Announcements

- HW 5 due **Wednesday at 8pm**
  - Please start early!
- Quiz 8 due **Thursday at 8pm**
- Project Milestone 2 due **Wednesday, 11/15 at 8pm**

Lecture 18: NLP (Part 2)

CIS 4190/5190

Fall 2023

Recap

- **Classical approach:** Feature engineering + Standard ML model
- **Semi-Classical approach:** Word2Vec + Standard ML model
  - Sum embeddings of words to get passage features:

$$\phi(x) = \sum_{\text{word } i \in \text{document } x} \text{Embed}(i)$$

  - Still "bag-of-words" like model! ($\text{Embed}(i) = \text{OneHot}(i)$) is bag of words)

Words in Context

- While word2vec is trained based on context, after training, it is applied independently to each word
  - E.g., train linear regression of sum of word vectors, or n-grams
- **Why is this problematic?**
  - “He ate a tasty apple”
  - “He wrote his essay on his Apple computer”
- Both use the same embedding!

From Words to Documents

- Sentence2Vec, Paragraph2Vec scale these Word2Vec ideas to learn direct embeddings for sentences / paragraphs

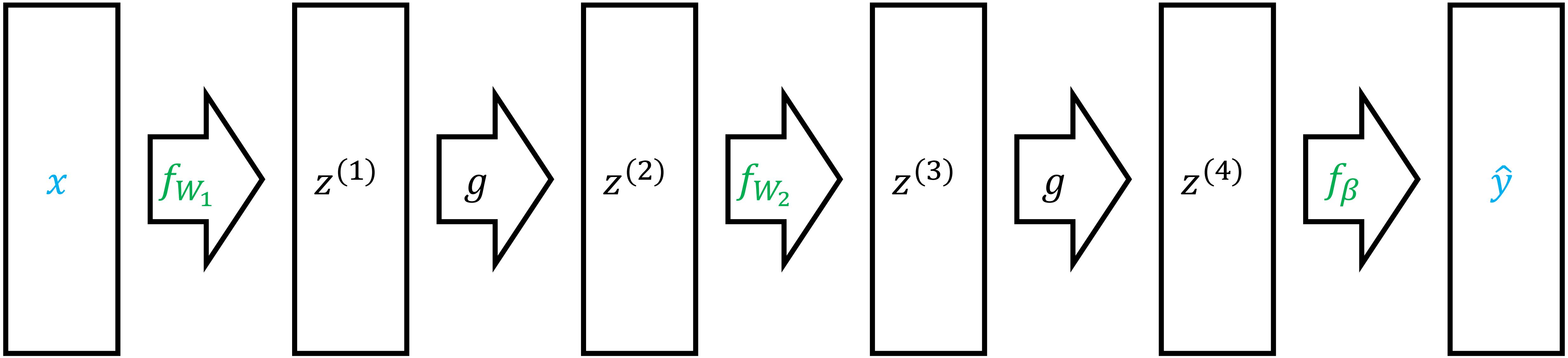
- However, much more common to treat as a sequence of words, and represent each word by its word2vec-style representation

- Sequence models have produced huge advances in NLP

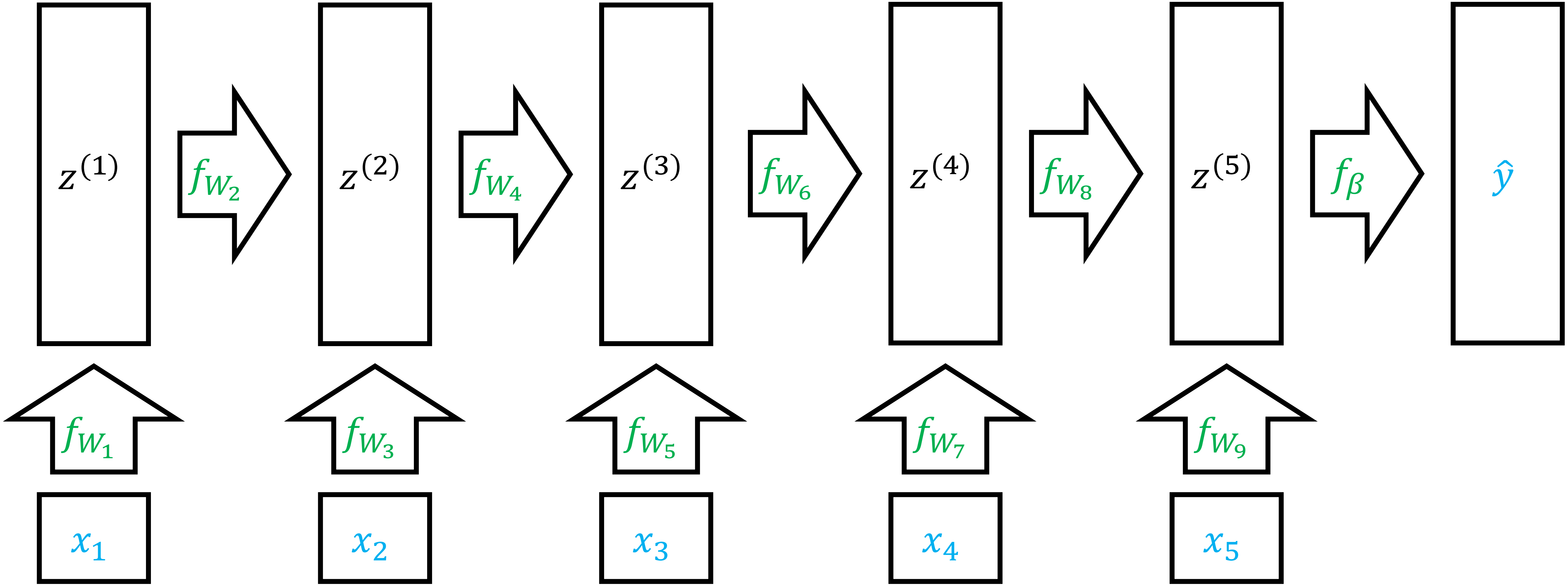
Recurrent Neural Networks

- Handle inputs/outputs that are **sequences**
- **Naïve strategy**
  - Pad inputs to fixed length and use feedforward network
  - **Ignores temporal structure**
- **Recurrent neural networks (RNNs):** Process input sequentially

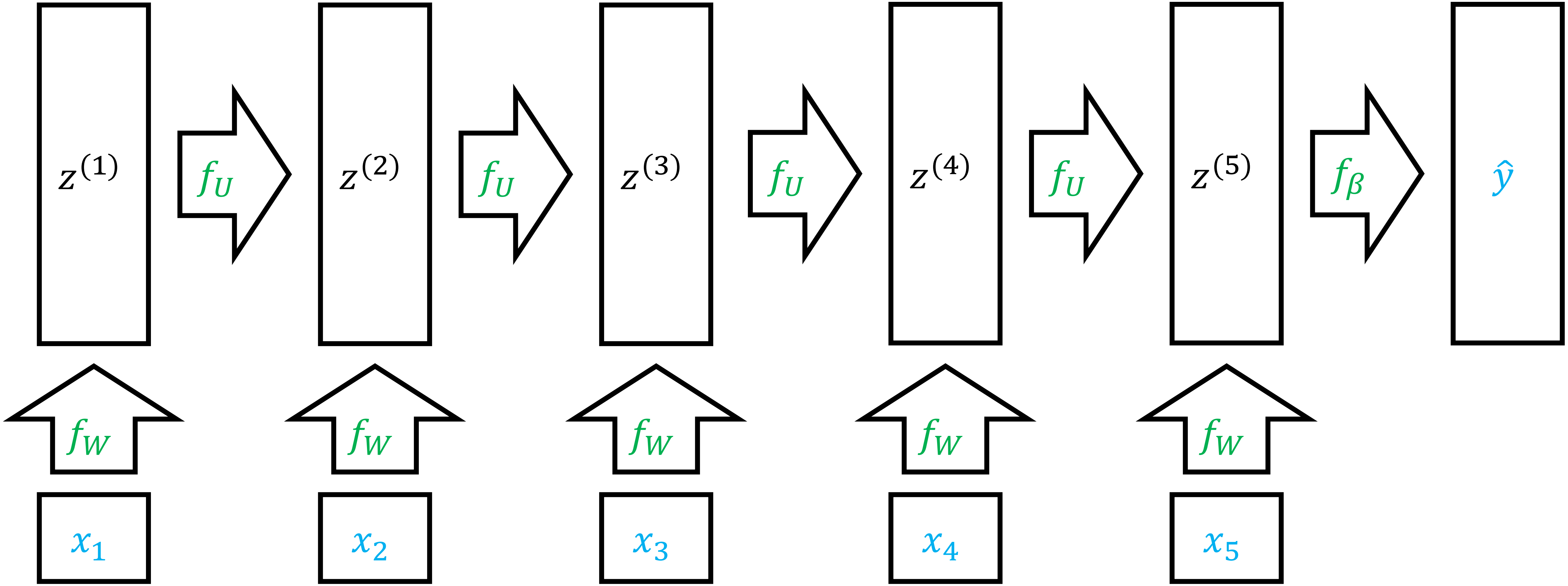
Feedforward Neural Networks



Recurrent Neural Networks



Recurrent Neural Networks



Recurrent Neural Networks

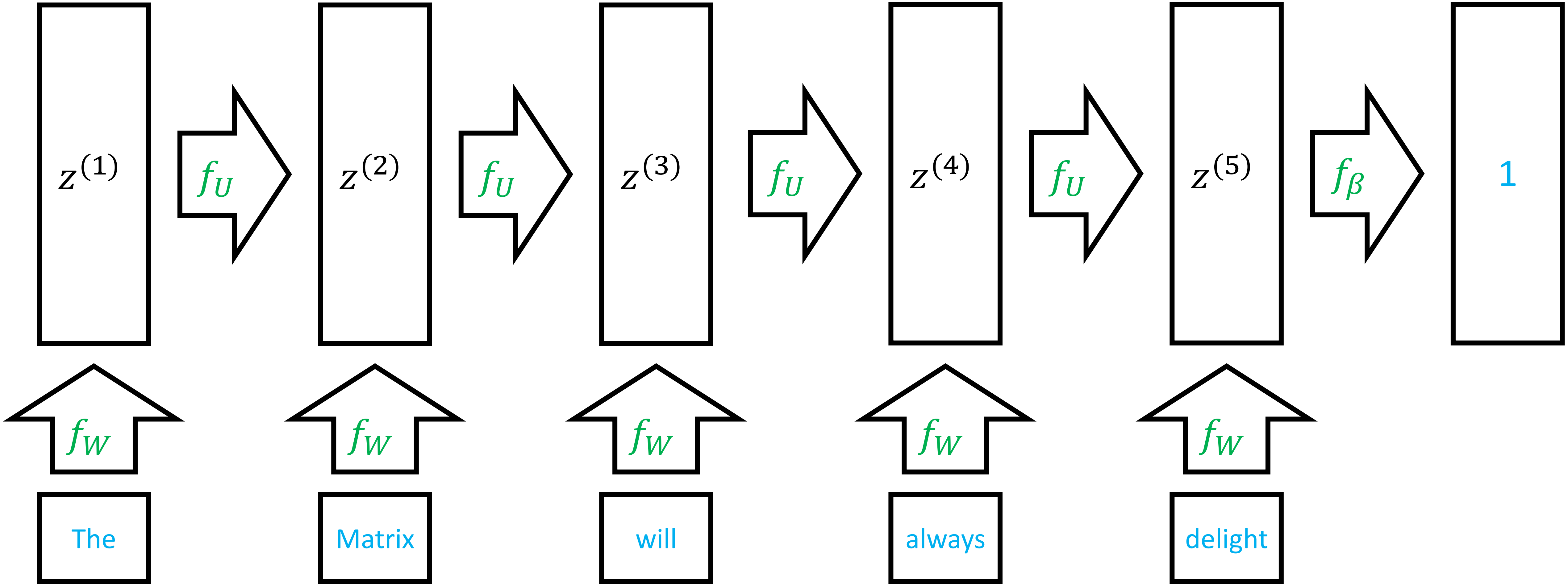
- Initialize $z^{(0)} = \vec{0}$
- Iteratively compute (for $t \in \{1, \dots, \mathrm{T}\}$):

$$z^{(t)} = g\left(Wx_t + Uz^{(t-1)}\right)$$

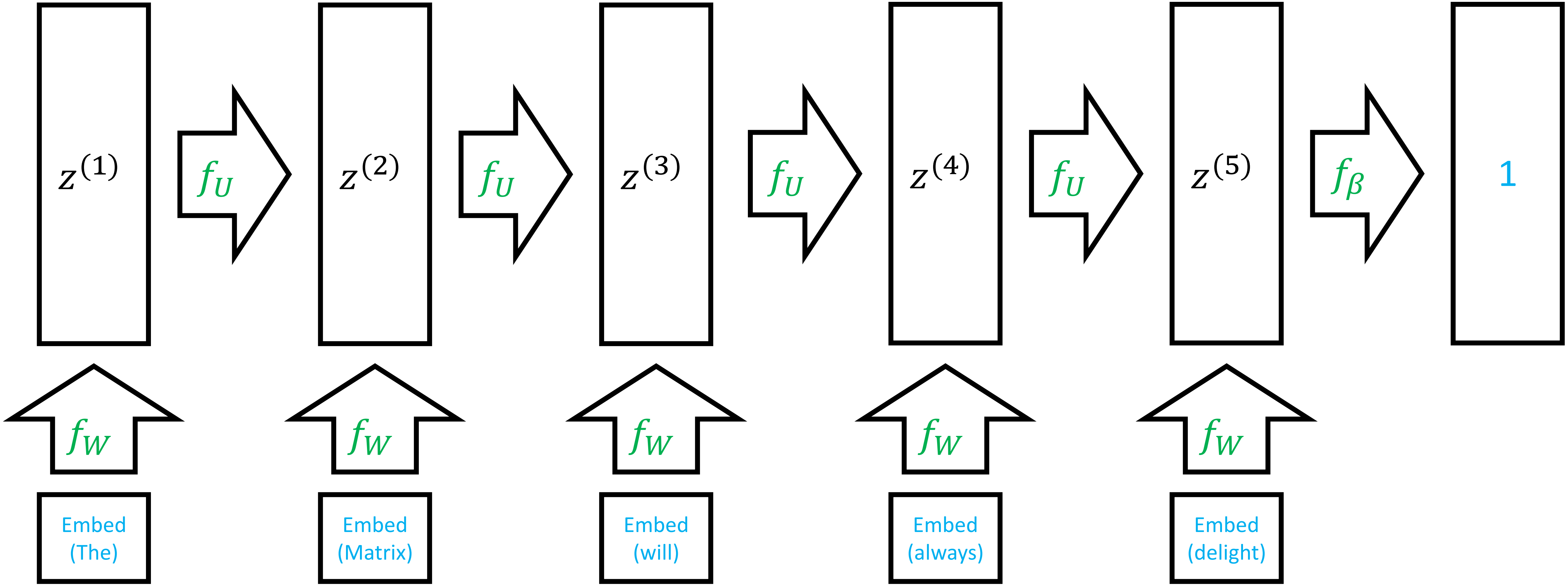
- Compute output:

$$y = \beta^\top z^{(T)}$$

Sentiment Classification



Sentiment Classification



Recurrent Neural Networks

- Initialize $z^{(0)} = \vec{0}$
- Iteratively compute (for $t \in \{1, \dots, \mathrm{T}\}$):

$$z^{(t)} = g\left(W\ \mathrm{Embed}(x_t) + U z^{(t-1)}\right)$$

- Compute output:

