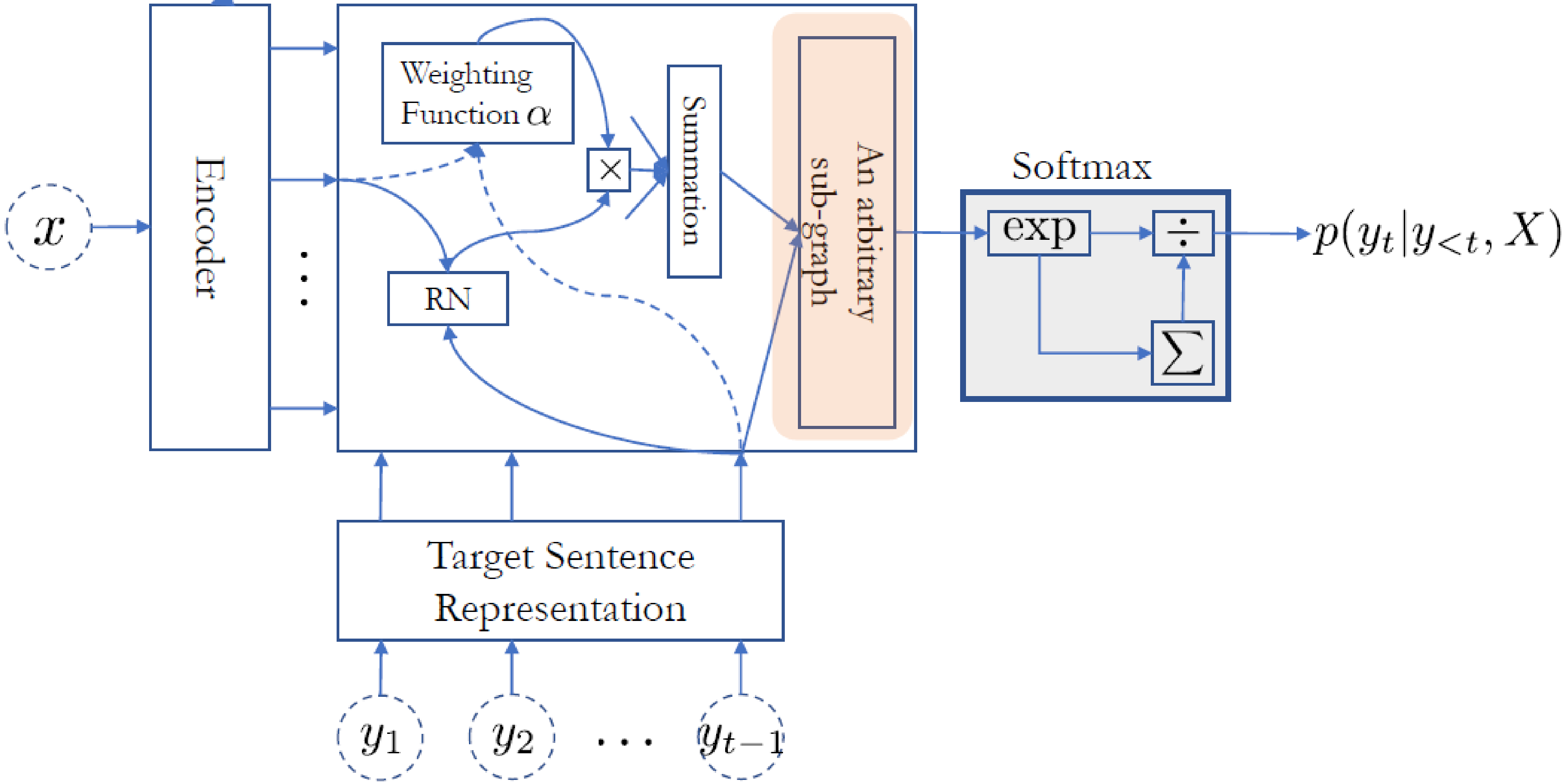


Image Caption Generation

- Input: an image
- Output: an image caption
- Network Architecture
  - Encoder: deep convolution network
  - Decoder: recurrent language model with the attention mechanism.
- Data: image-caption pairs

