Lecture 16: NLP (Part 2)

CIS 4190/5190

Spring 2025

Slides adapted from Chris Callison-Berch and Luke Zettlemoyer and Mark Yatskar

Recap: Bag of Words Representation

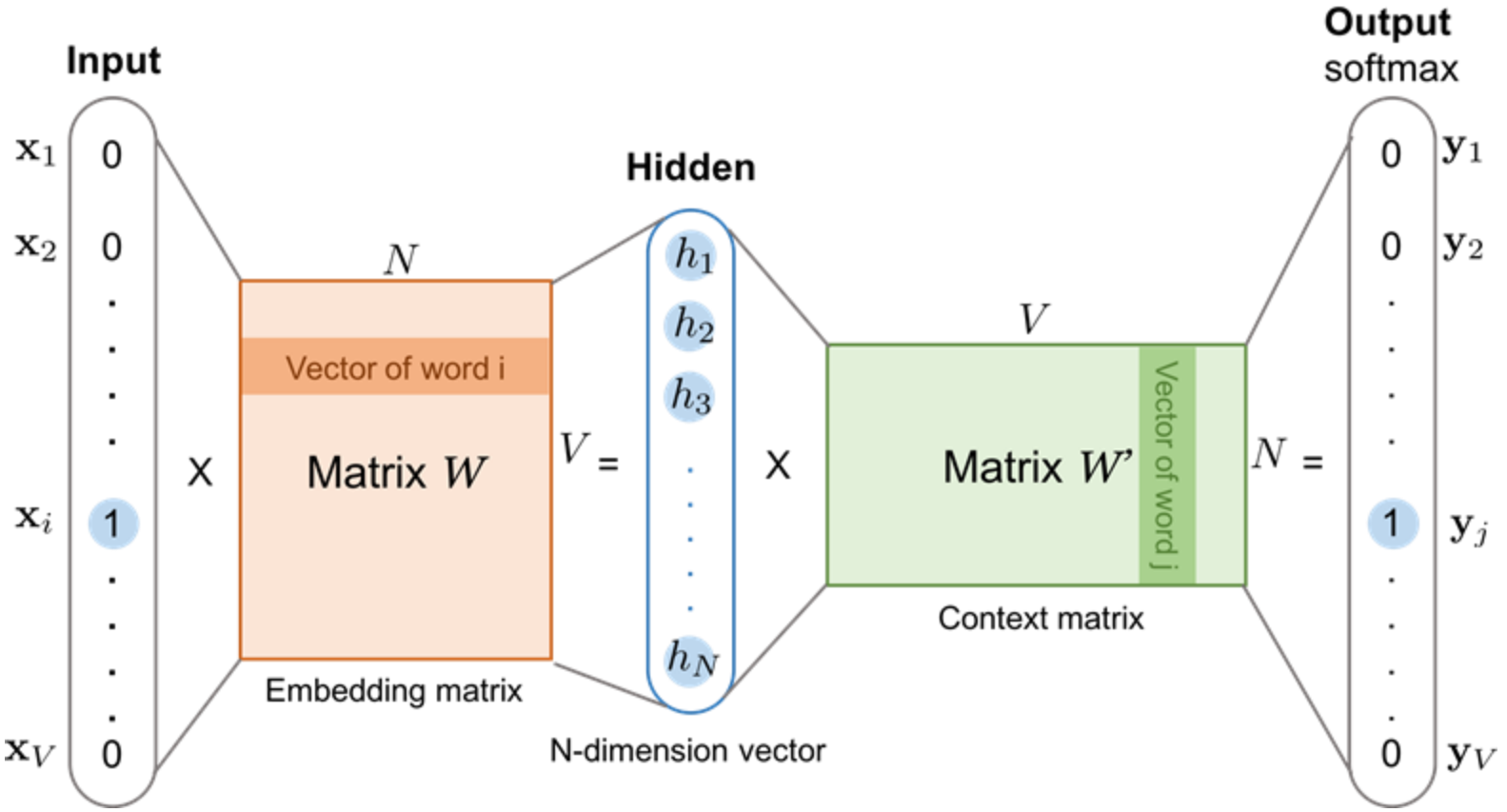
Assumption: The ordering of words does not matter, only what occurred

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it
I
the
to
and
seen
yet
would
whimsical
times
sweet
satirical
adventure
genre
fairy
humor
have
great
…

Recap: Word2Vec

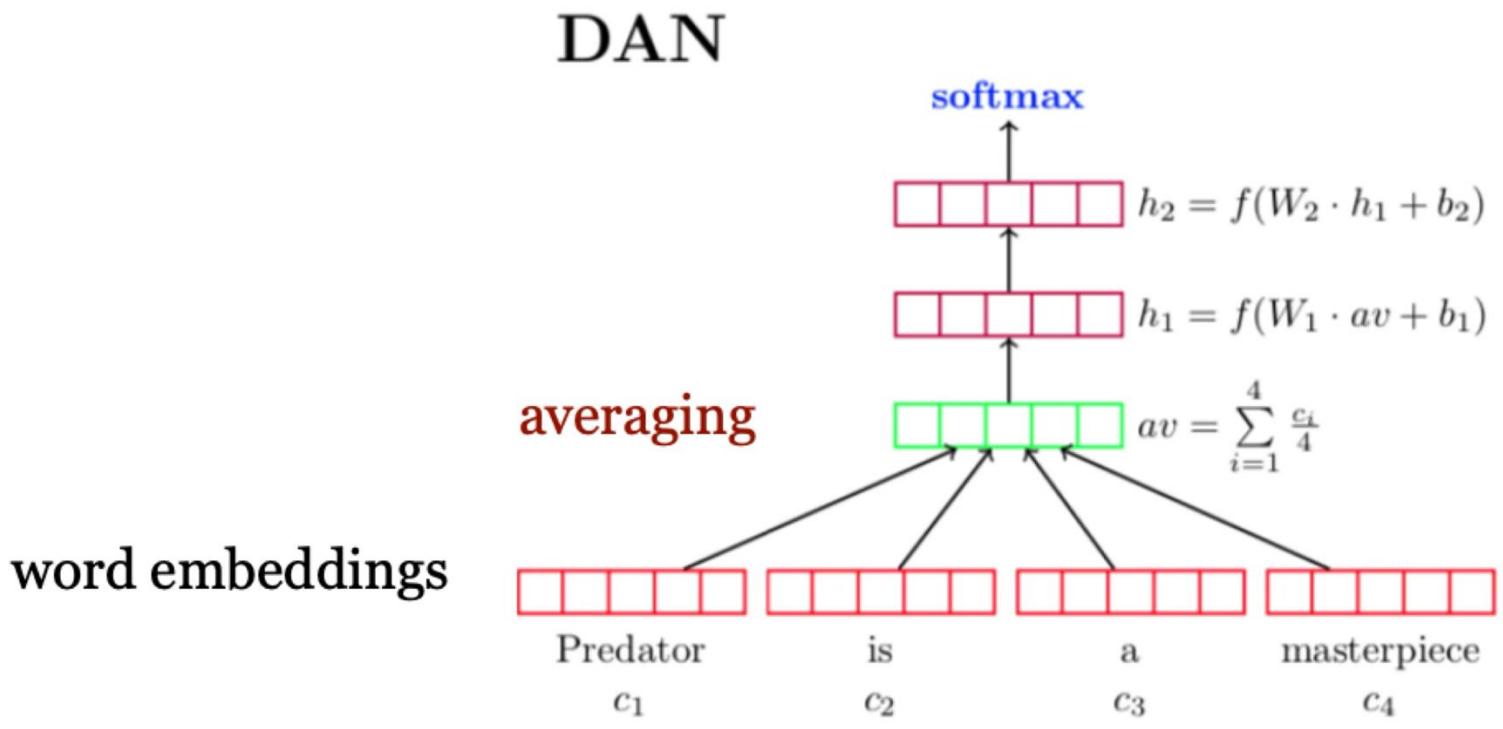
Idea: Given a word, predict the words you expect to see in the context



We can concatenate the target and context embeddings to form our final word embedding

Deep Averaging Networks

- Deep Averaging Networks (DAN) for Text Classification



(Iyyer et 2015): Deep Unordered Composition Rivals Syntactic Methods for Text Classification

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!

Updating Word Embeddings

- Word embeddings can be treated as parameters too!

$$\theta = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}, \mathbf{w}^{(o)}, b^{(o)}, \mathbf{E}_{emb}\}$$

- When the training set is small, don't re-train word embeddings (think of them as features!).