$$y = \beta^\top z^{(T)}$$

Recurrent Neural Networks

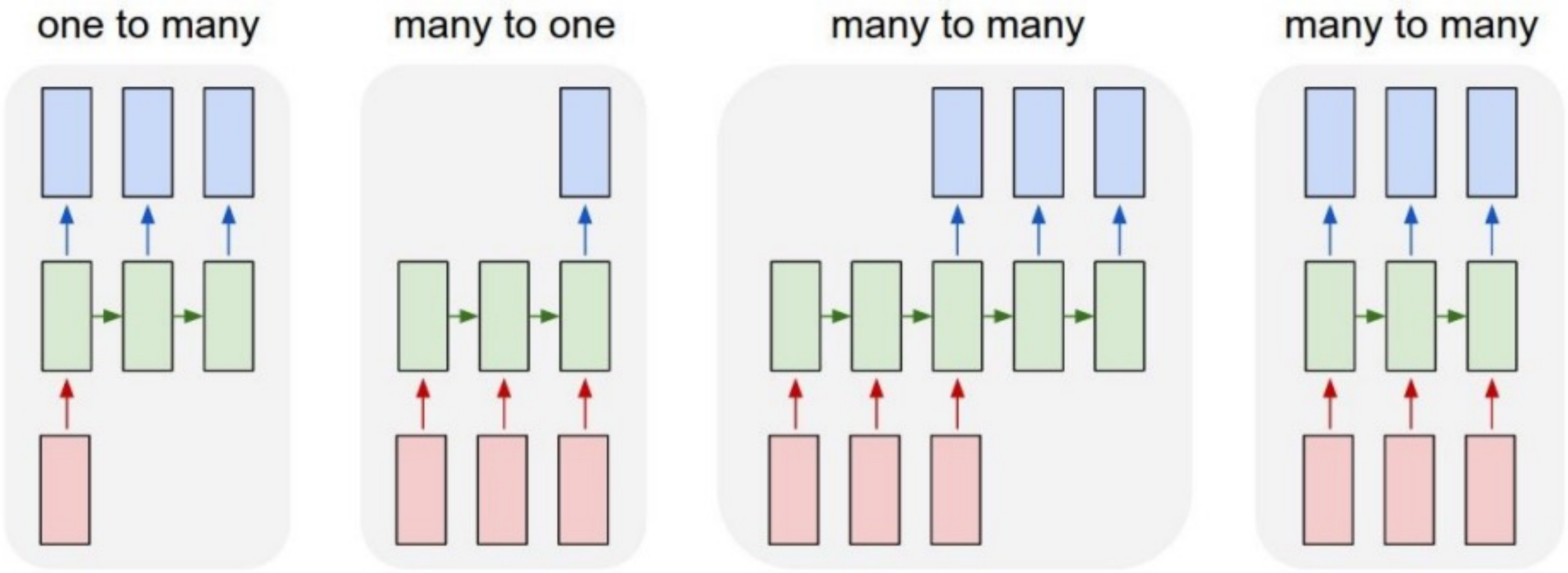


Image captioning

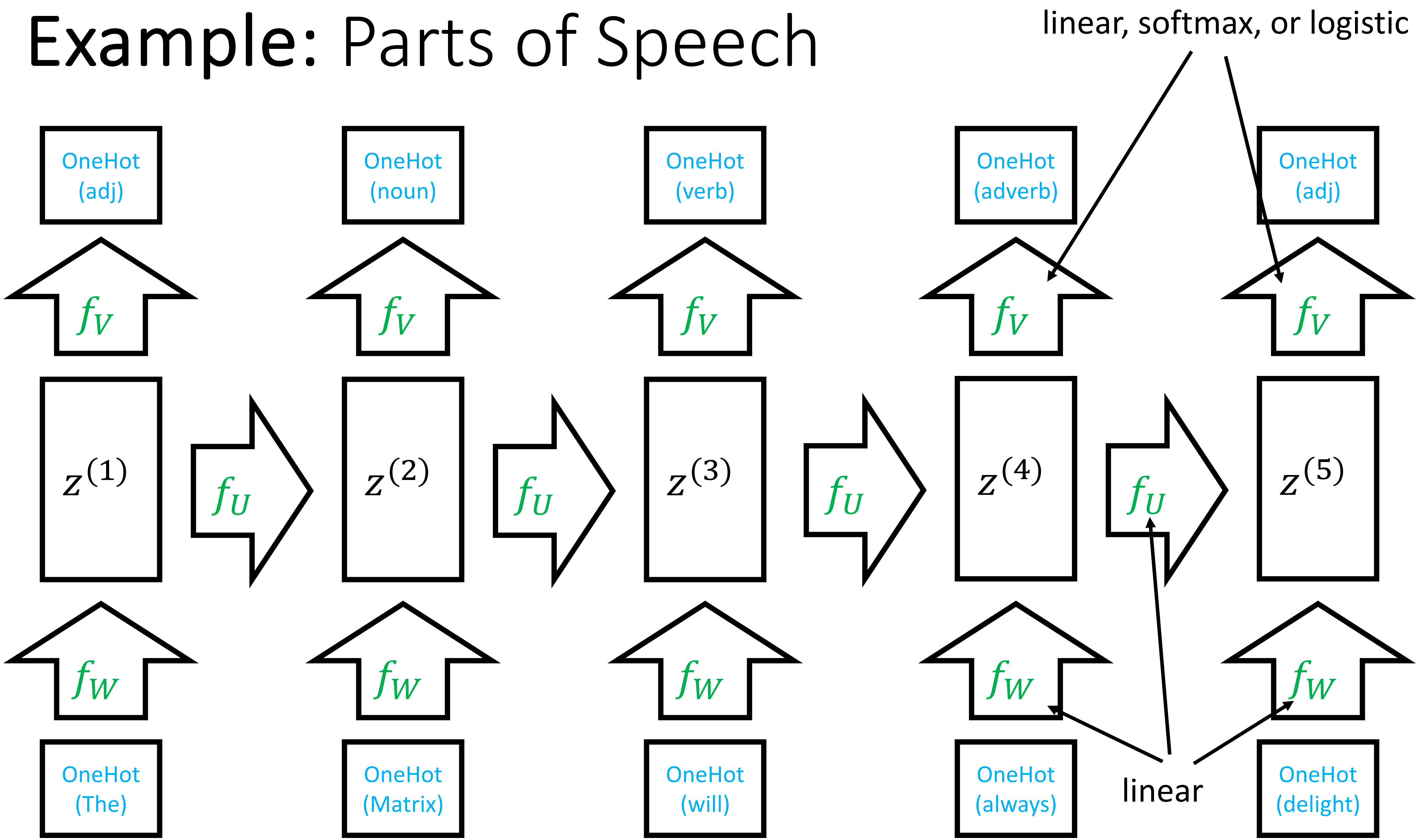
Sentiment prediction

Machine translation

Video captioning

Fei-Fei Li, Justin Johnson, Serena Yeung

Example: Parts of Speech



Training RNNs

- Backpropagation works as before
  - For shared parameters, we can show that the overall gradient is sum of gradient at each usage

- Exploding/vanishing gradients can be particularly problematic

- LSTM (“long short-term memory”) and GRU (“gated recurrent unit”) do clever things to better maintain hidden state

Training RNNs

$$z_1 = g(Wx_1 + Uz_0)$$
$$z_2 = g(Wx_2 + Uz_1)$$
$$z_3 = g(Wx_3 + Uz_2)$$

$$\frac{\partial L}{\partial U} = \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial U} + \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial U} + \frac{\partial L}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial U}$$

Local Contribution Historical Contribution

Pretraining RNNs

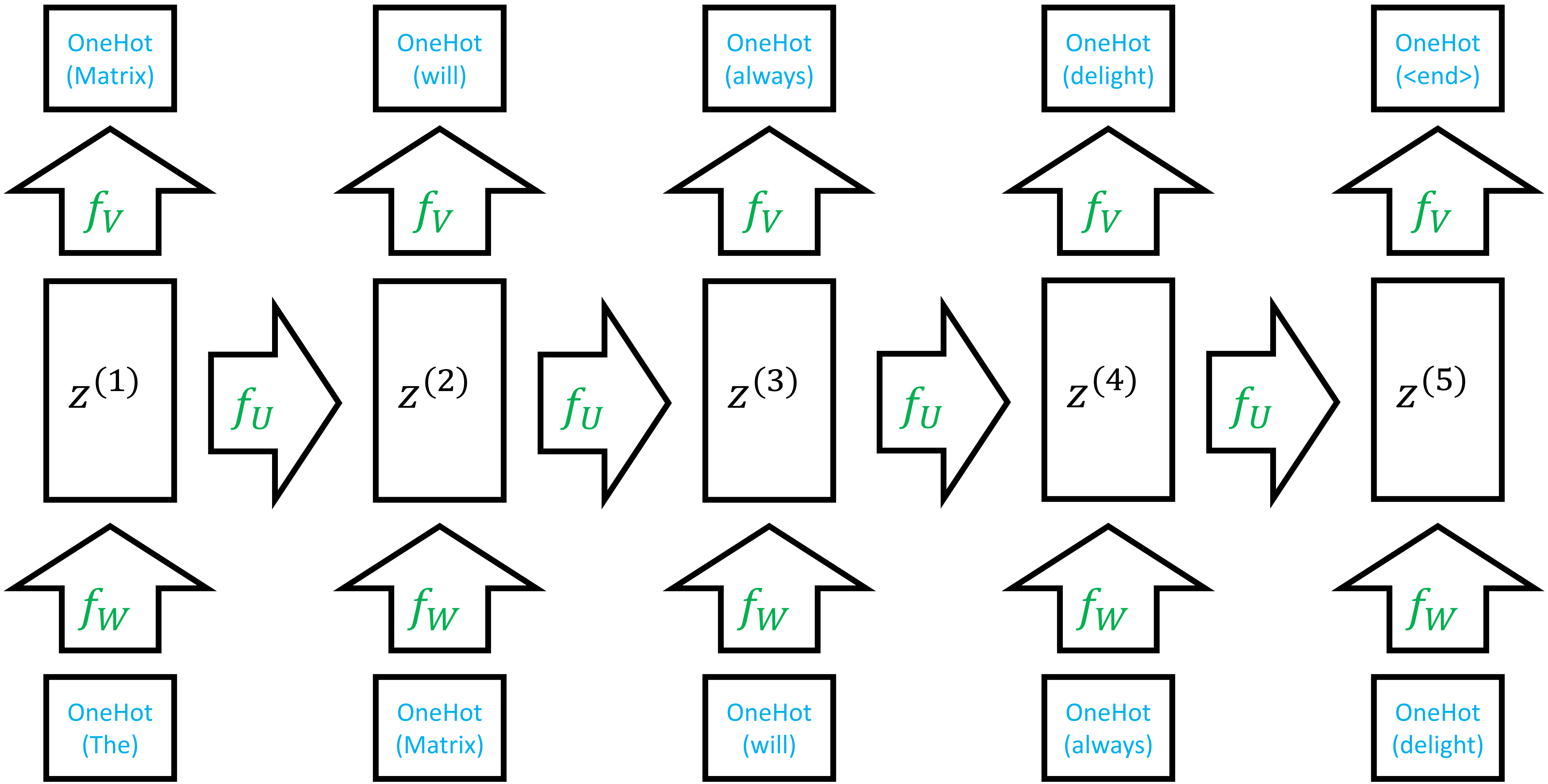
- **Unsupervised pretraining**
  - Train on dataset of text to predict next word (classification problem)
  - $x = w_1 w_2 \dots w_t$ and $y = w_{t+1}$ (usually y is one-hot even if x is not)

- Finetune pretrained RNN on downstream task

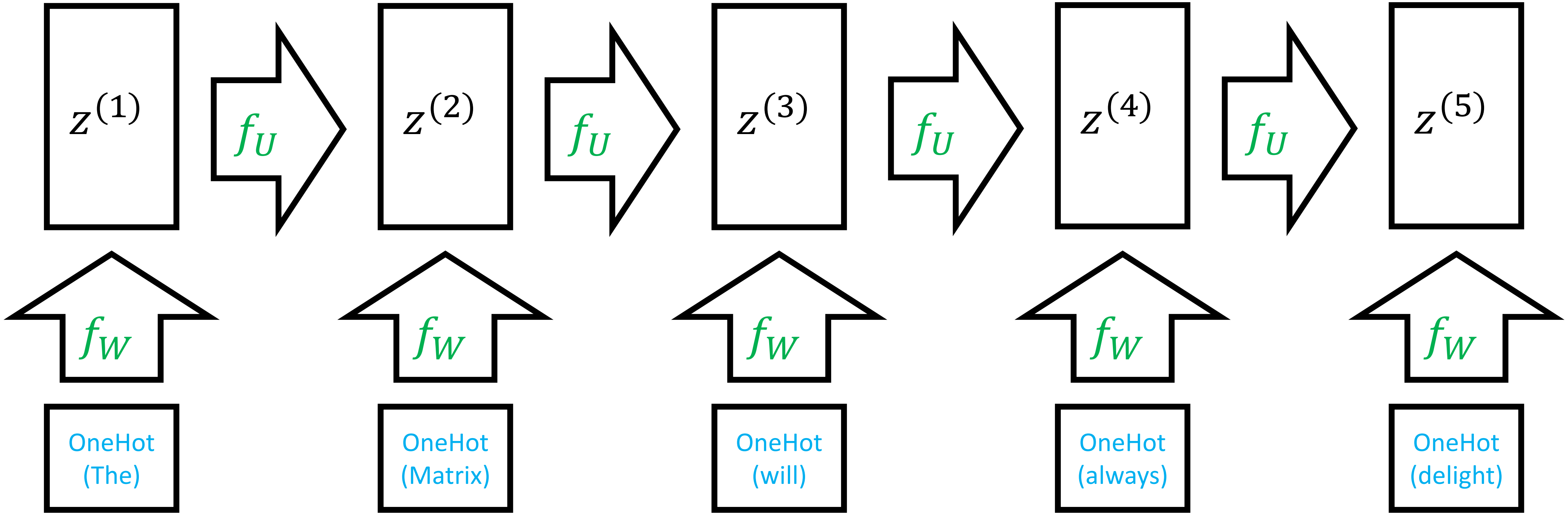
Pretraining RNNs

- **Step 0:** Pretrained on a large **unlabeled** text dataset
  - Also called “self-supervised”
  - Trained using supervised learning, but labels are predicting data itself
- **Step 1:** Replace next-word prediction layer with new layer for task
- **Step 2:** Train new layer or finetune end-to-end
  - Can think of last layer of pretrained RNN as a “contextual word embedding”