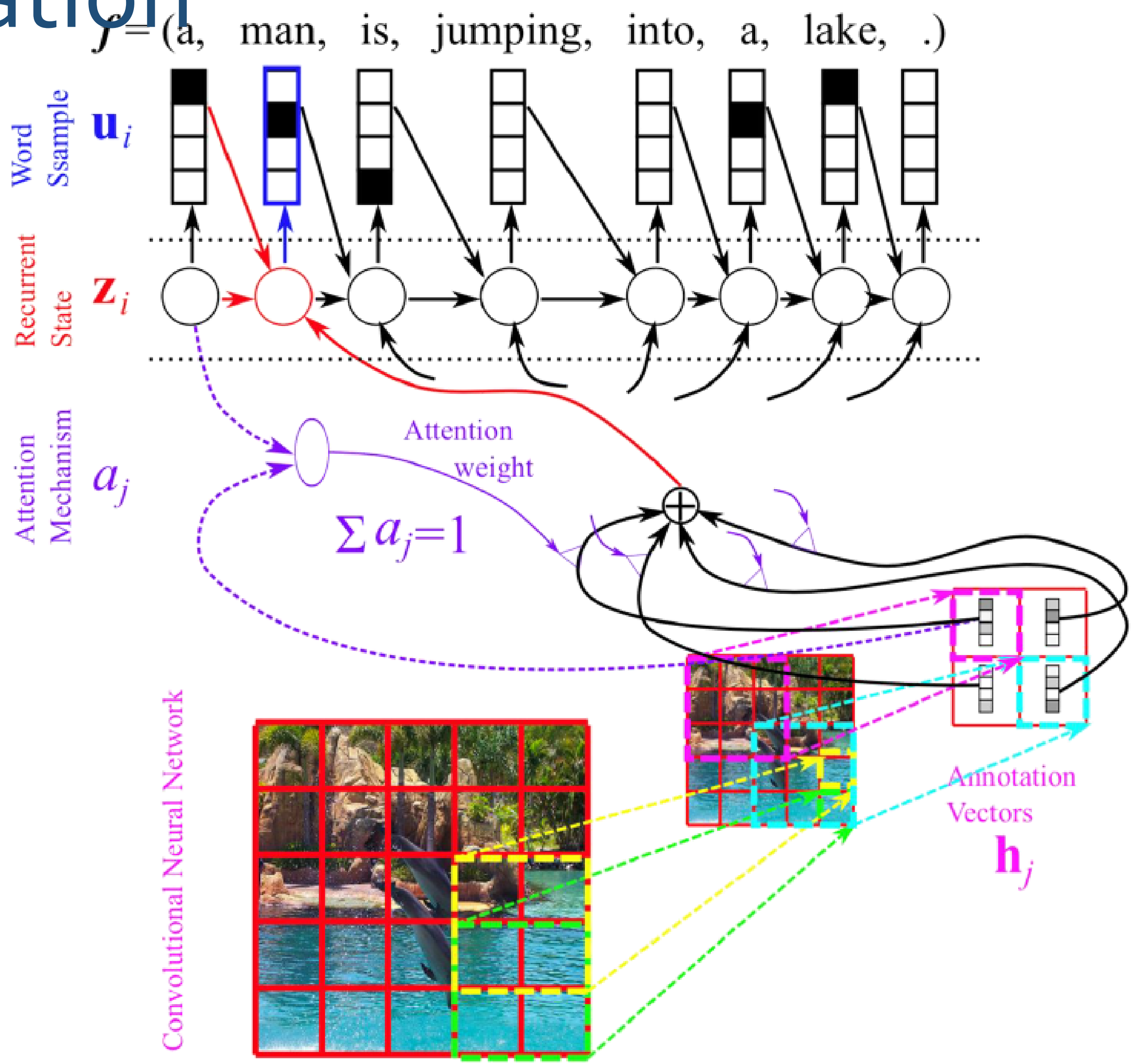


Image Caption Generation

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Speech Recognition

- Input: Speech
- Output: transcription