- Most cases: initialize word embeddings using pre-trained ones (word2vec, Glove) and re-train them for the task

- When you have enough data, you can just randomly initialize them and train from scratch (e.g. machine translation)

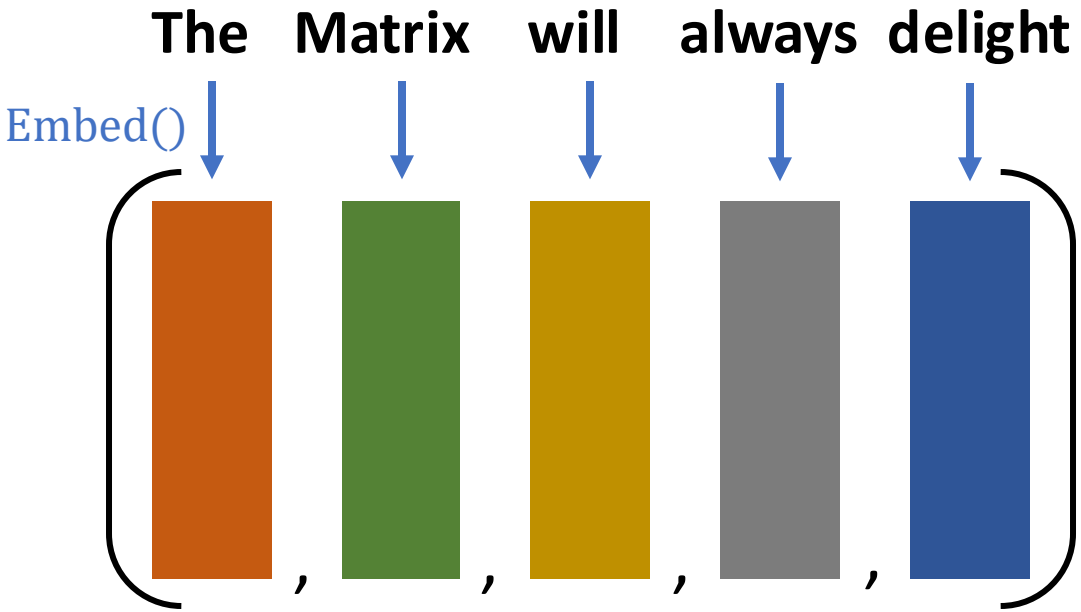
So far

- **Classical approach:** Feature engineering + Standard ML model

- **Semi-Classical approach:** Word2Vec + Standard ML model
  - Sum embeddings of words to get document features
  - Still “bag-of-words” like model! ($\mathrm{Embed}(i) = \mathrm{OneHot}(i)$) is bag of words)

This Lecture: Sequence Models

- Recurrent Neural Networks
- Attention and Transformers



Sequence models have produced huge advances in NLP

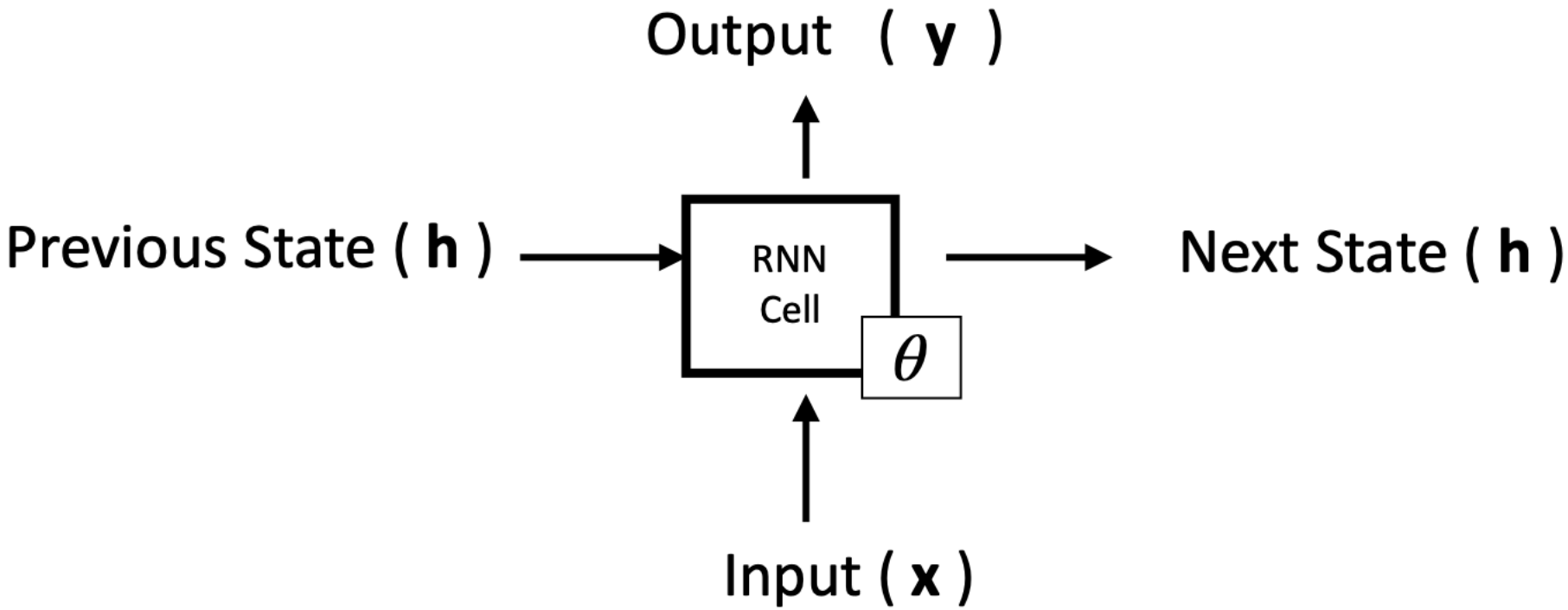
Recurrent Neural Networks (RNN)

- A class of NNs allowing to handle **variable length inputs**
  - Why variable length: Relationships in sentences can be extremely long distance

- Process input **sequentially**

Abstract RNN

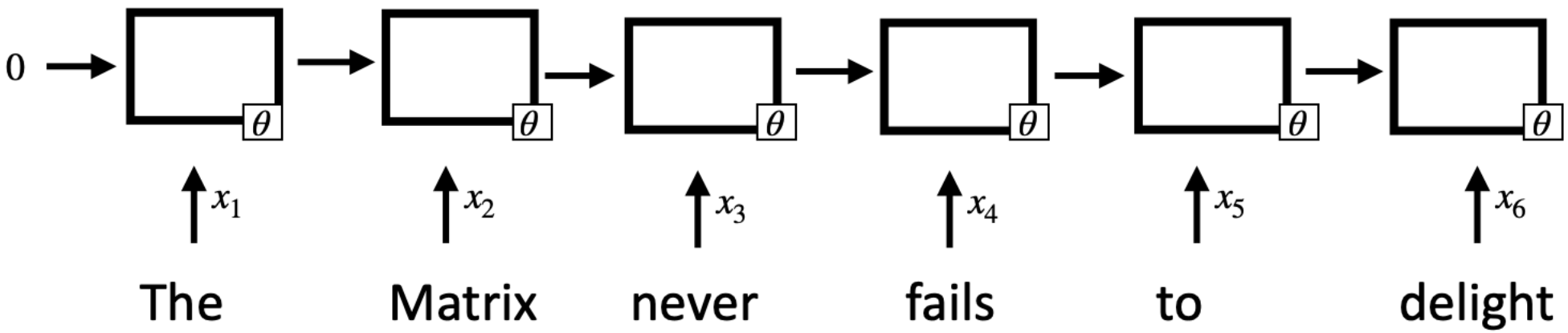
A function which takes some input **x** (i.e. vector for a word) and some previous state **h**, and outputs a vector **y** and updates the hidden state



Abstract RNN - Unrolled in Time

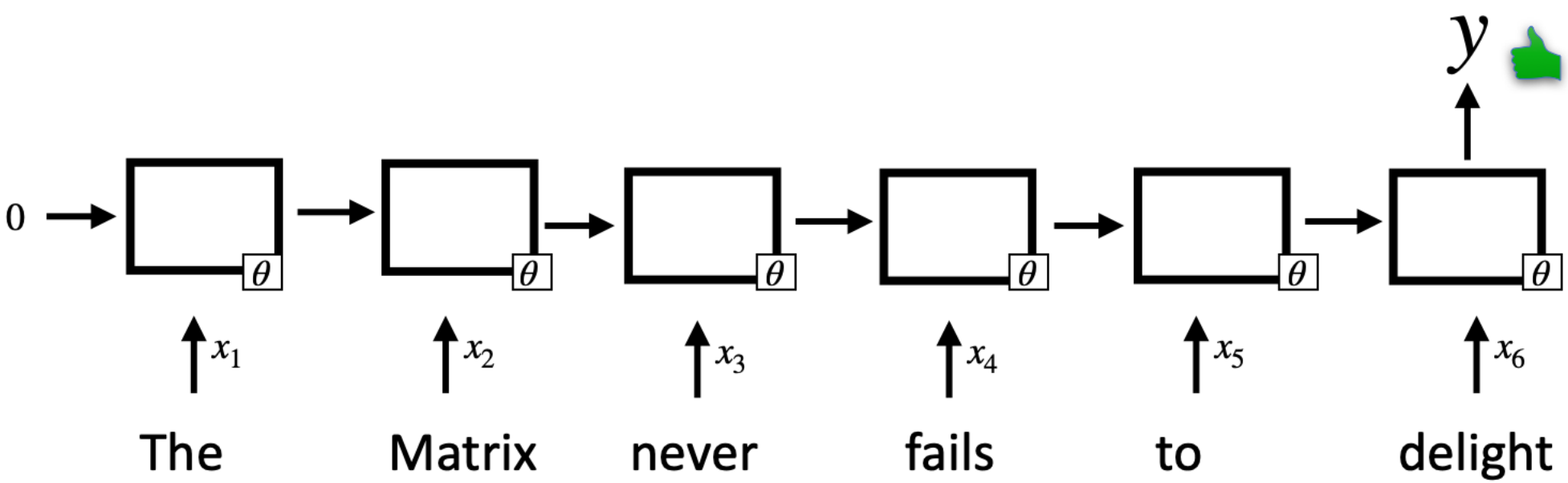
A function which takes some input **x** (i.e. vector for a word) and some previous state **h**, and outputs a vector **y** and updates the hidden state