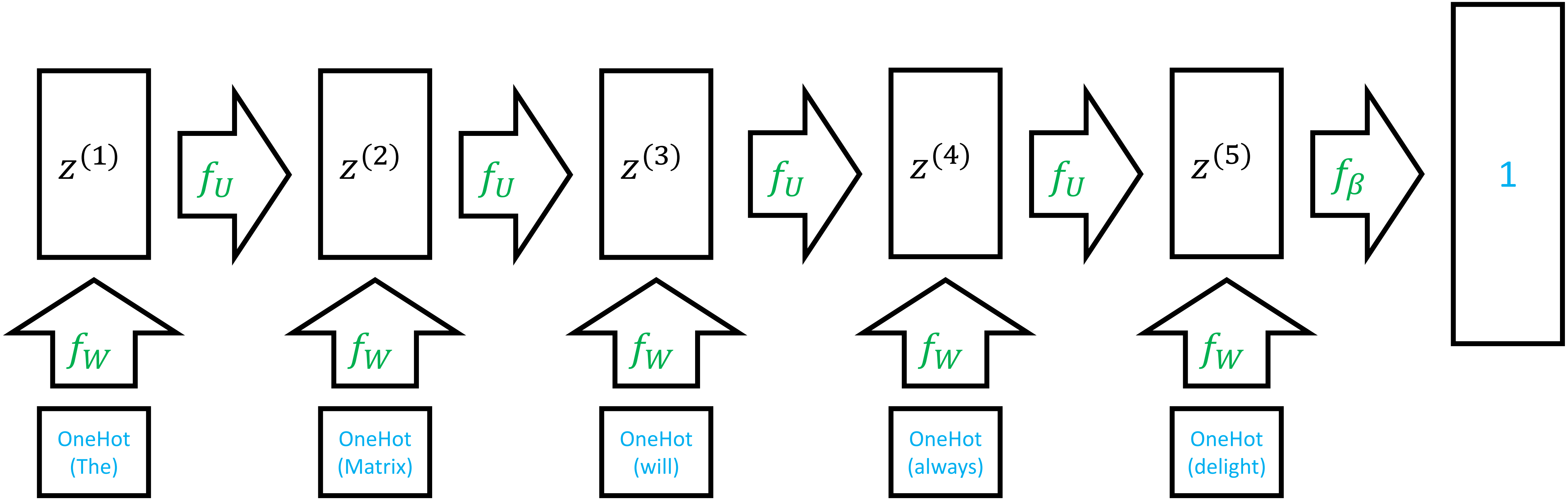
Pretraining RNNs



Pretraining RNNs



Pretraining RNNs

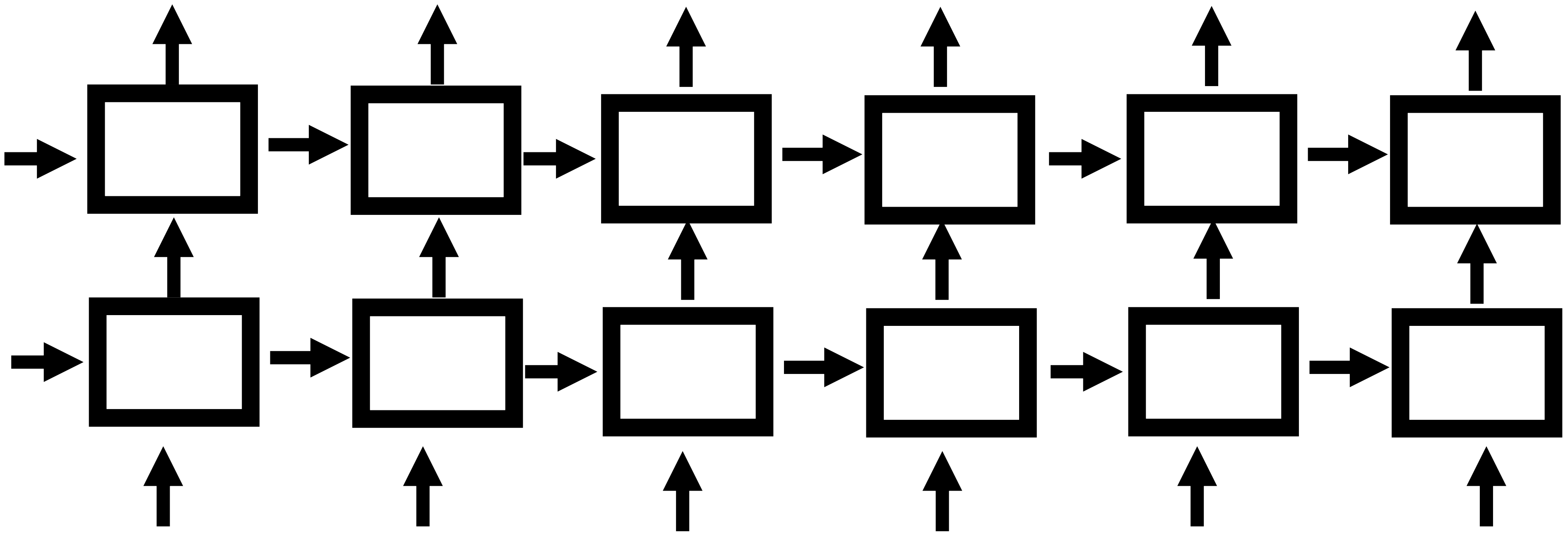


Shortcomings of RNNs

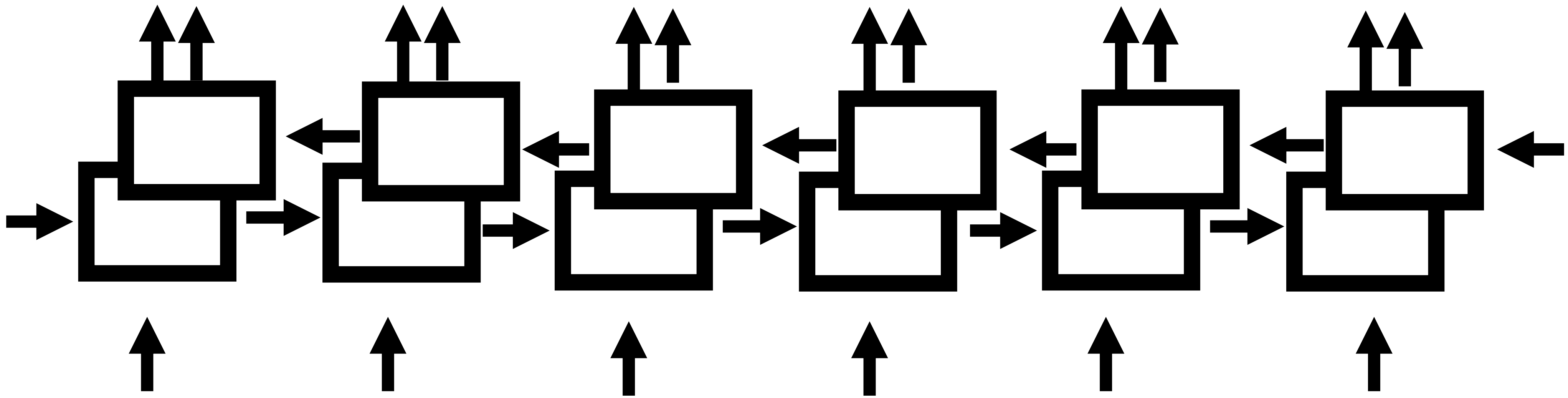
- **Shortcomings**
  - Unidirectional information flow (must remember everything relevant)
  - Need to remember everything until it is needed

- **Improvements/alternatives**
  - Stacked/Bidirectional models
  - LSTMs/GRUs
  - CNNs
  - Transformers

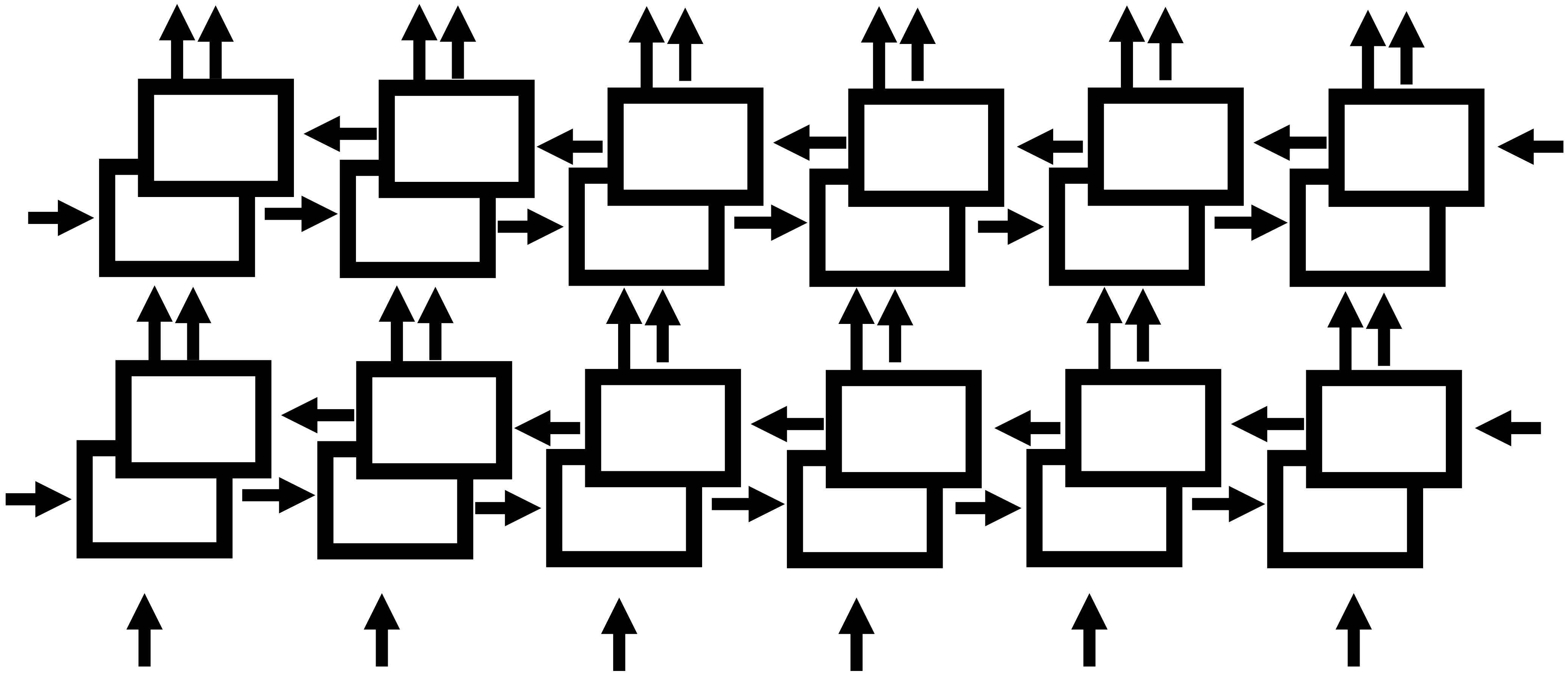
Stacked RNN



Bidirectional RNN

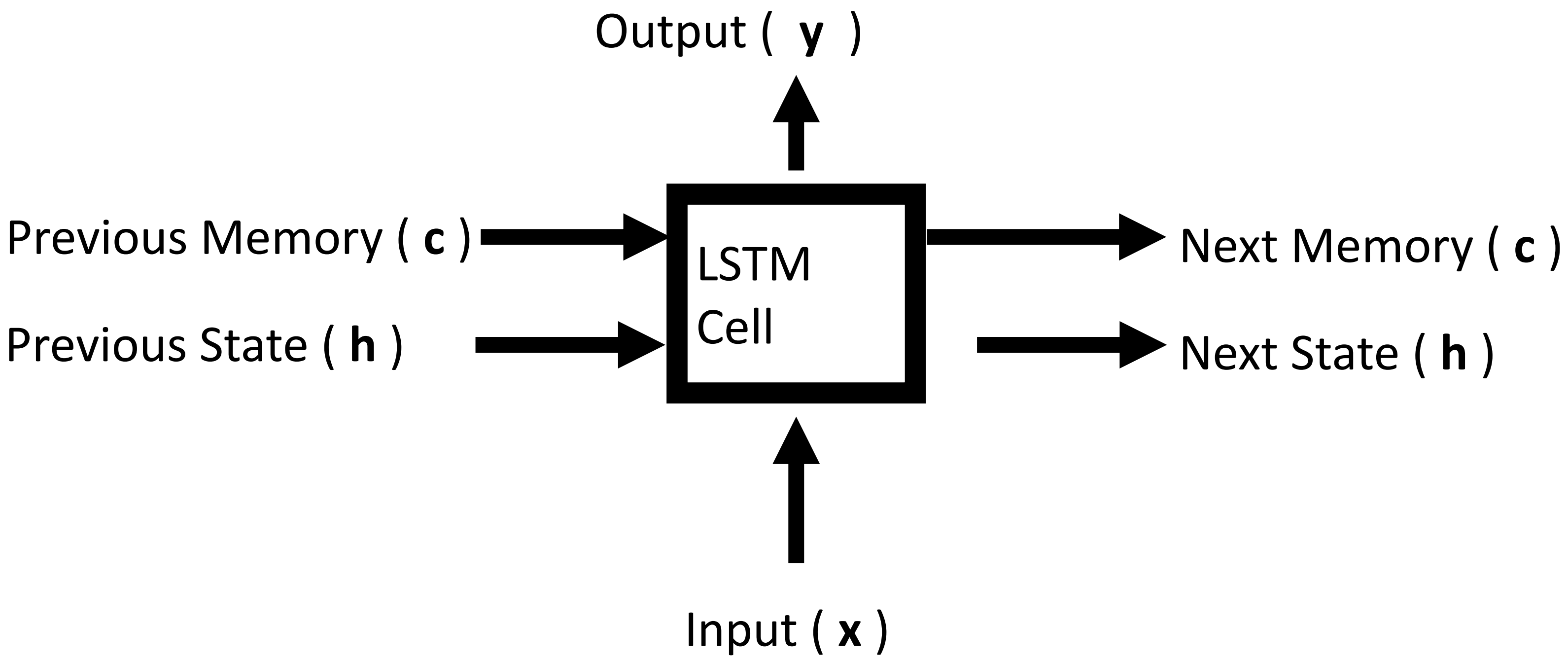


Stacked + Bidirectional RNN



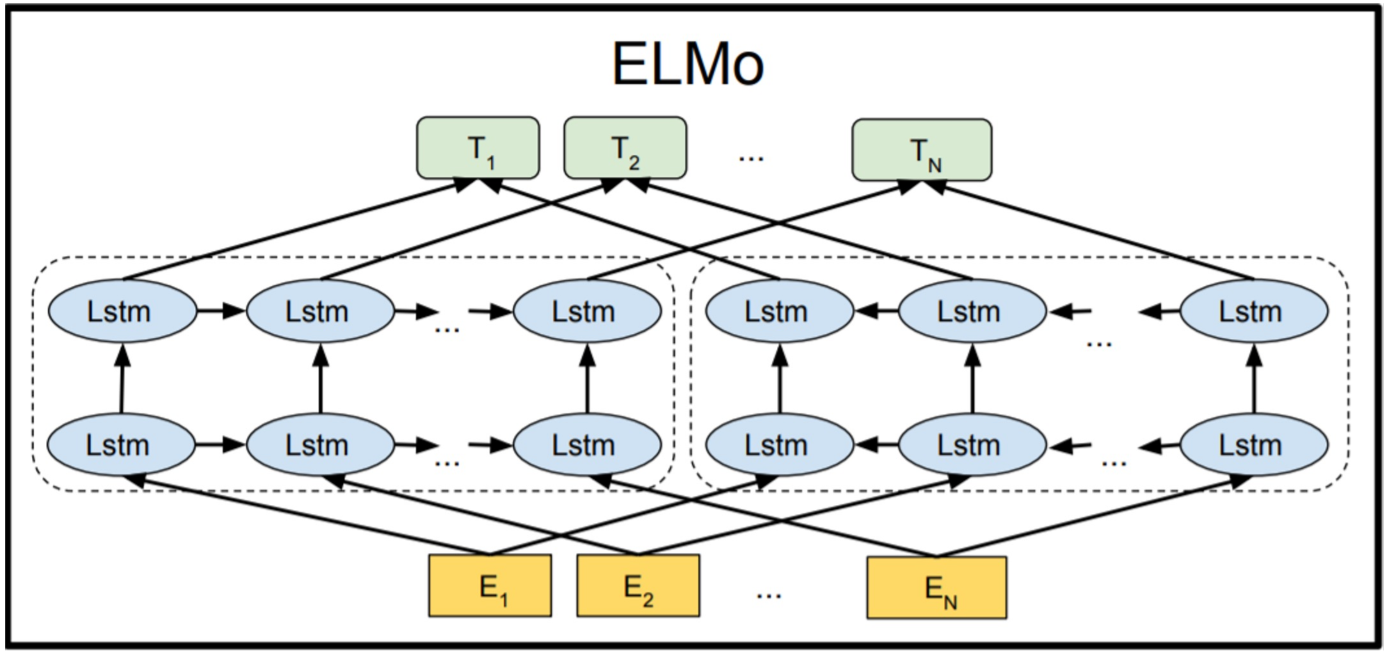
Long Short Term Memory

- **Goal:** Replace some multiplicative relationships in hidden state with additive relationships