Input sequence of any length — treat as one giant neural network



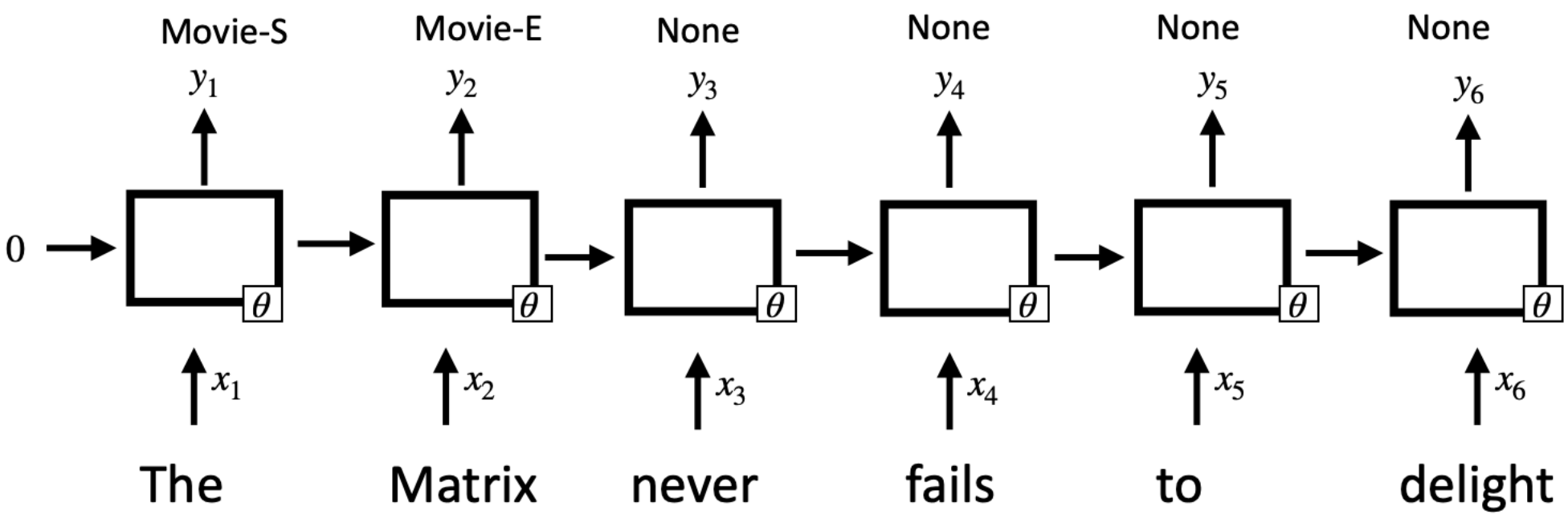
Abstract RNN - Unrolled in Time

Sentiment Analysis:
Did the writer like the movie?



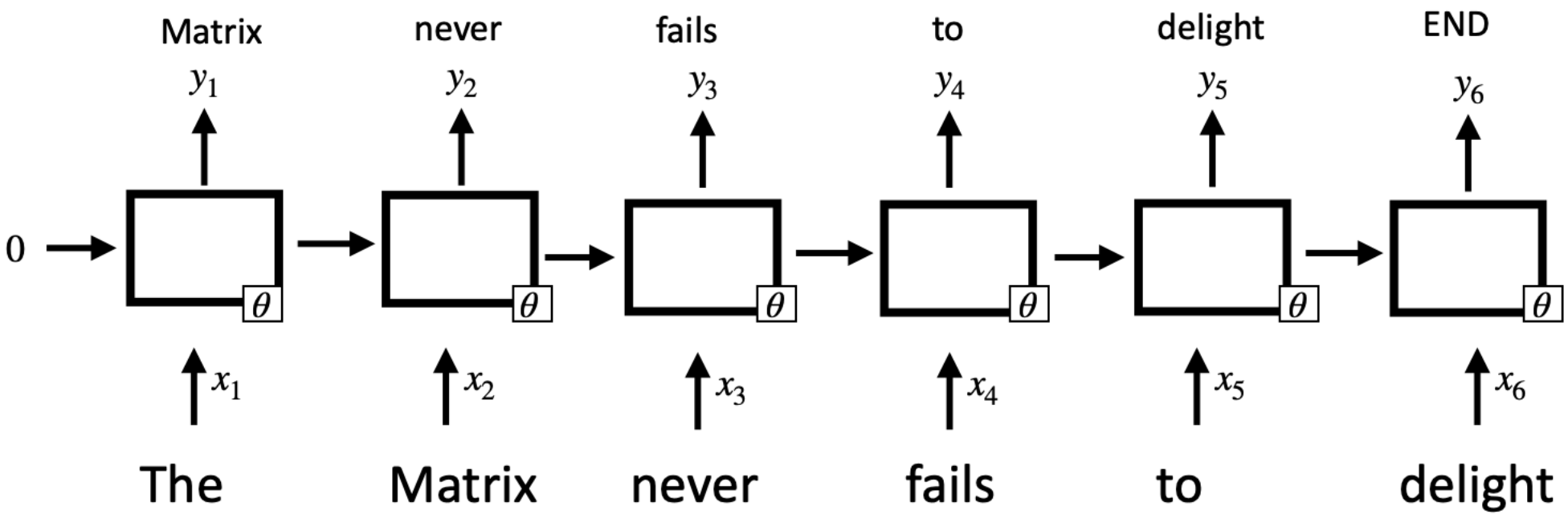
Abstract RNN - Unrolled in Time

Named Entity Recognition:
Which words are movies?

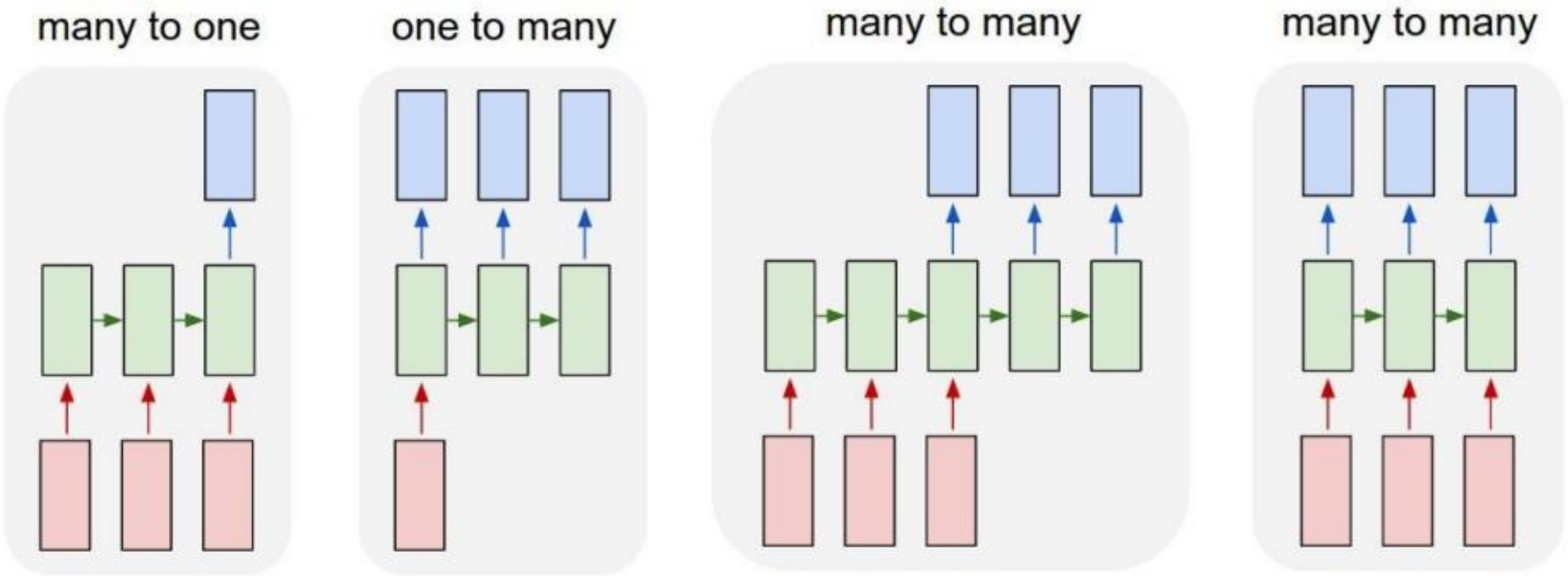


Abstract RNN - Unrolled in Time

Language Modeling:
Which word comes next?



Recurrent Neural Networks



Sentiment prediction

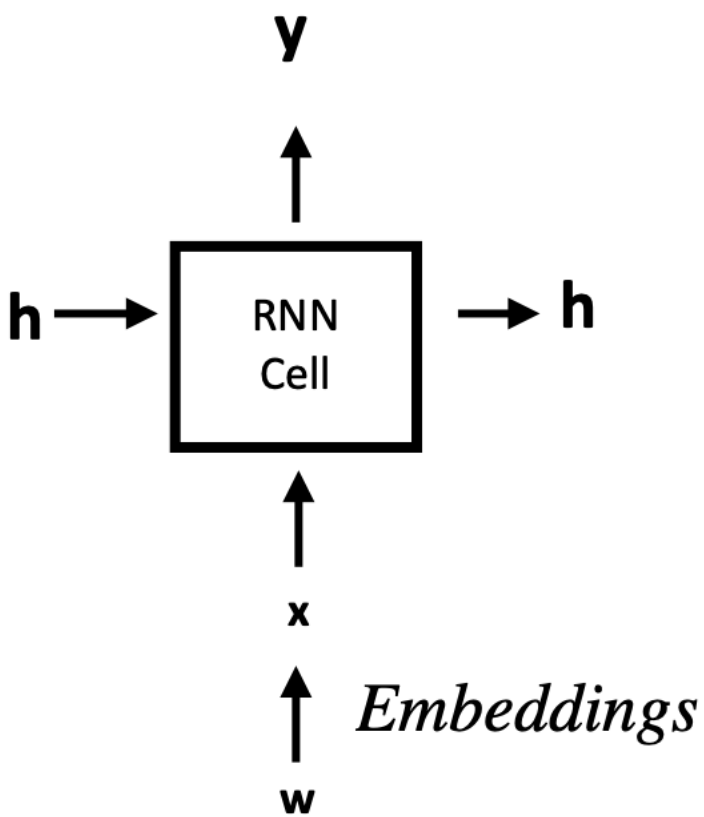
Image captioning

Machine translation

Video captioning

Simple RNN

Next state : non-linearity applied to sum of projections of input and previous state



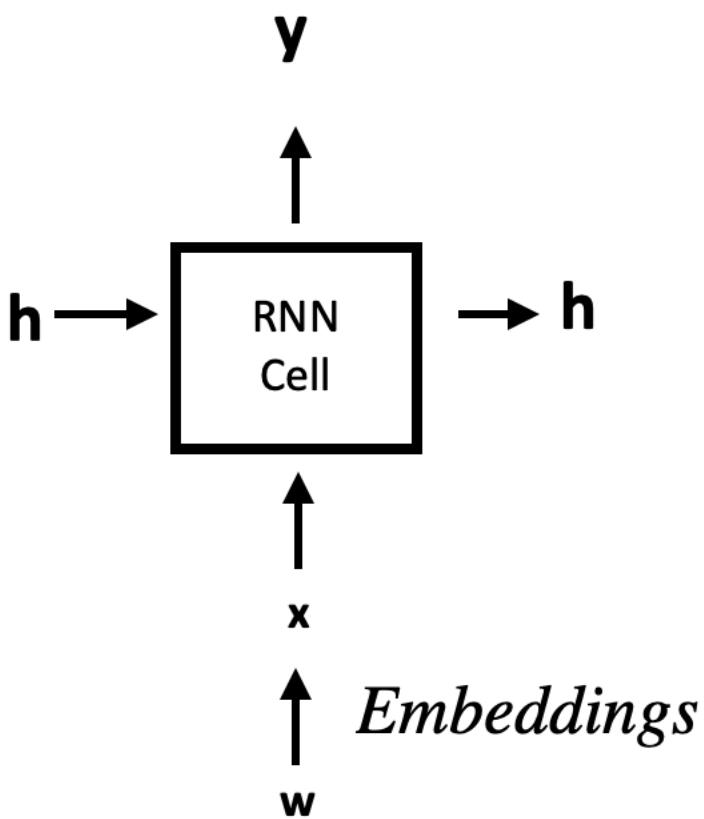
$$h_t = g(Vx_t + Uh_{t-1} + b_h) \in R^d$$

g: Non-linearity (tanh)

Elman (1990)

Simple RNN

Next state : non-linearity applied to sum of projections of input and previous state



$$h_t = g(Vx_t + Uh_{t-1} + b_h) \in R^d$$

g: Non-linearity (tanh)

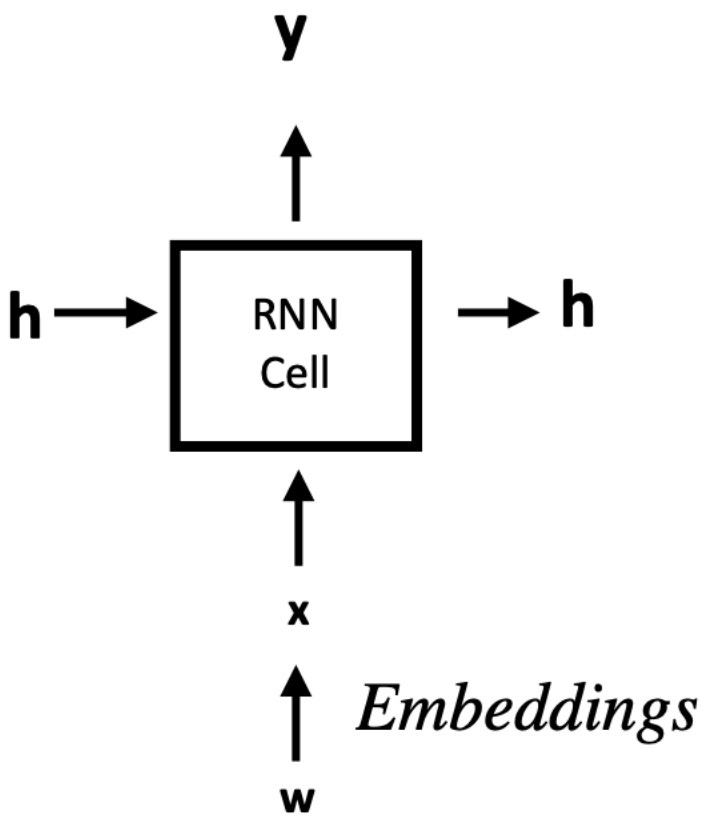
How interested is a cell in the features of the input?

How interested is a cell in the features of the previous state?

Elman (1990)

Simple RNN

Next state : non-linearity applied to sum of projections of input and previous state



$$h_t = g(Vx_t + Uh_{t-1} + b_h) \in R^d$$

g: Non-linearity (tanh)

$$y_t = Oh_t$$

What parts of the hidden state are relevant to output

Elman (1990)

Training RNNs

- Backpropagation works as before
  - For shared parameters, we can show that the overall gradient is the 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

$$h_1 = g(Vx_1 + Uh_0 + b_h)$$
$$h_2 = g(Vx_2 + Uh_1 + b_h)$$
$$h_3 = g(Vx_3 + Uh_2 + b_h)$$

$$\frac{\partial L}{\partial U} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial U} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial U} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial U}$$

Local Contribution

Historical Contribution

Exploding & Vanishing Gradients

$$\frac{\partial L}{\partial U} = \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial U} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial U} + \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial U}$$

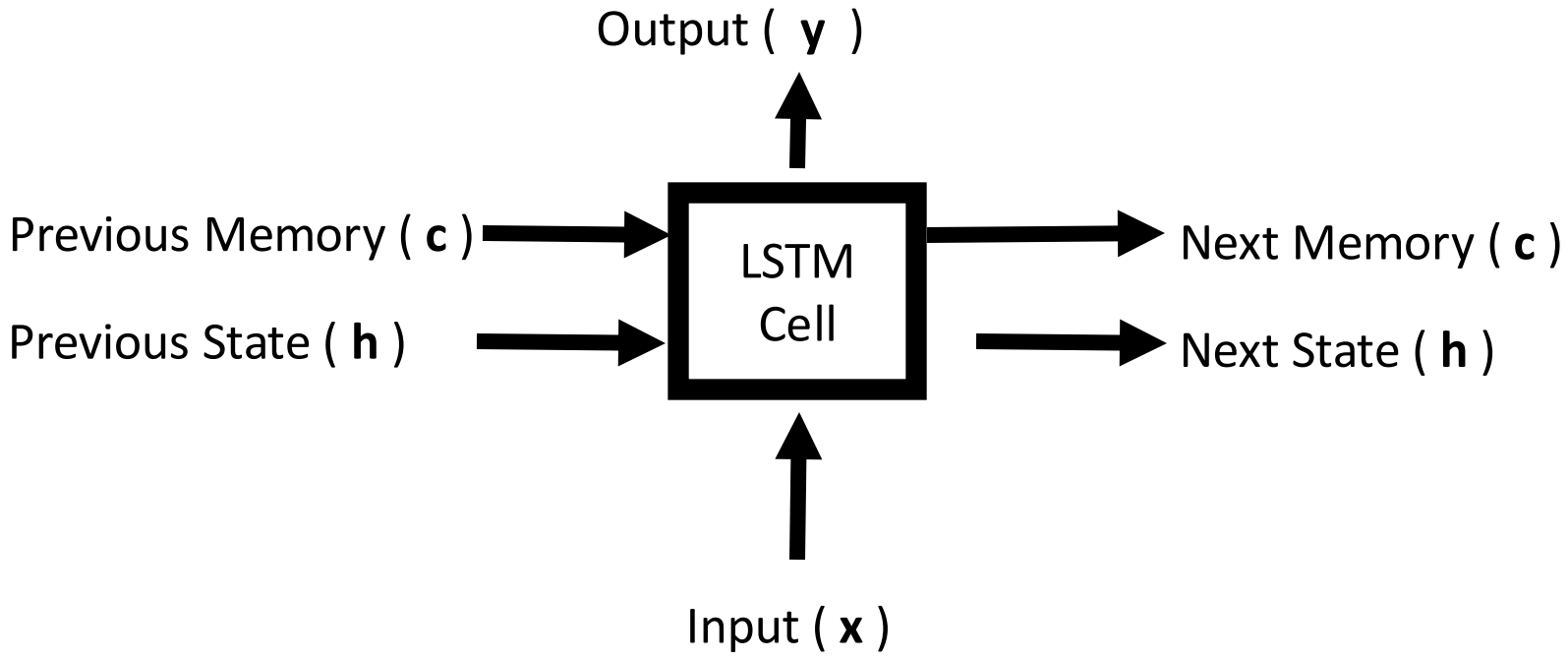
- Multiplicative contributions cause exploding/vanishing gradients
  - If too large — Large gradients will destabilize gradient descent algorithms.
  - If too small — History doesn't matter anymore in the optimization!