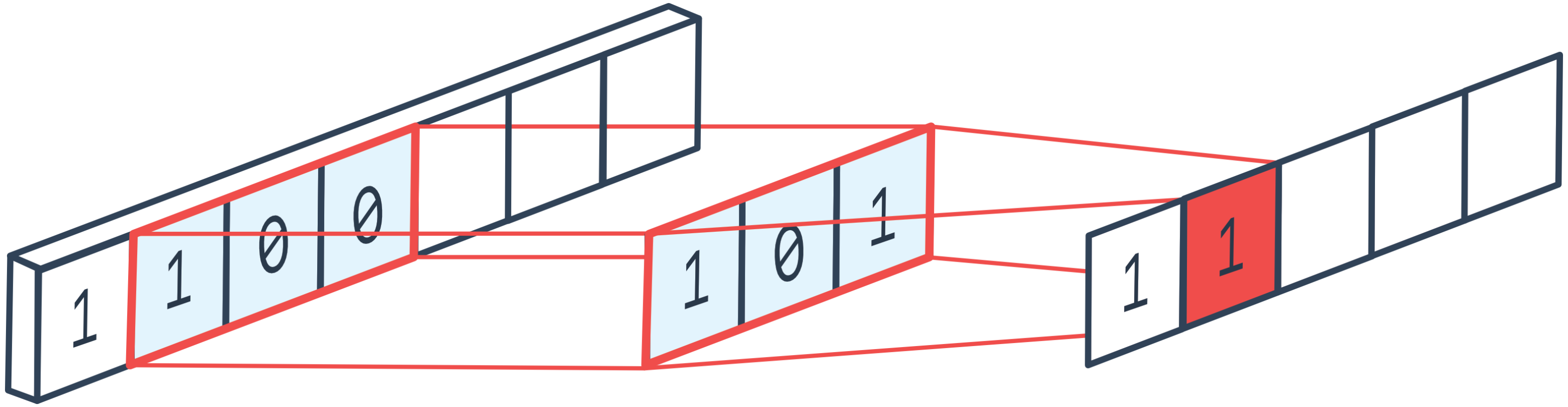
ELMo Word Embeddings

- **Bidirectional LSTM:** Combine one LSTM to predict next word given previous words, another to predict previous word given later words



CNNs

- **Model**
  - 1D convolutional layers
  - Input is word embedding sequence
  - # channels is word embedding dimension



CNNs

- **Shortcomings**
  - Hard to reason about interactions between words that are far apart

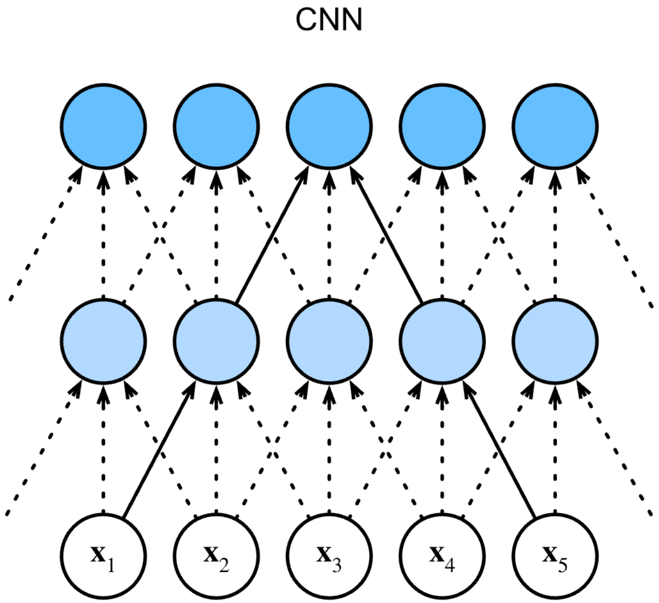
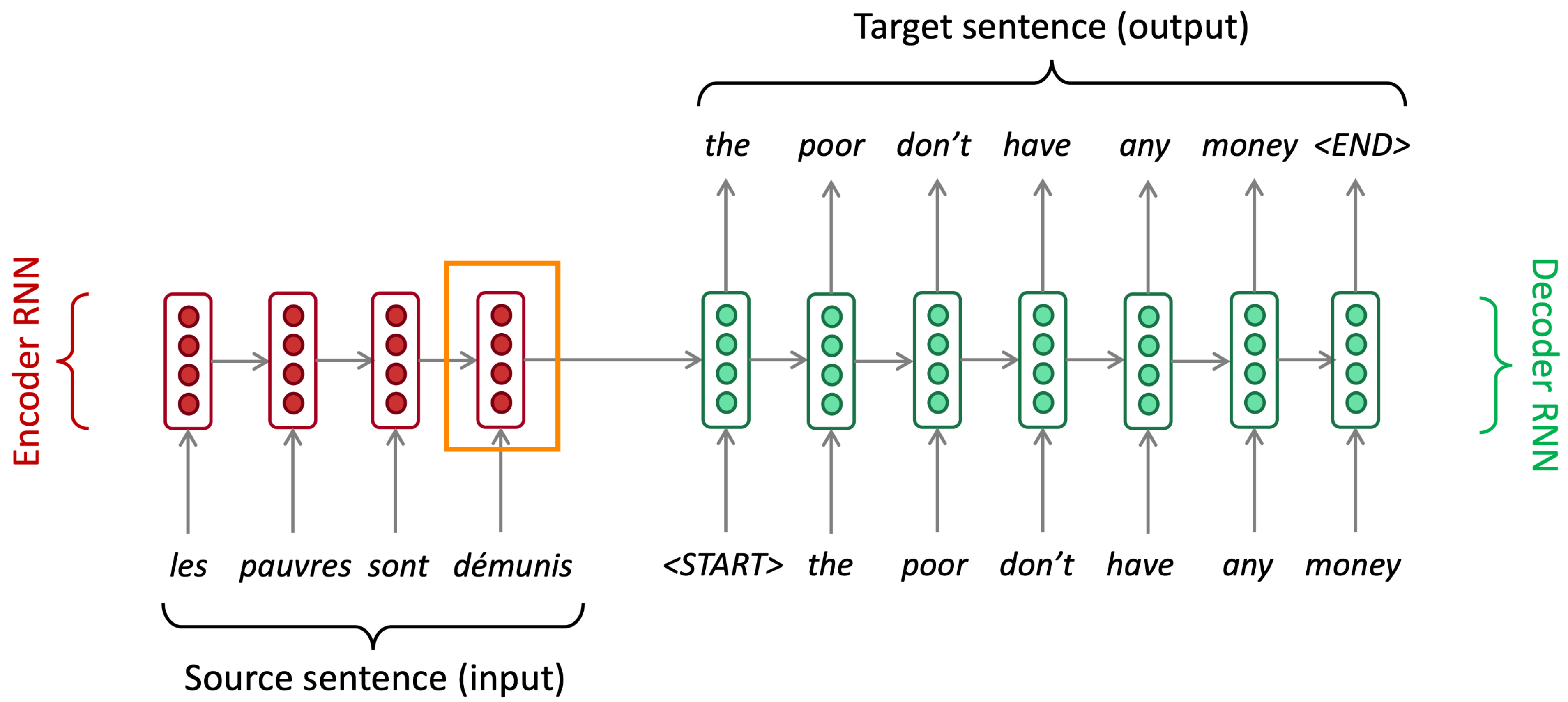


Figure credit to d2l.ai

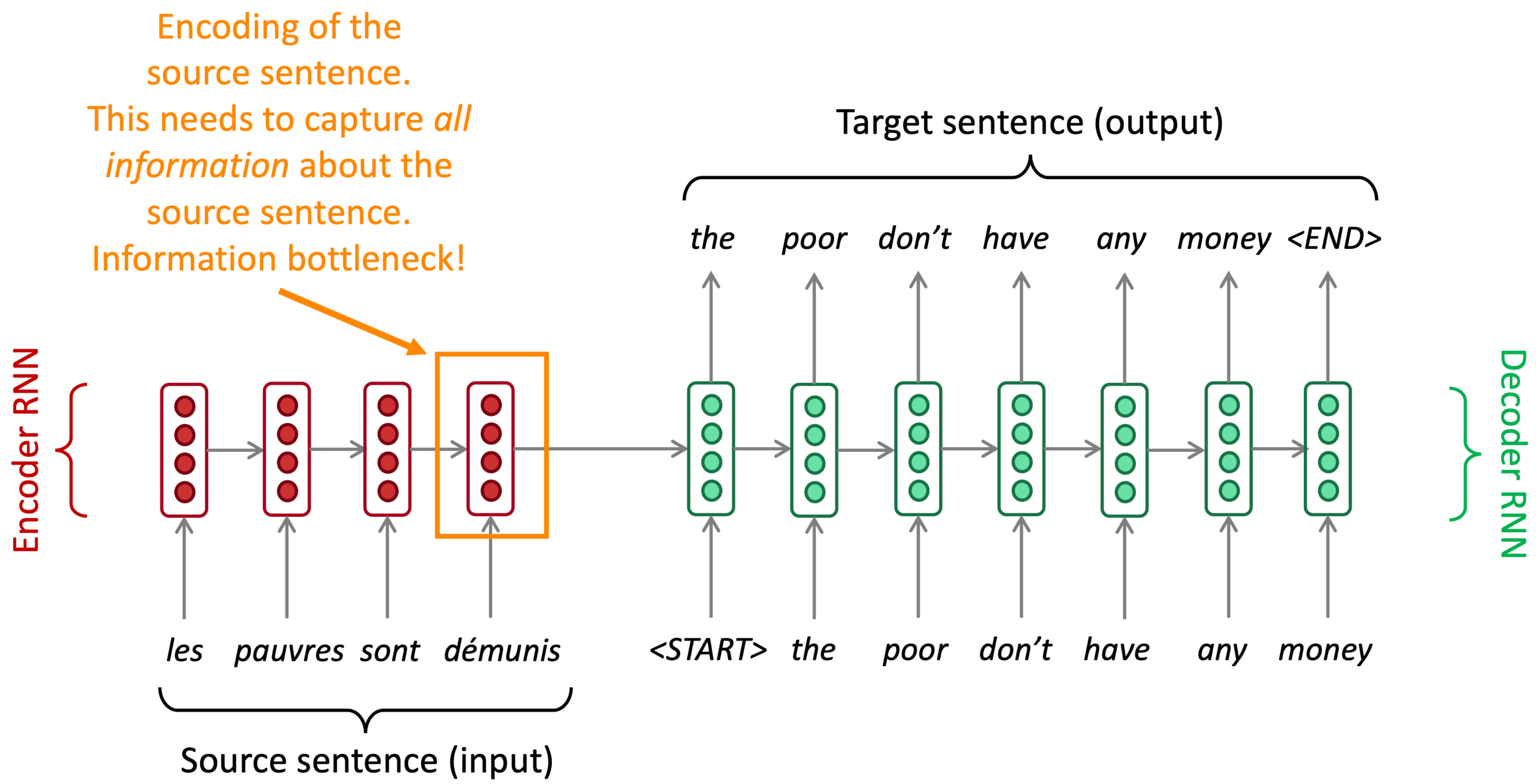
Attention

- RNNs have trouble propagating information forwards
- **Solution:** Let RNN “pay attention” to small part of past sequence

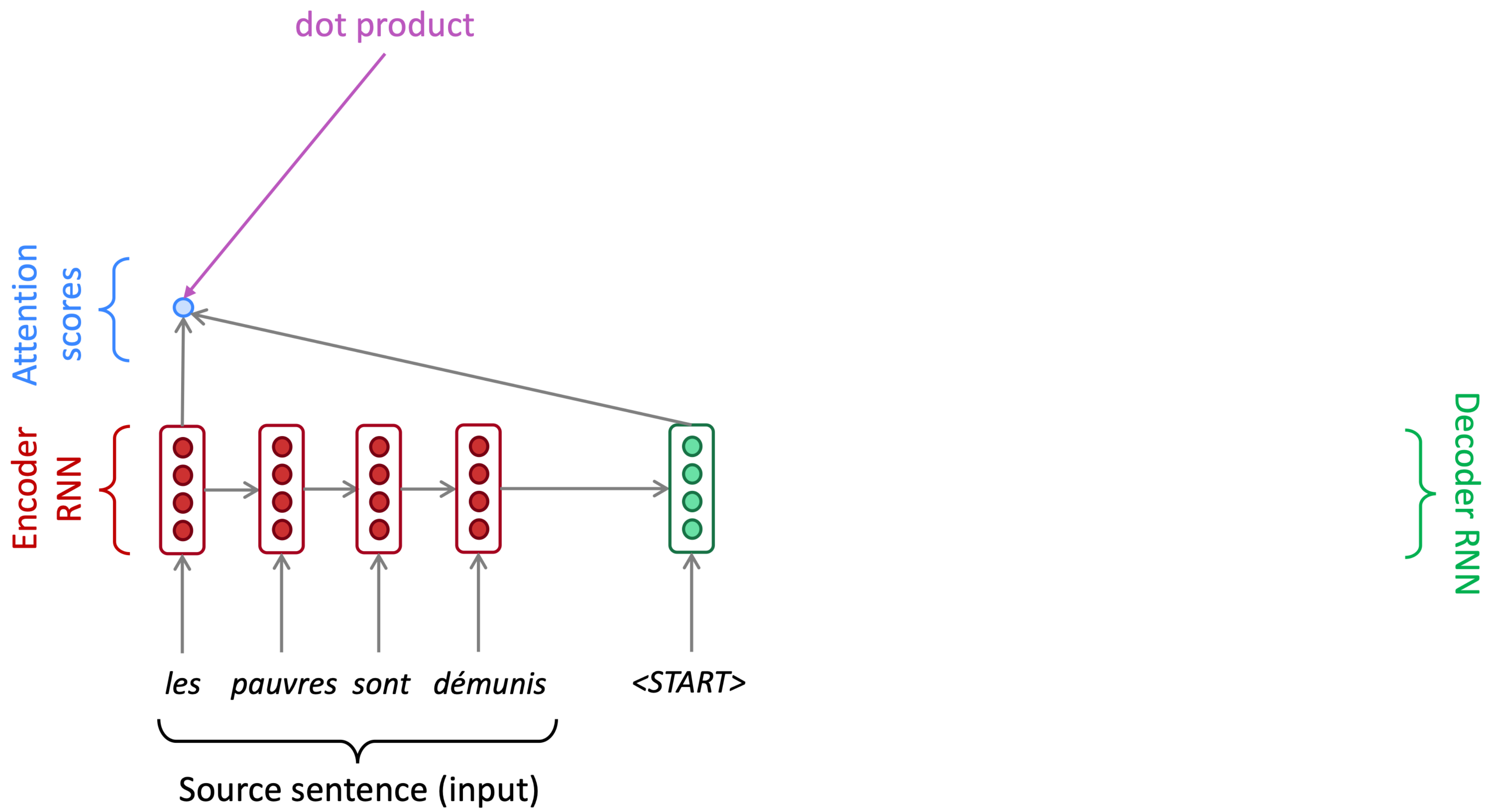
Example: Machine Translation



Example: Machine Translation

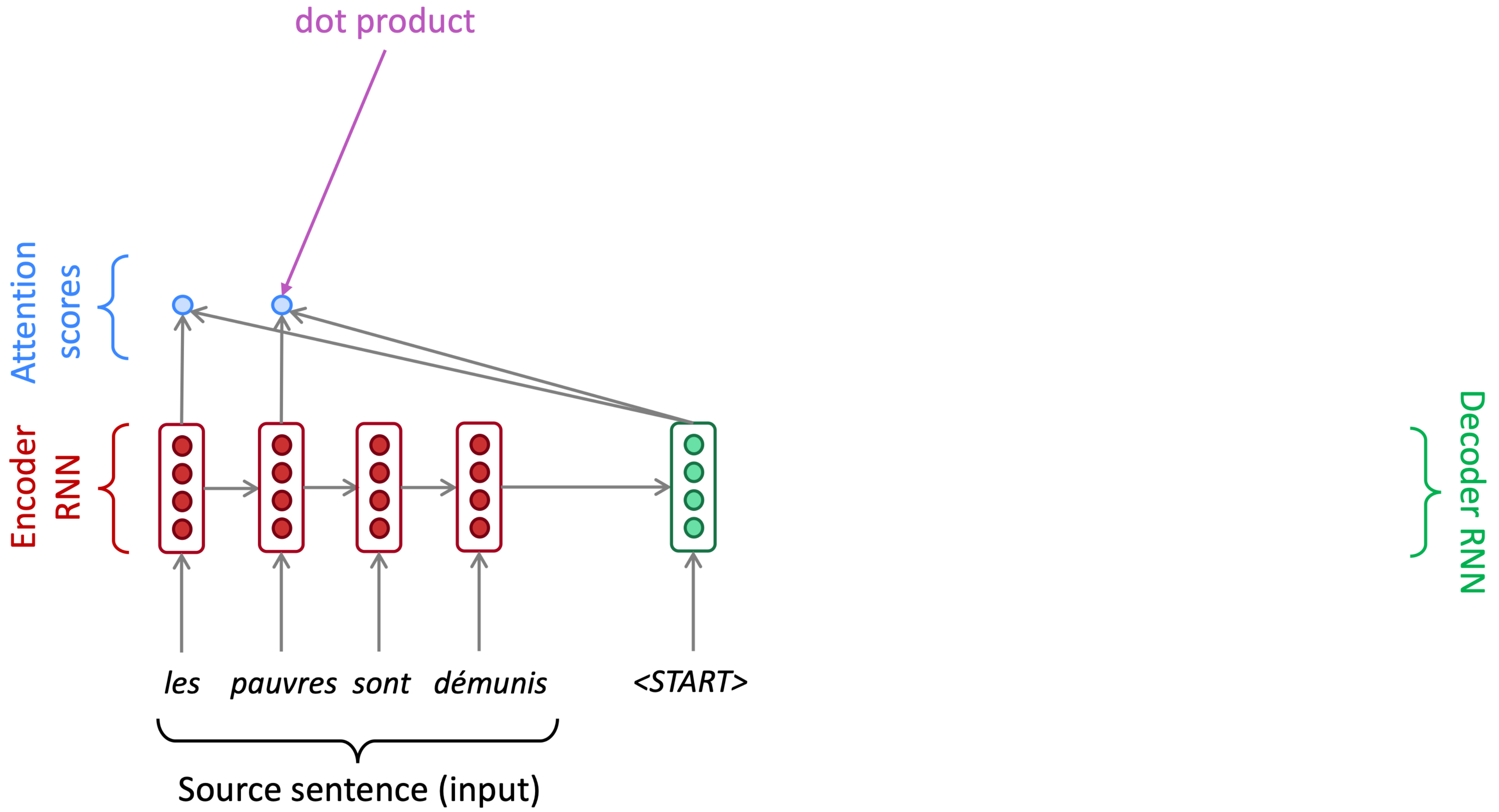


Attention

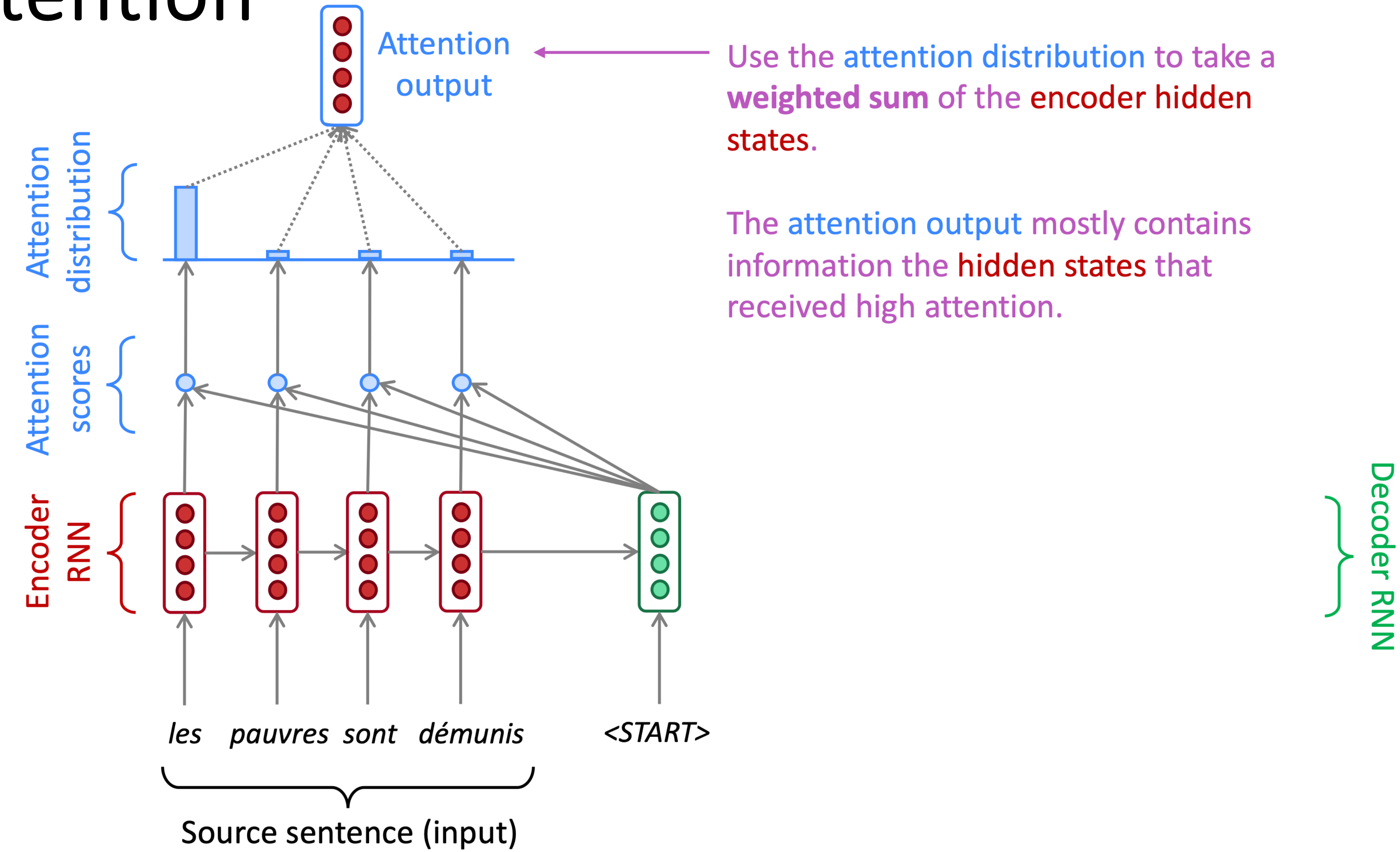


dot product
Attention scores
Encoder RNN
Decoder RNN
les *pauvres* *sont* *démunis*
<START>
Source sentence (input)

Attention


dot product
Attention scores
Encoder RNN
Decoder RNN
les
pauvres
sont
démunis
<START>
Source sentence (input)

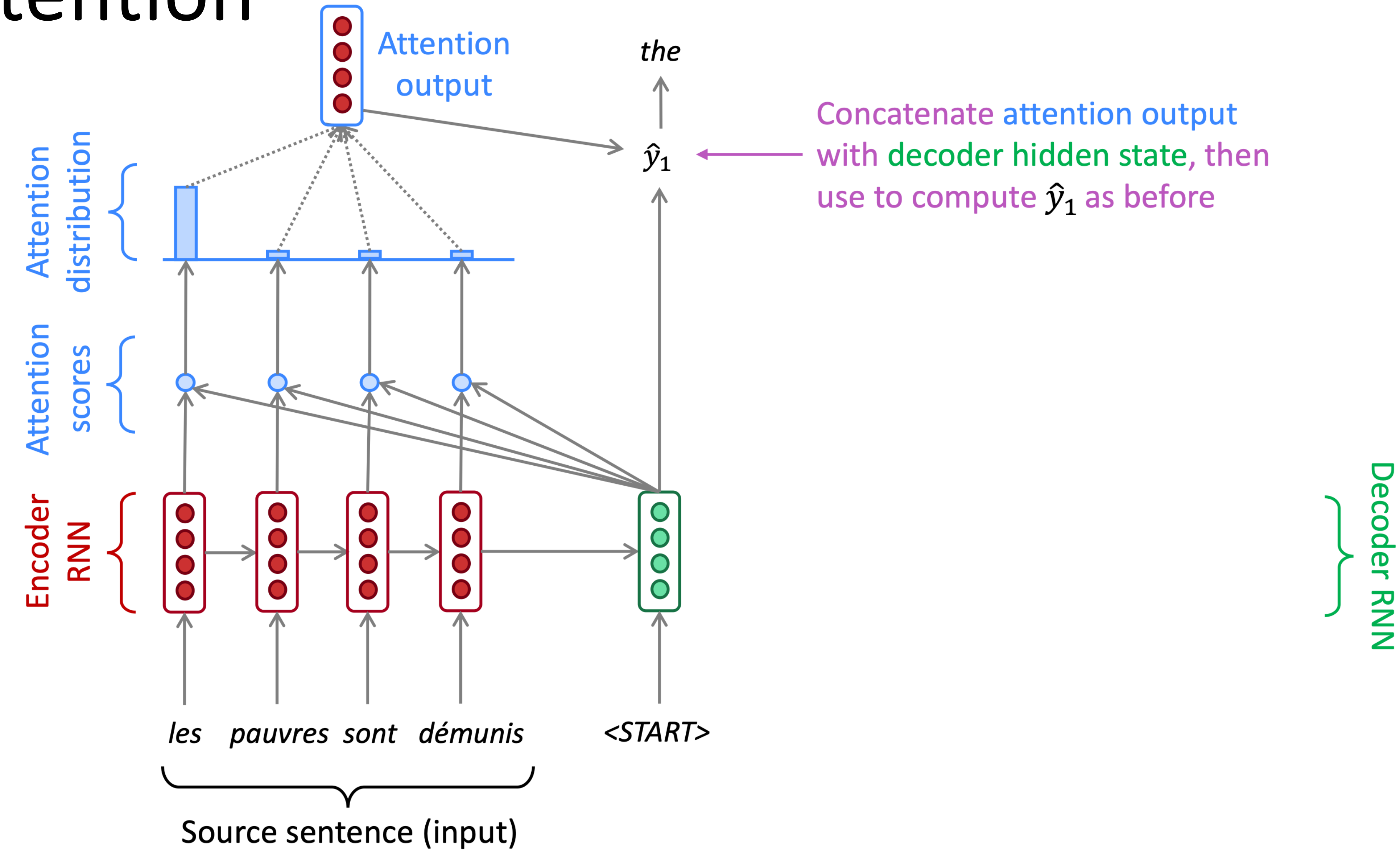
Attention



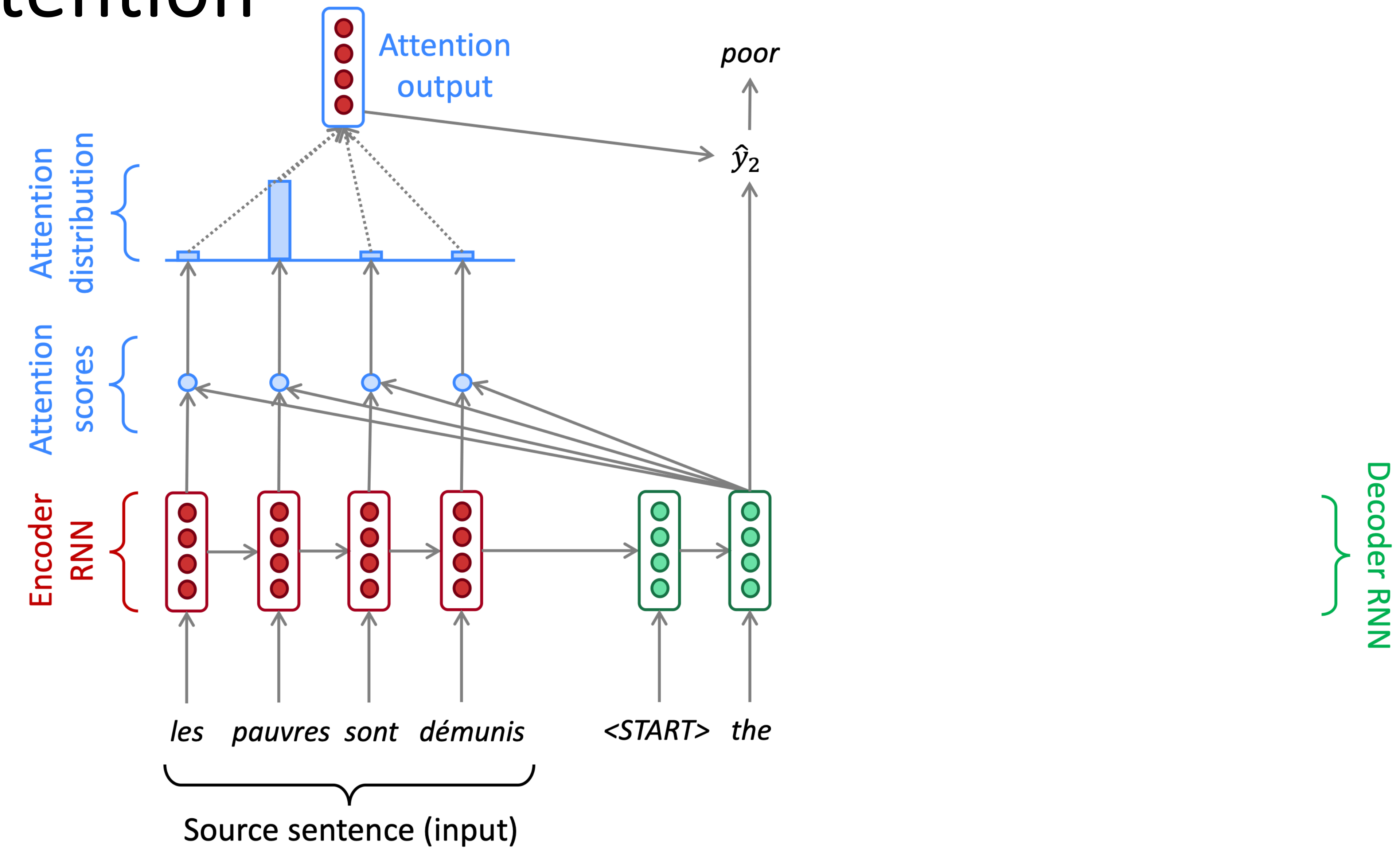
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

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

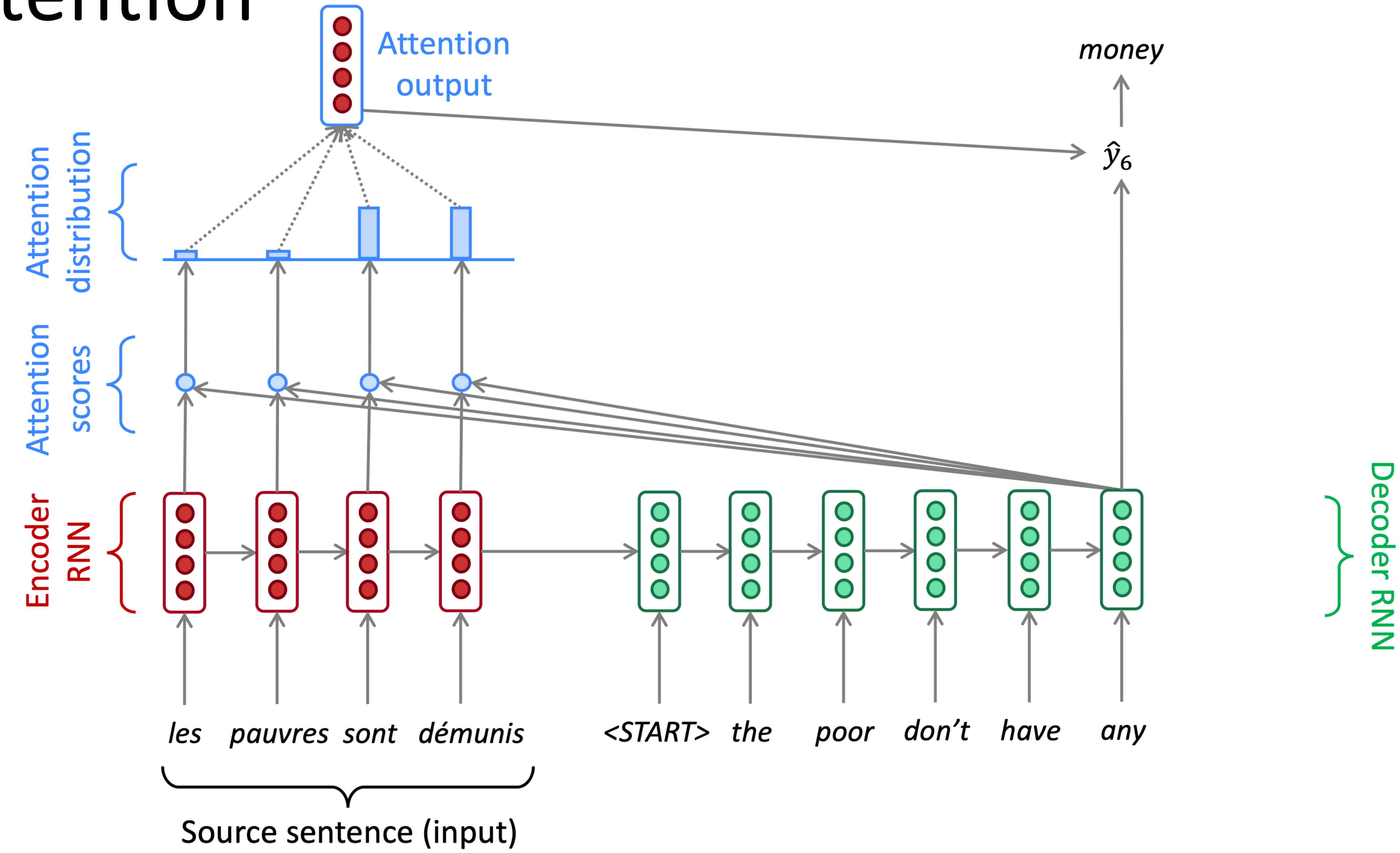
Attention


Attention output
the
$\hat{y}_1$
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
les
pauvres
sont
démunis
<START>
Source sentence (input)

Attention


Attention output
poor
$\hat{y}_2$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
les
pauvres
sont
démunis
<START>
the
Source sentence (input)

Attention



Attention output
money
$\hat{y}_6$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
les
pauvres
sont
démunis
<START>
the
poor
don't
have
any
Source sentence (input)

Attention

- 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

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

Transformers

- Composition of **self-attention layers**

- **Intuition**
  - Want sparse connection structure of CNNs, but with different structure
  - Can we **learn** the connection structure?