Solution: Change the form of the recurrent cell

Long Short Term Memory (LSTM)

Replace multiplicative relationships in hidden state with additive ones.

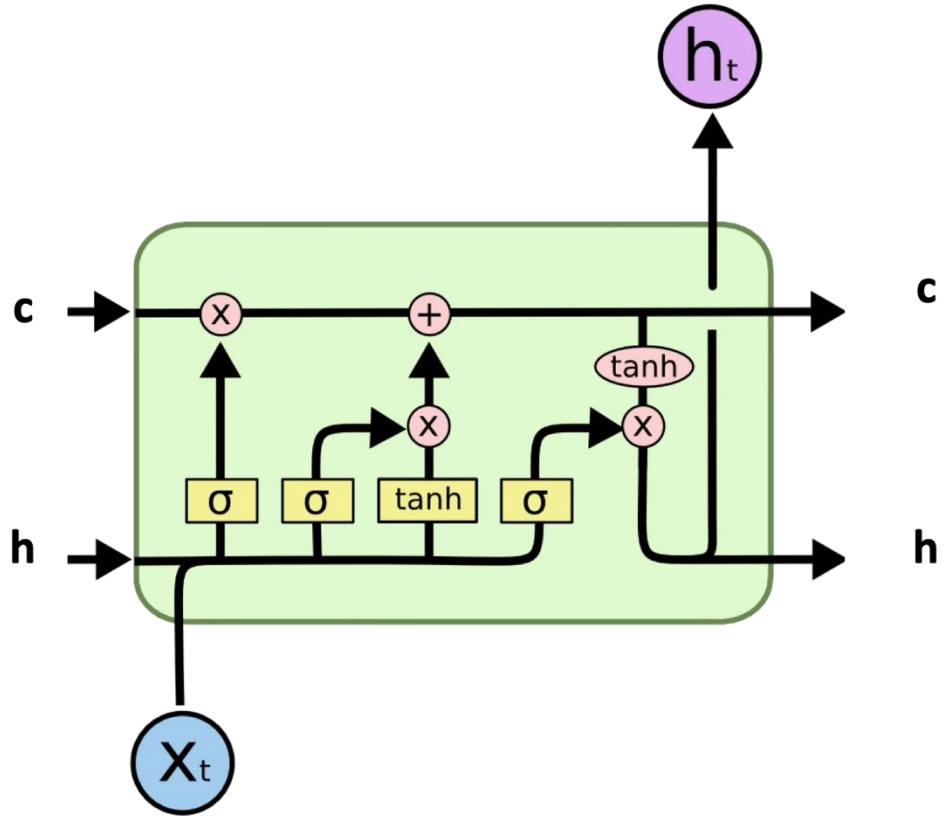
Introduce second hidden state cell memory (c)



Long Short Term Memory (LSTM)

Replace multiplicative relationships in hidden state with additive ones.

Introduce second hidden state cell memory (c)



Pretraining RNNs

- **Unsupervised pretraining (Next word/token prediction)**
  - Train on dataset of text to predict next word (classification problem)
  - $x = w_1 w_2 \ldots w_t$ 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”