Self-Attention Layer

- **Self-attention layer:**

$$y[t] = \sum_{s=1}^{T} \text{attention}(x[s], x[t]) \cdot f(x[s])$$

  - Input first processed by local layer f
  - All inputs can affect $y[t]$
  - But weighted by $\text{attention}(x[s], x[t])$

- Resembles convolution but connection is learned instead of hardcoded

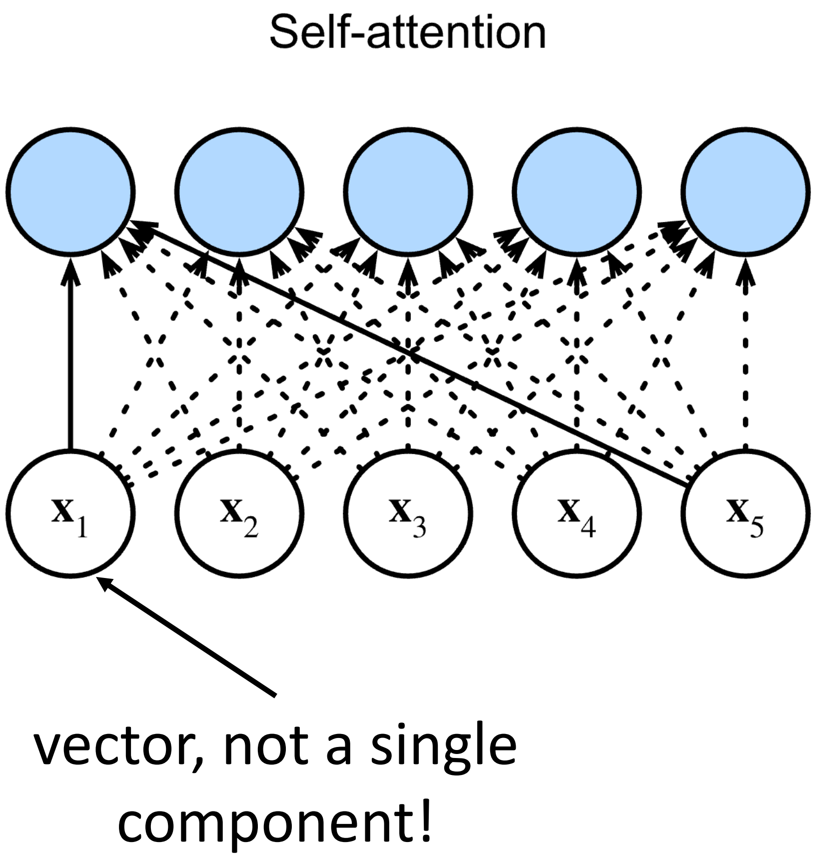


Figure credit to d2l.ai

Self-Attention Layer

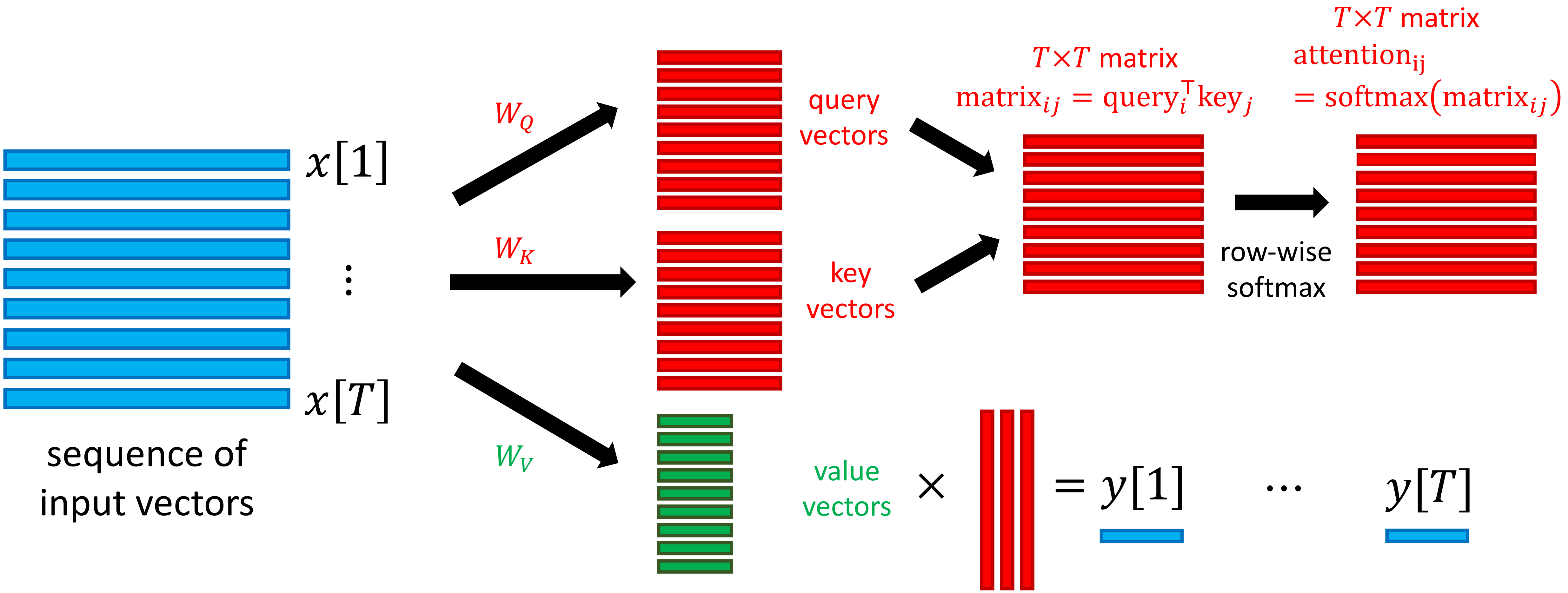
- **Self-attention layer:**

$$y[t] = \sum_{s=1}^{T} \mathrm{softmax}([\mathrm{query}(x[t])^\top \mathrm{key}(x[s])]) \cdot \mathrm{value}(x[s])$$

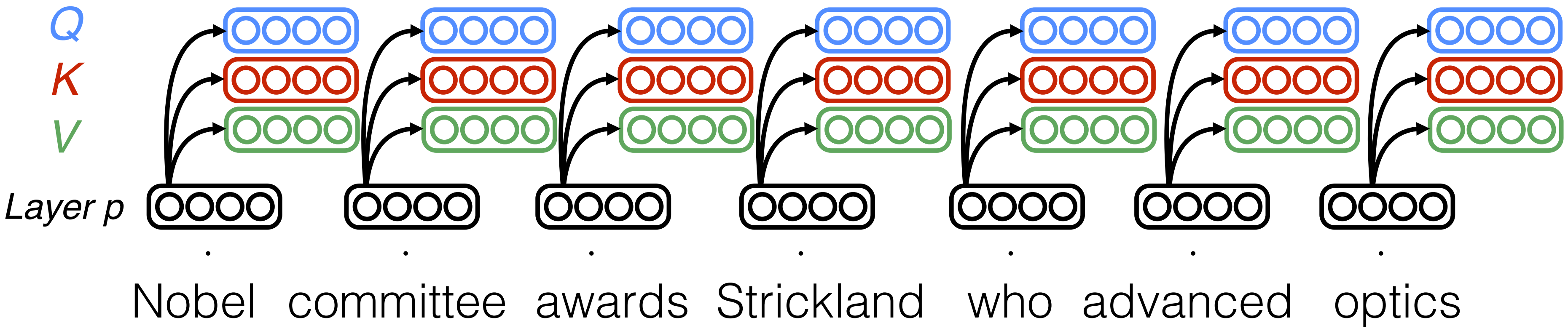
- Here, we have (learnable parameters are W_Q, W_K, and W_V):

$$\mathrm{query}(x[s]) = W_Q x[s]$$
$$\mathrm{key}(x[s]) = W_K x[s]$$
$$\mathrm{value}(x[s]) = W_V x[s]$$