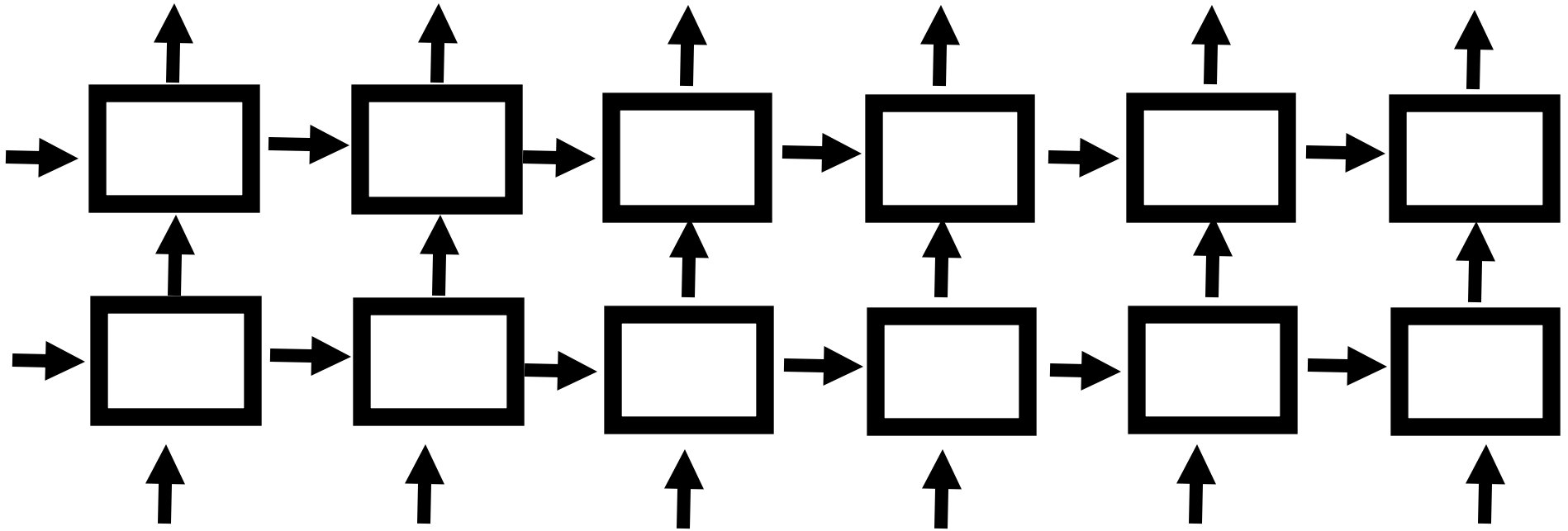
Shortcomings of RNNs

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

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

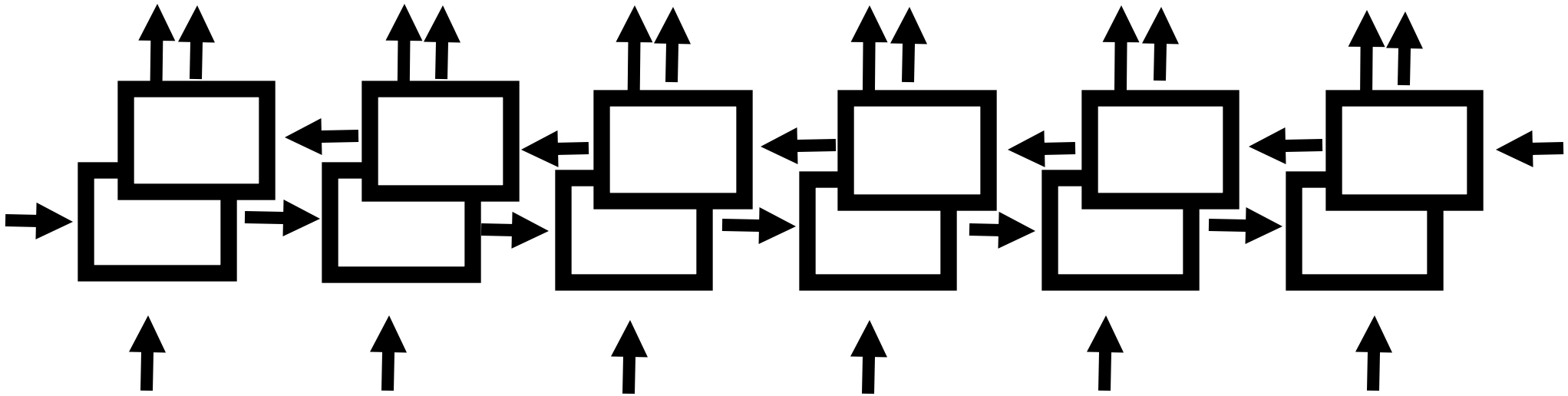
Stacked RNN

Allow multiple levels of processing

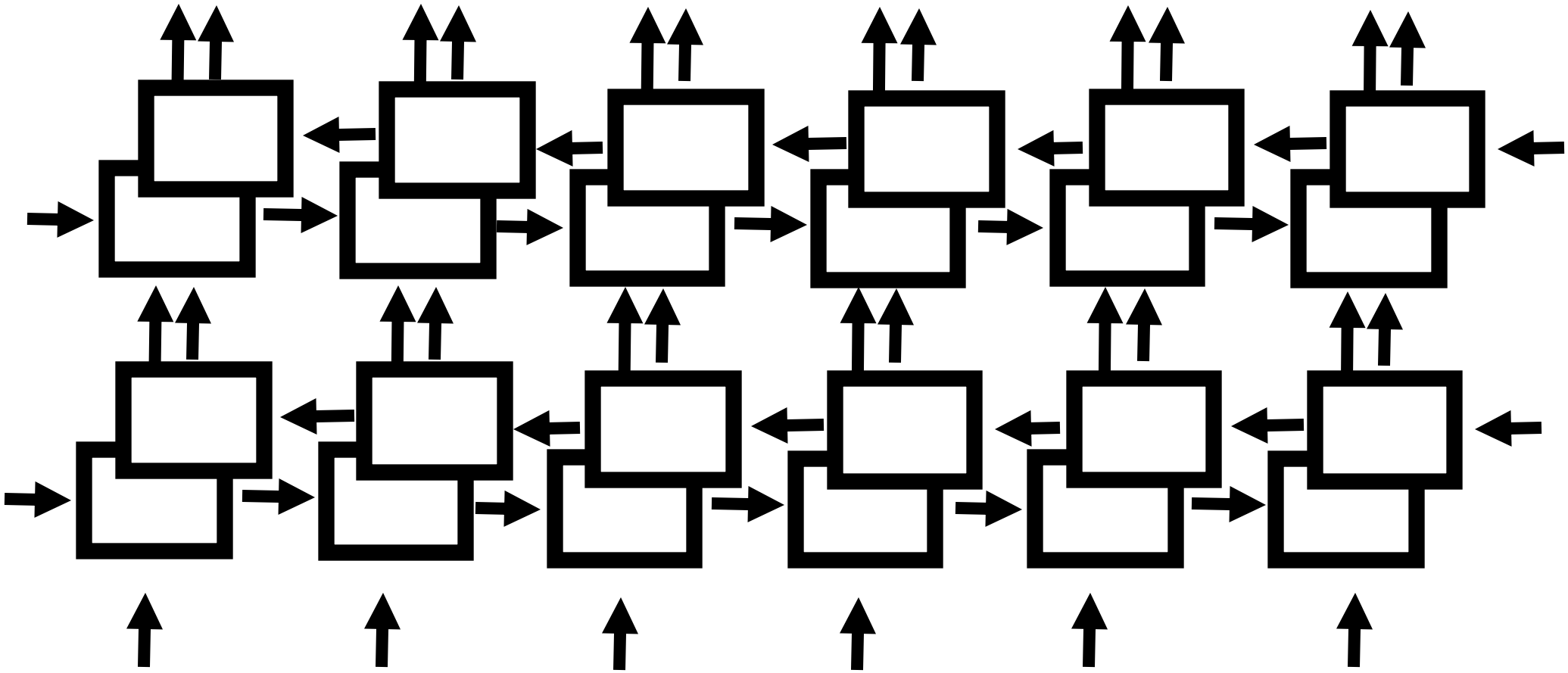


Bidirectional RNN

Run both ways, and then concat the outputs together

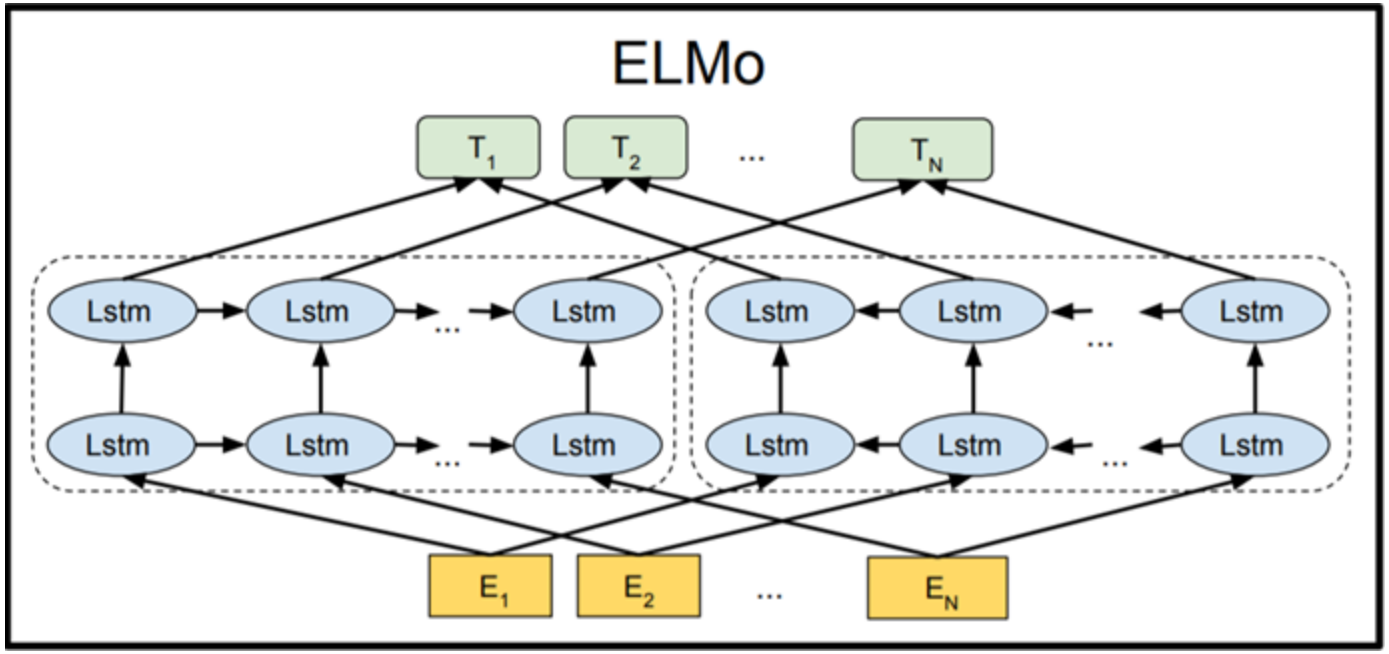


Stacked + Bidirectional RNN

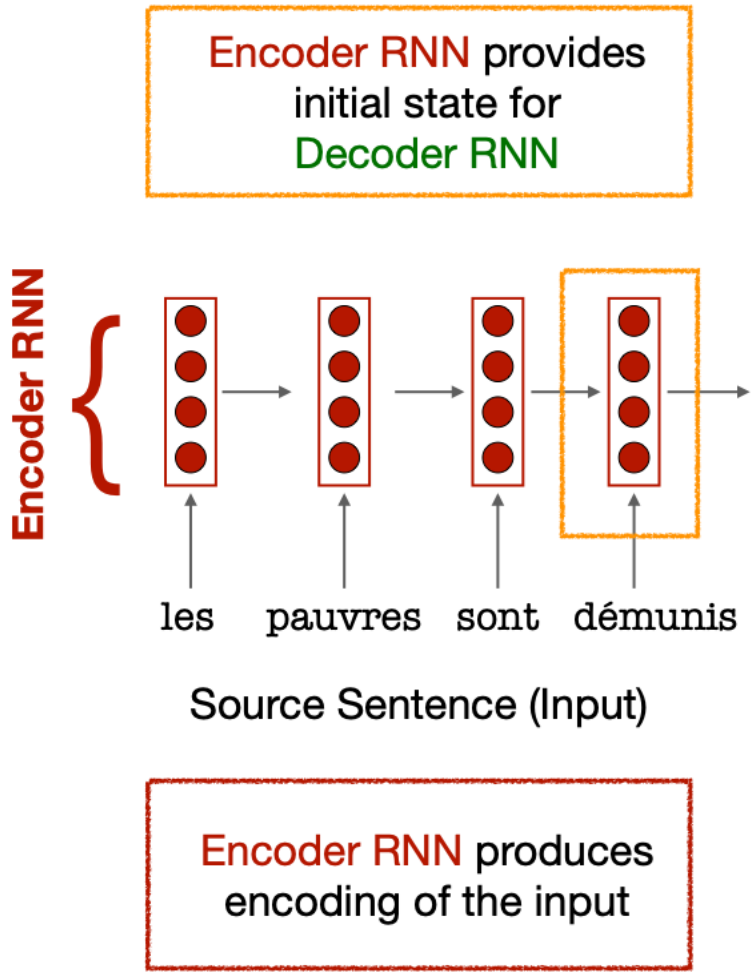


ELMo Word Embeddings

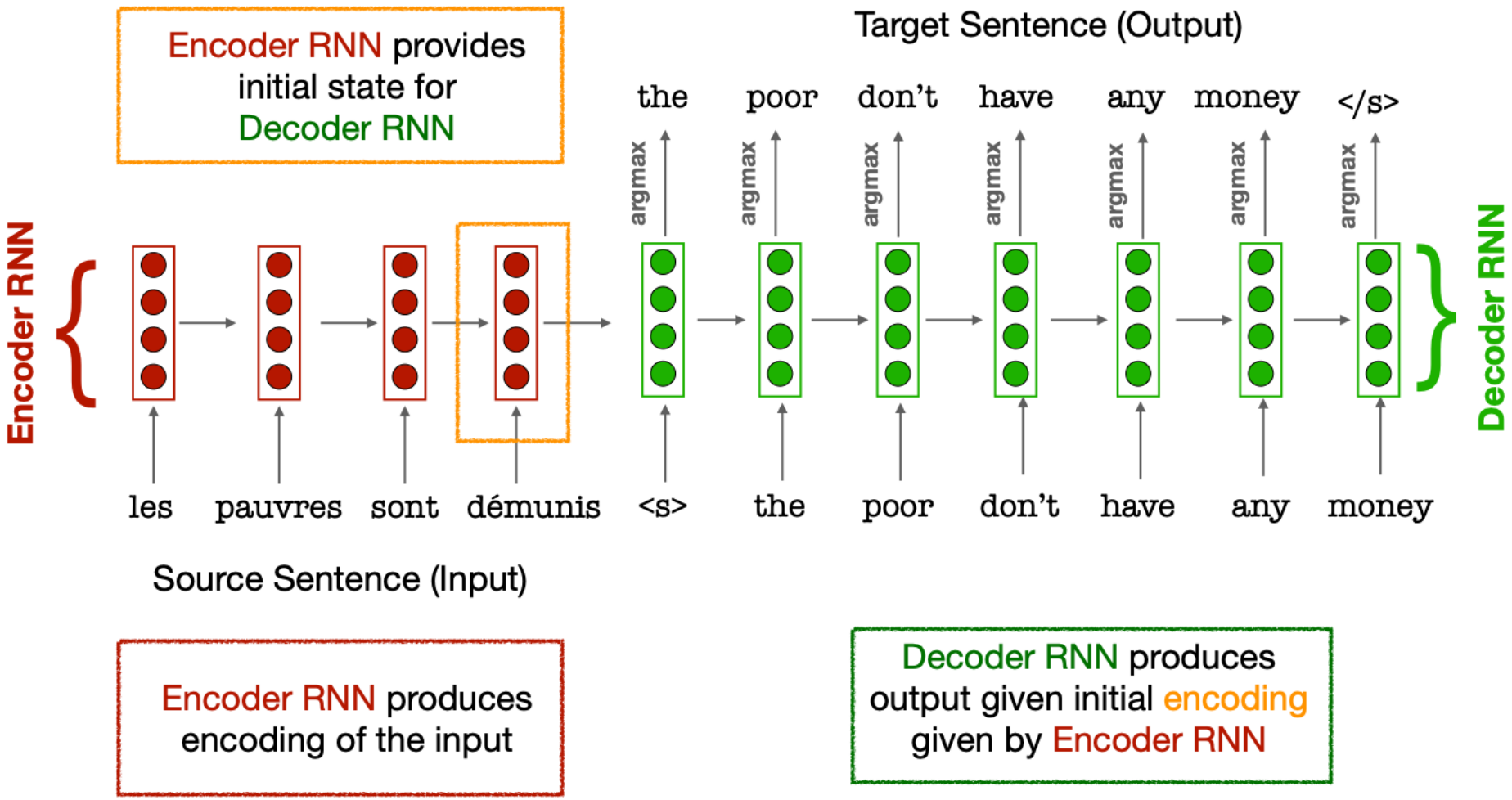