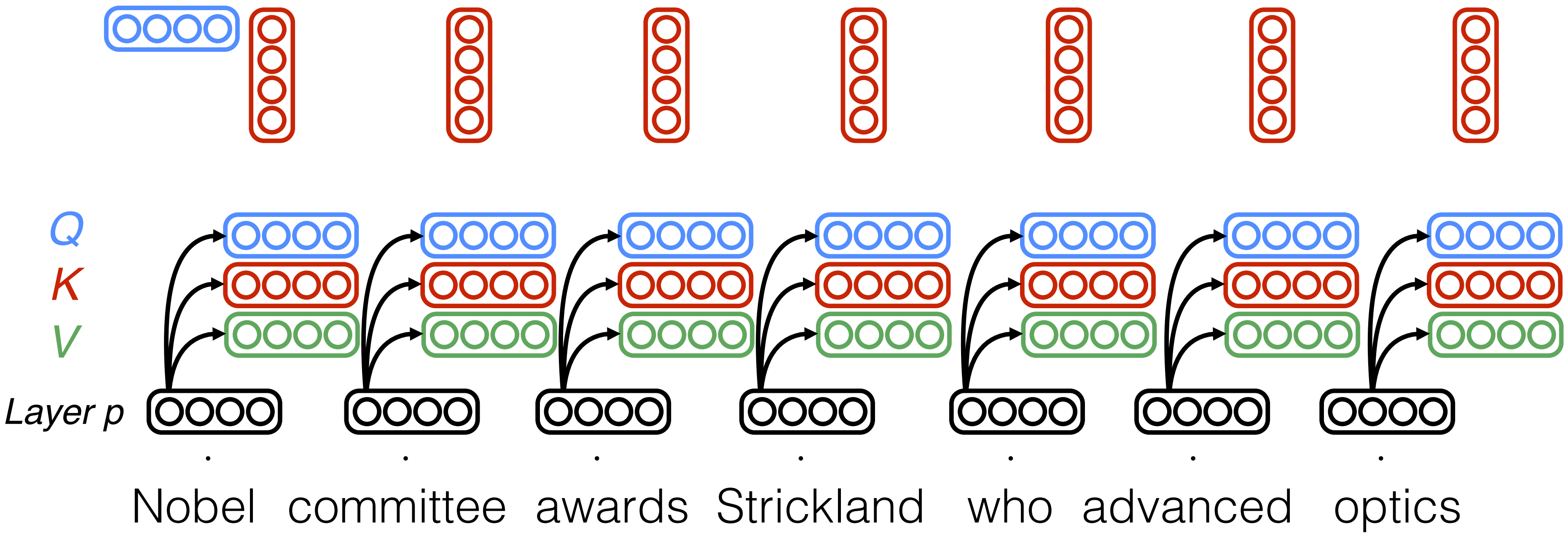
Self-Attention Layer



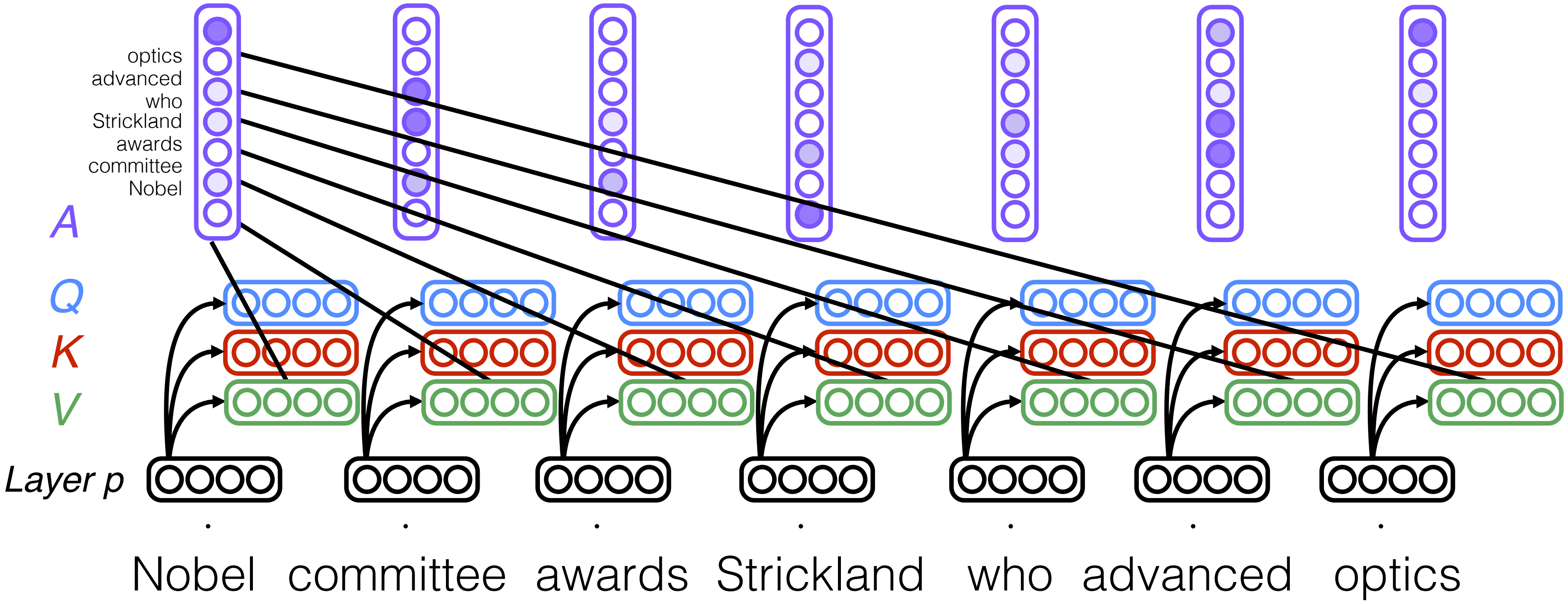
Self-Attention Layer



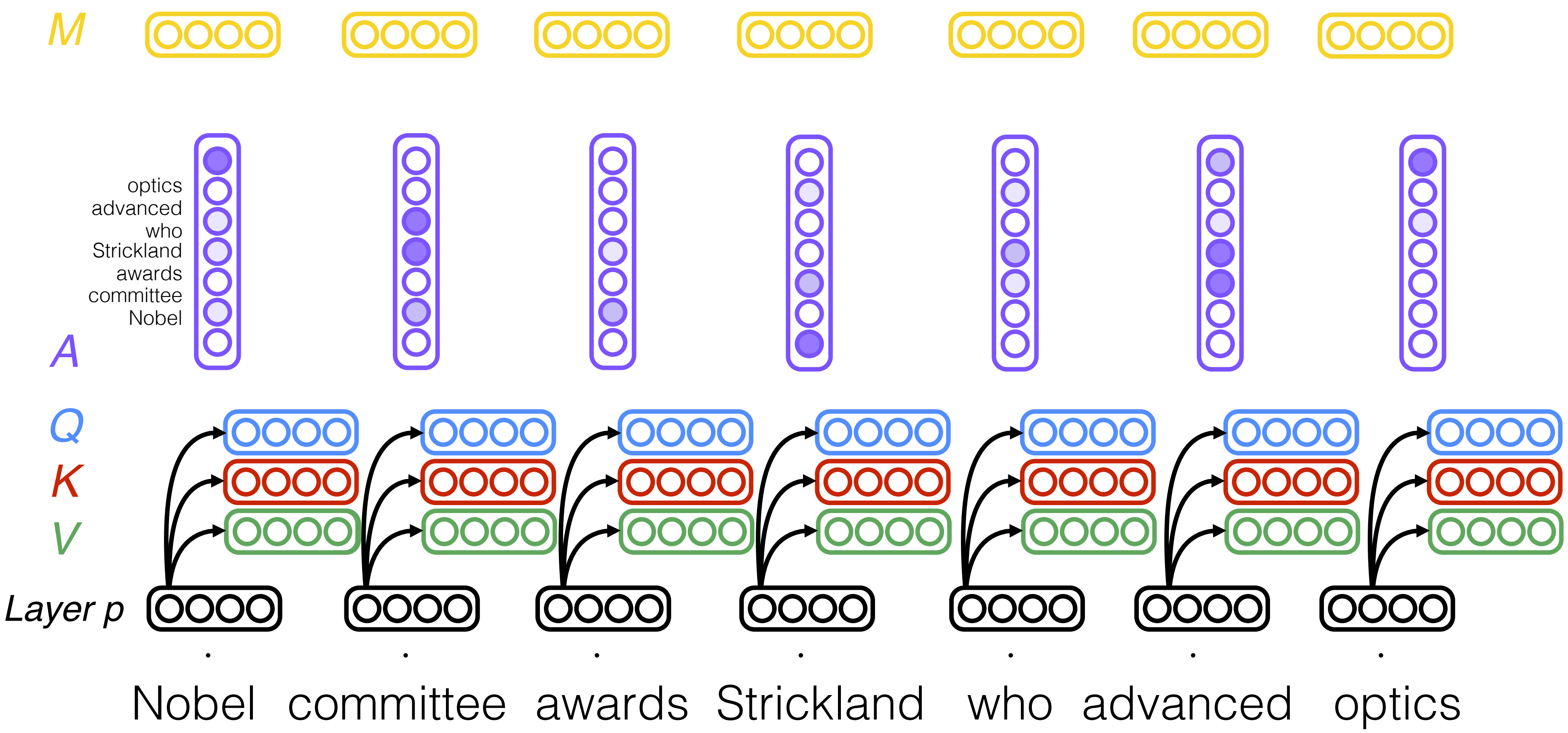
Self-Attention Layer



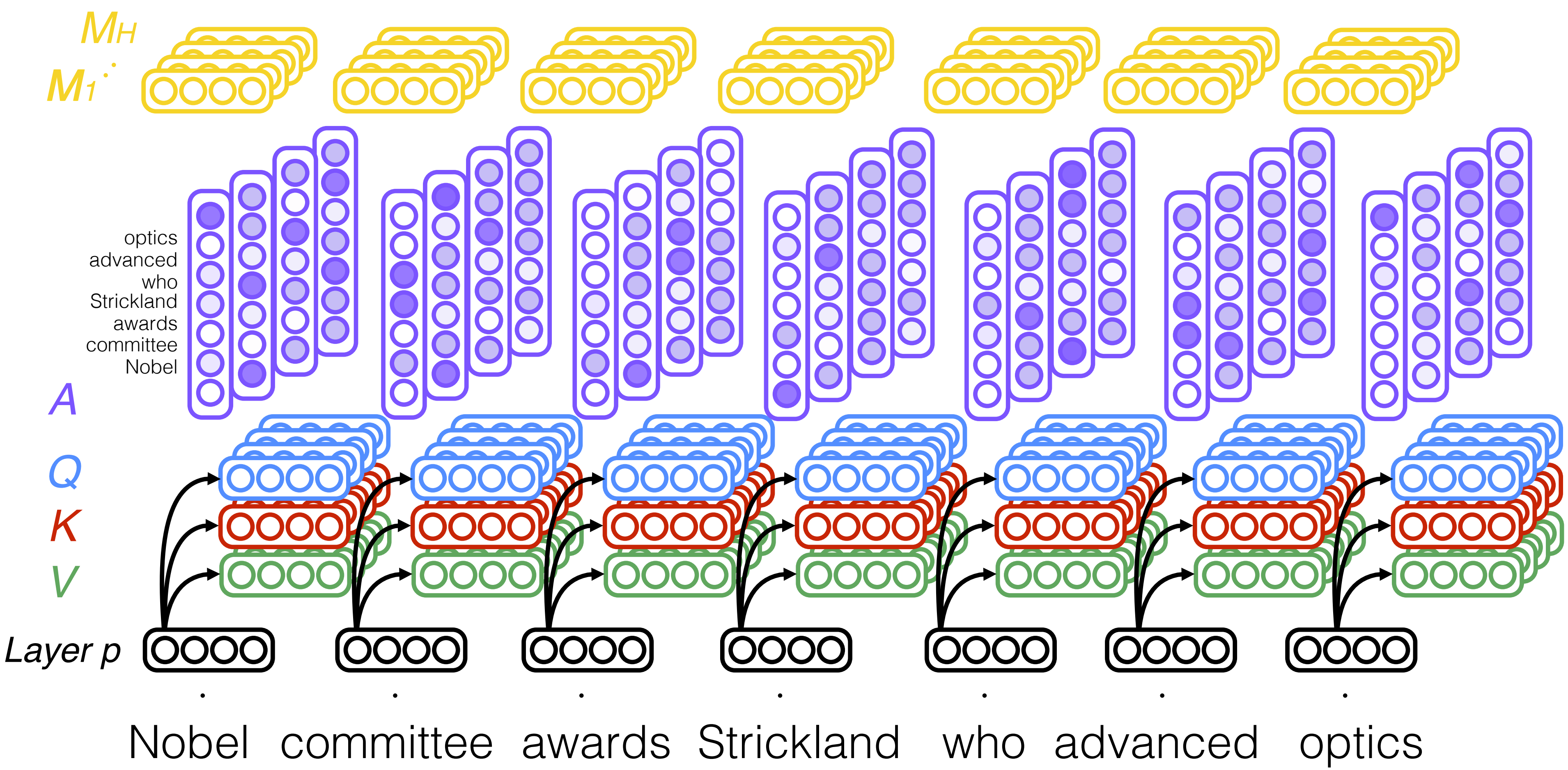
Self-Attention Layer



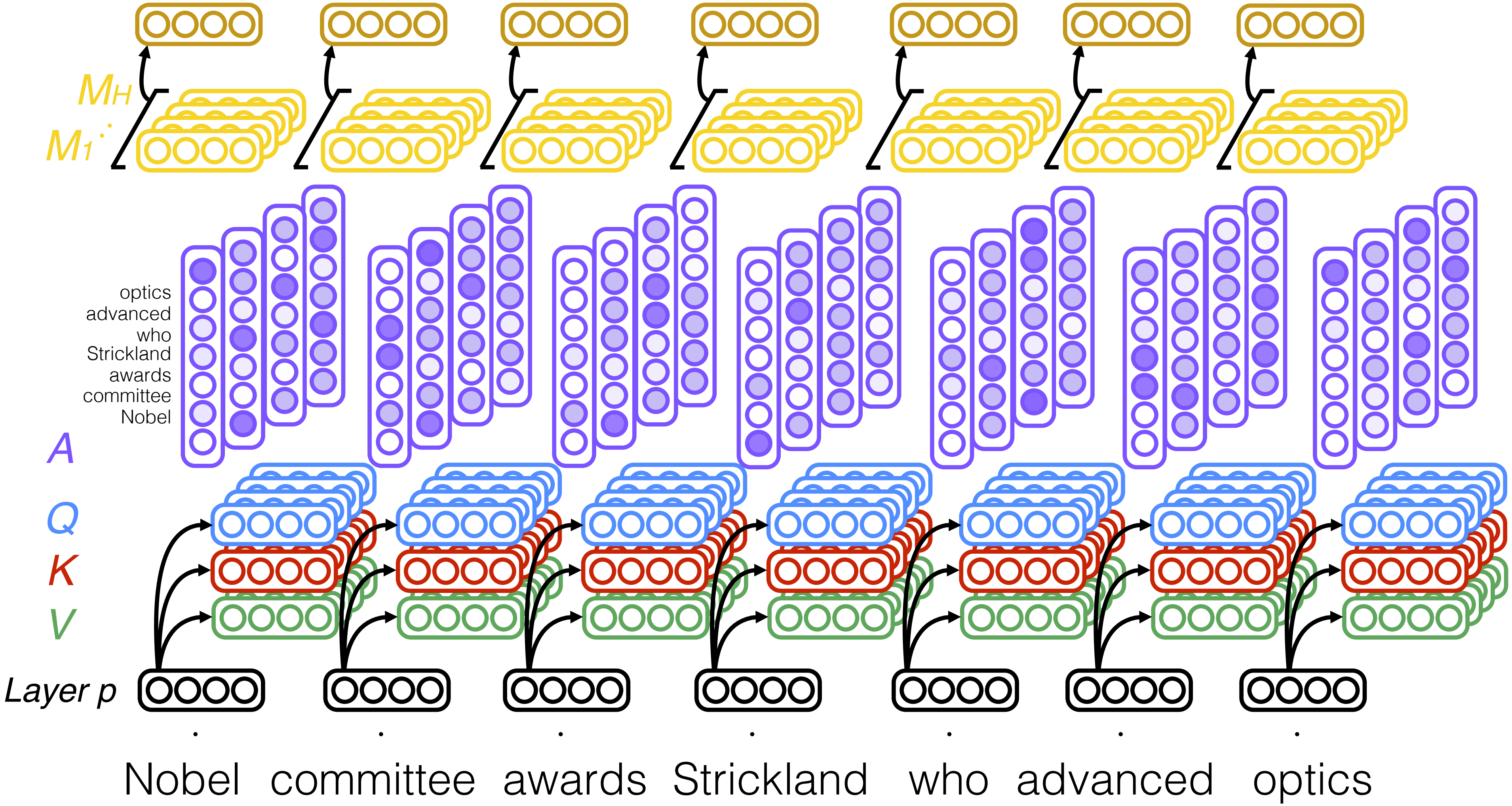
Self-Attention Layer

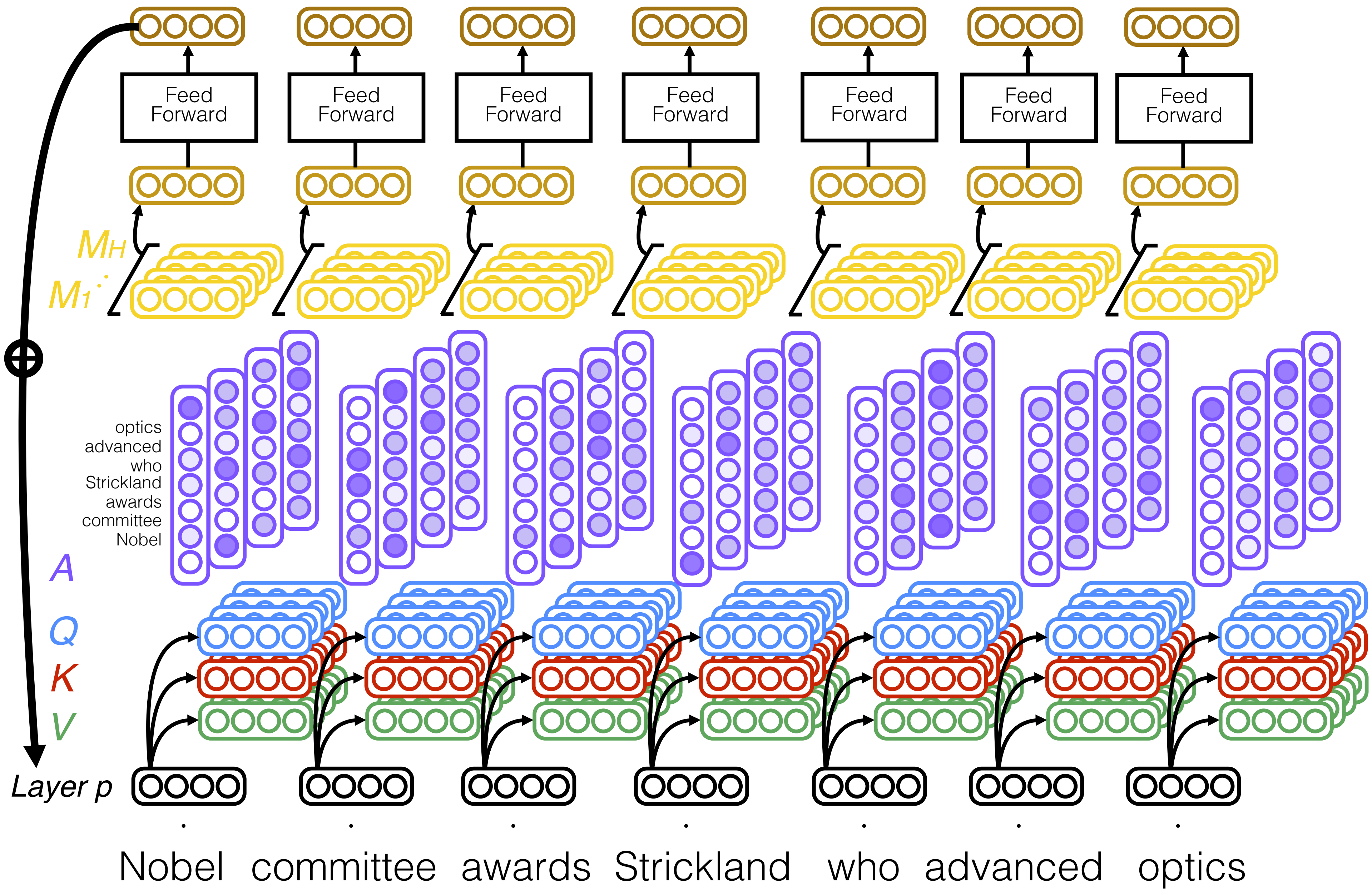


Multi-Head Self-Attention



Multi-Head Self-Attention




Feed Forward
M_H
M_1
optics
advanced
who
Strickland
awards
committee
Nobel
A
Q
K
V
Layer p
Nobel
committee
awards
Strickland
who
advanced
optics

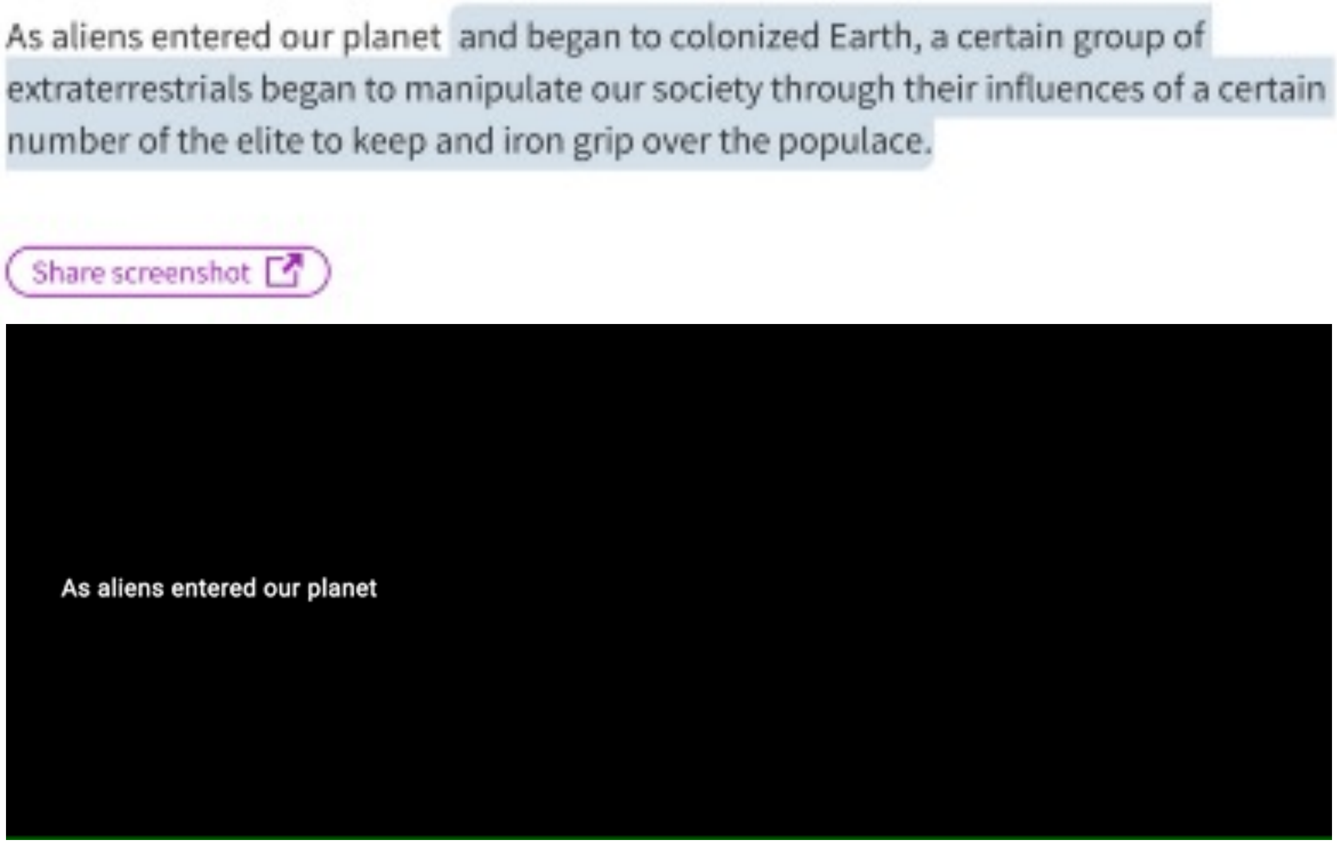
Transformers

- Stack self-attention layers to form a neural network architecture

- **Examples:**
  - **BERT:** Bidirectional transformer similar to ELMo, useful for prediction
  - **GPT:** Unidirectional model suited to text generation

- **Aside:** Self-attention layers subsume convolutional layers
  - Use “positional encodings” as auxiliary input so each input knows its position
  - https://d2l.ai/chapter_attention-mechanisms/self-attention-and-positional-encoding.html#
  - Then, the attention mechanism can learn convolutional connection structure

Visualizing Attention Outputs

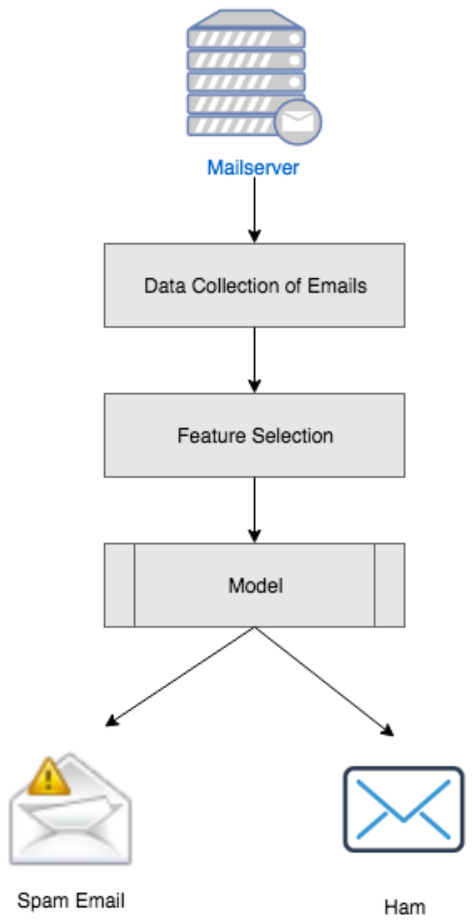


https://transformer.huggingface.co/

https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

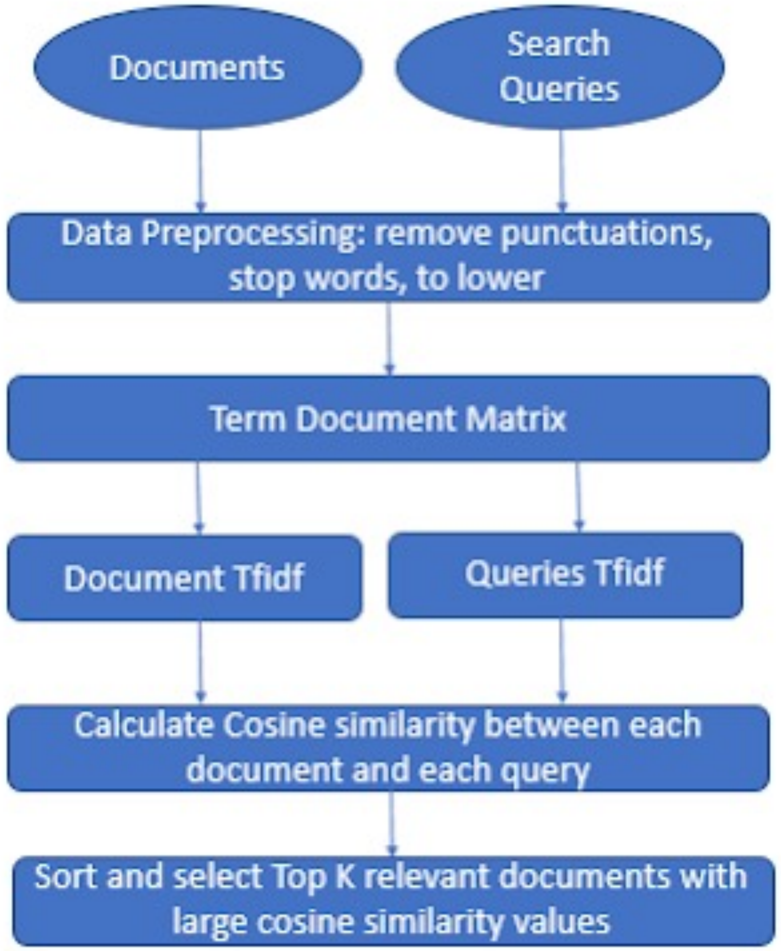
Applications: Spam Detection

- “Bag of words” + SVMs for spam classification
- **Features:** Words like “western union”, “wire transfer”, “bank” are suggestive of spam



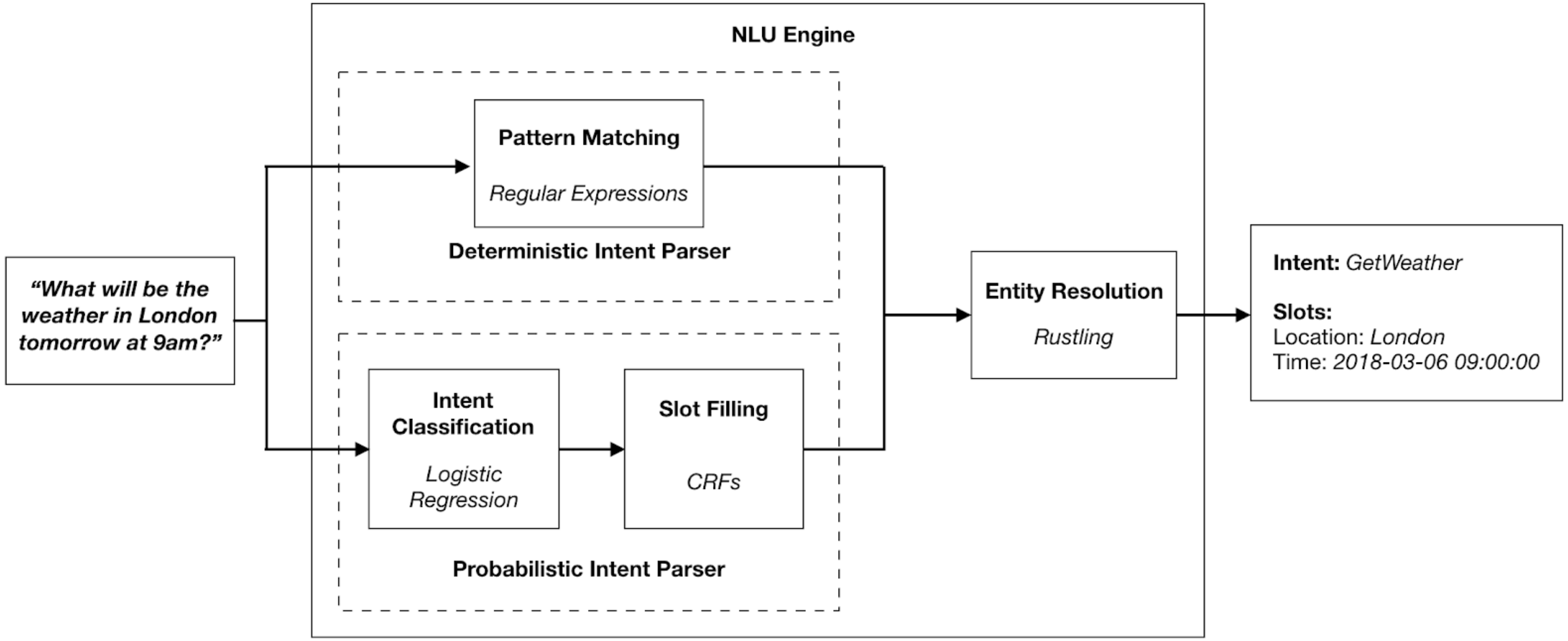
Applications: Search

- Use "bag of words" + TF-IDF to identify relevant documents for a search query



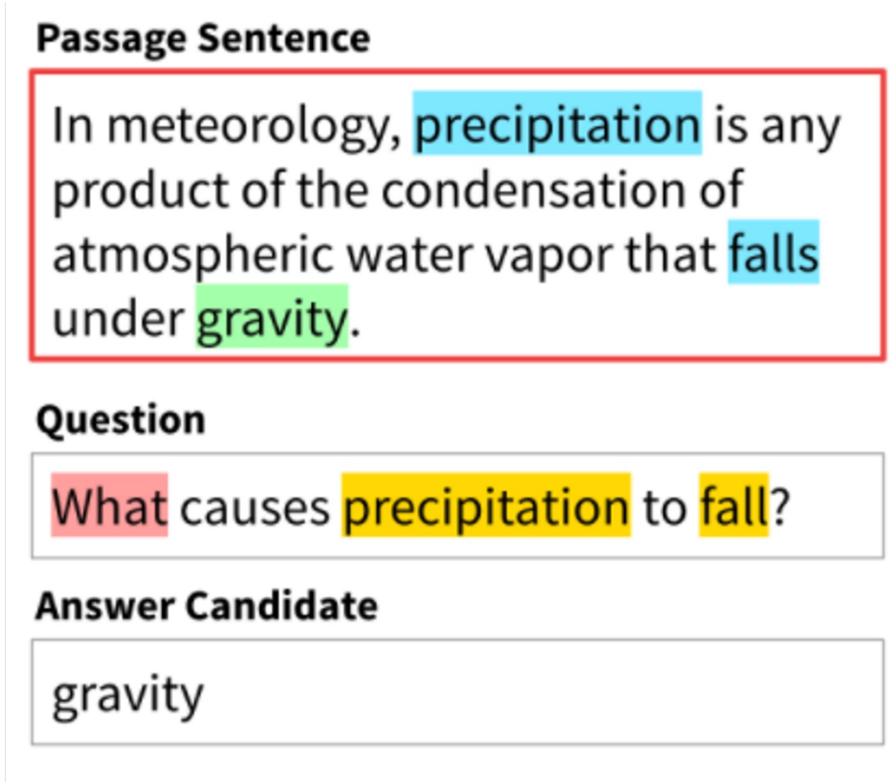
Applications: Virtual Assistants

- Use word vectors to predict intent of queries users ask



Applications: Question Answering

- Language models can be used to answer questions based on a given passage



Applications: Generation

- Language models can automatically generate text for applications such as video games

AI DUNGEON

AI Dungeon, an infinitely generated text adventure powered by deep learning.

Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: **After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.**
The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Transformers for Computer Vision

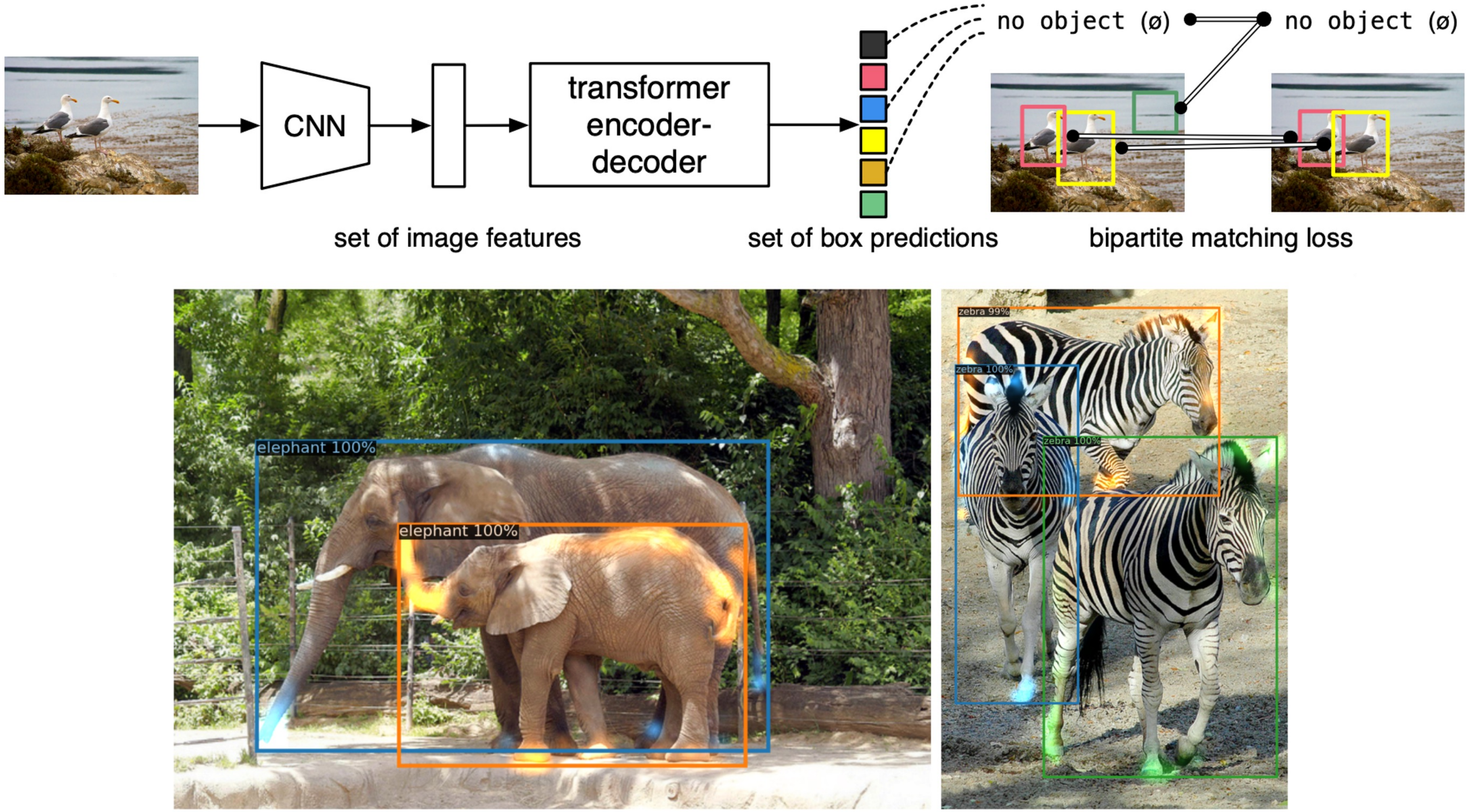


Figure credit to "End-to-End Object Detection with Transformers"