- **Bidirectional LSTM:** Combine one LSTM to predict next word given previous words, another to predict previous word given later words



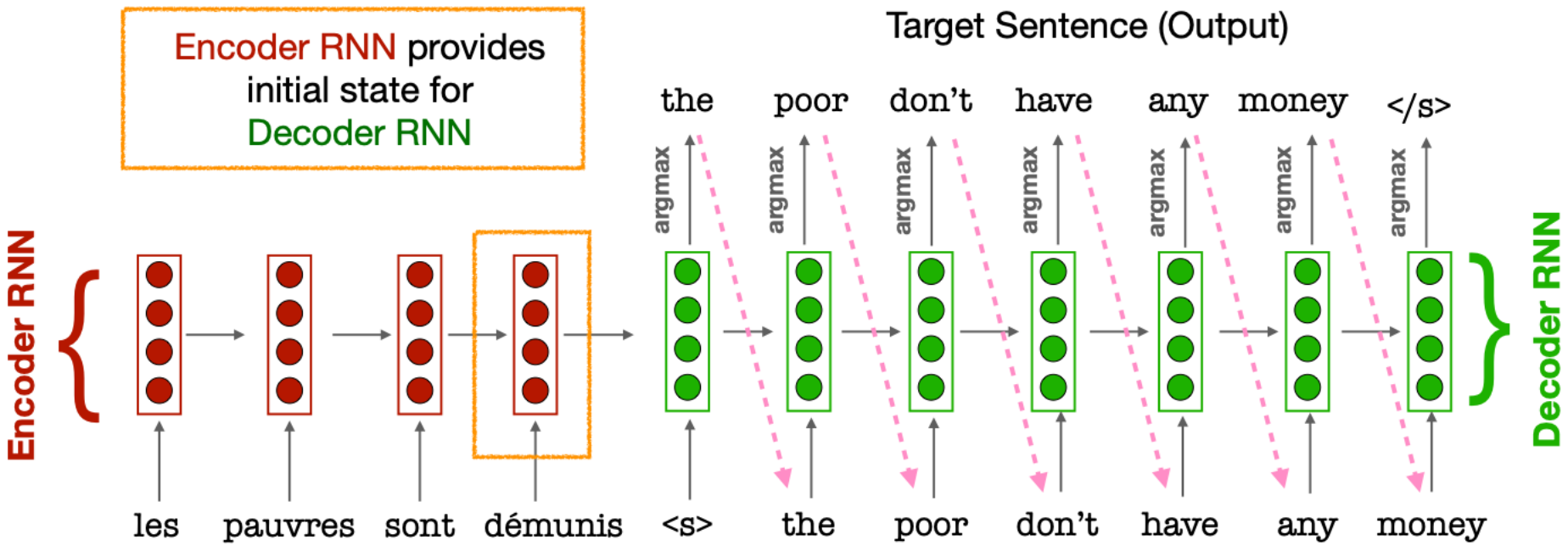
Sequence to Sequence Model — RNNs for String-to-String Problems (i.e Translation)



Sequence to Sequence Model — RNNs for String-to-String Problems (i.e Translation)



Sequence to Sequence Model — RNNs for String-to-String Problems (i.e Translation)

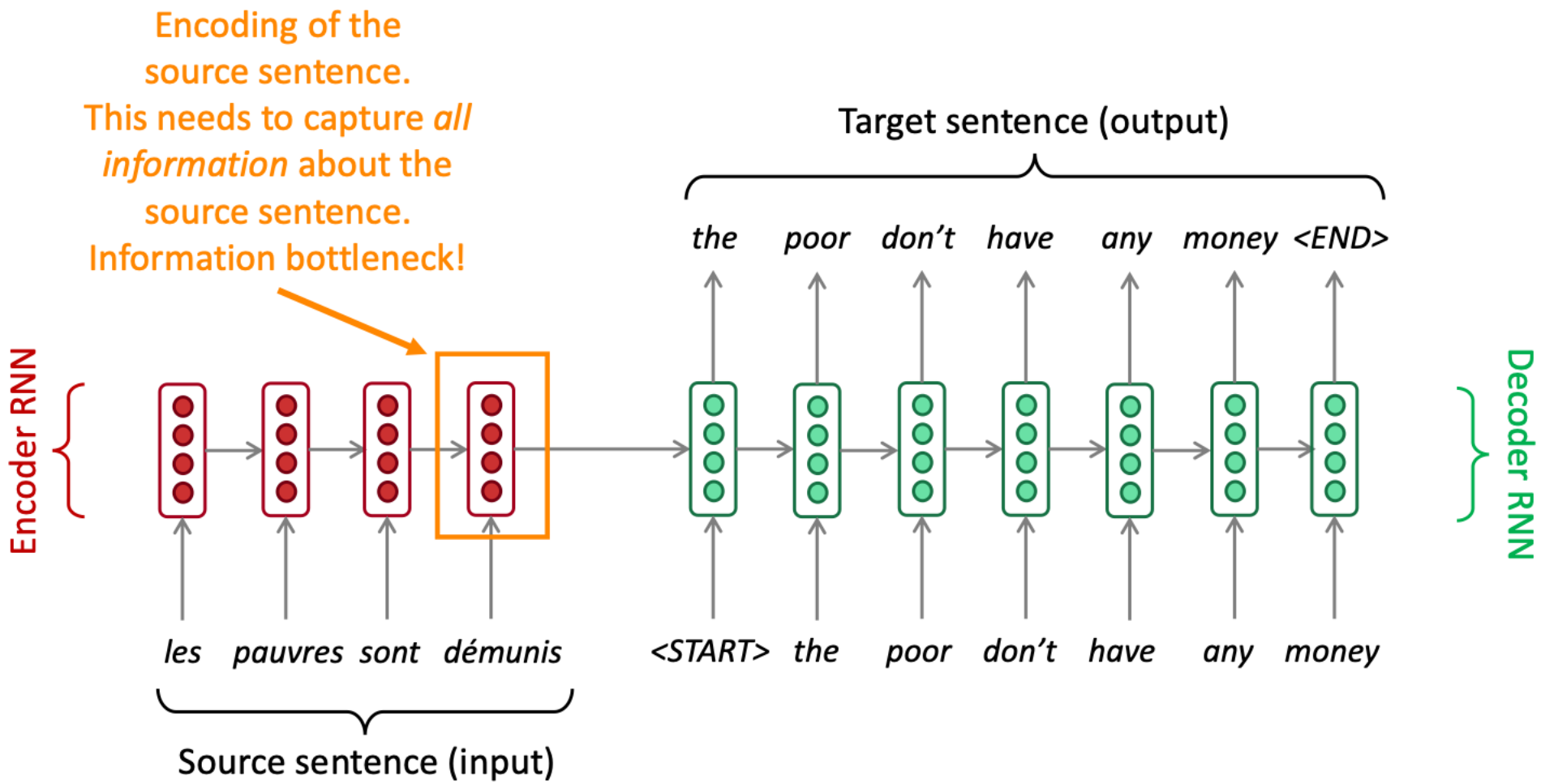


Problems that Can Be Written as String to String Transformation

- 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)

Model had huge impact on many structured problems

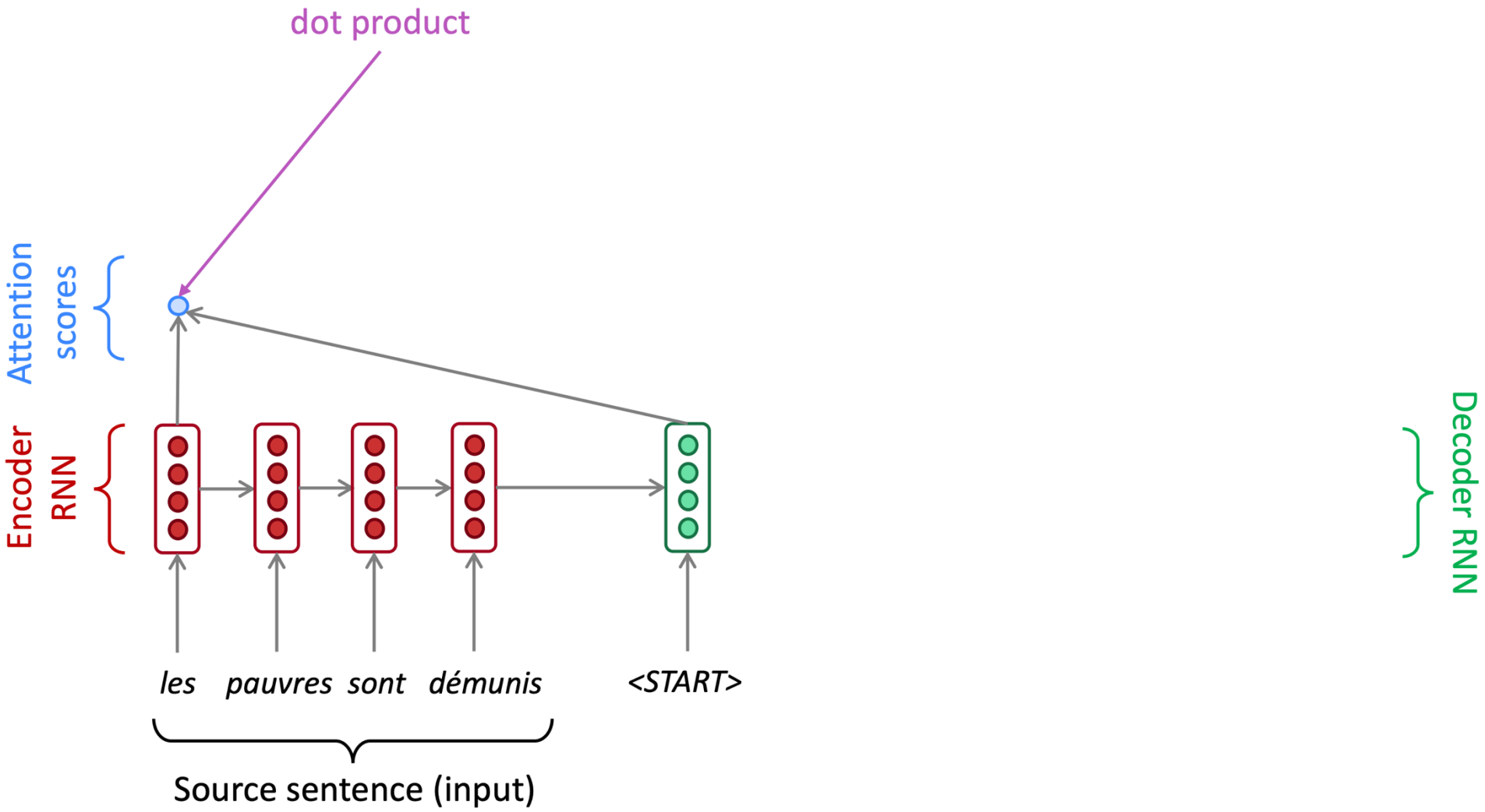
Sequence-to-sequence: the bottleneck problem



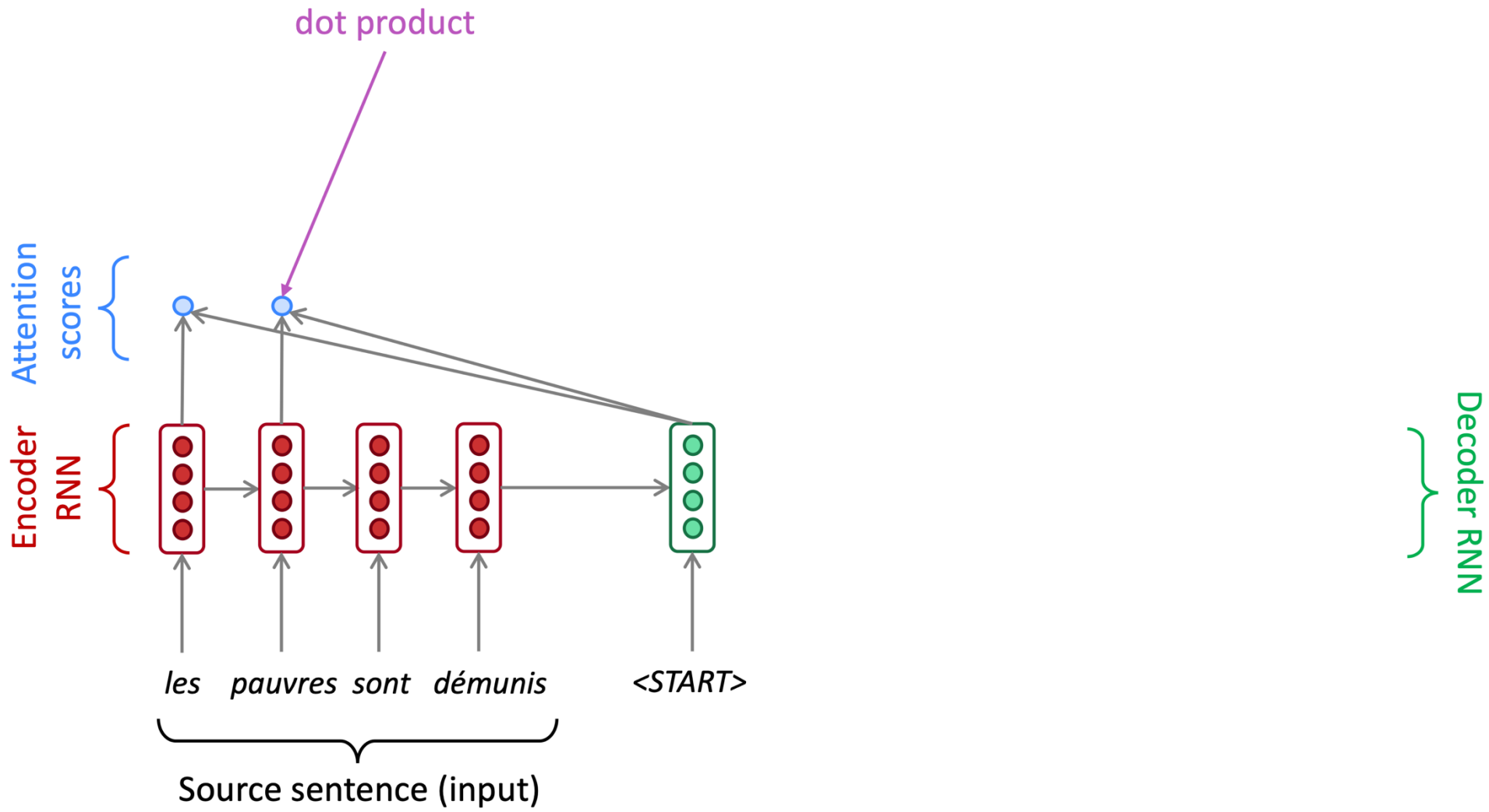
Attention

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

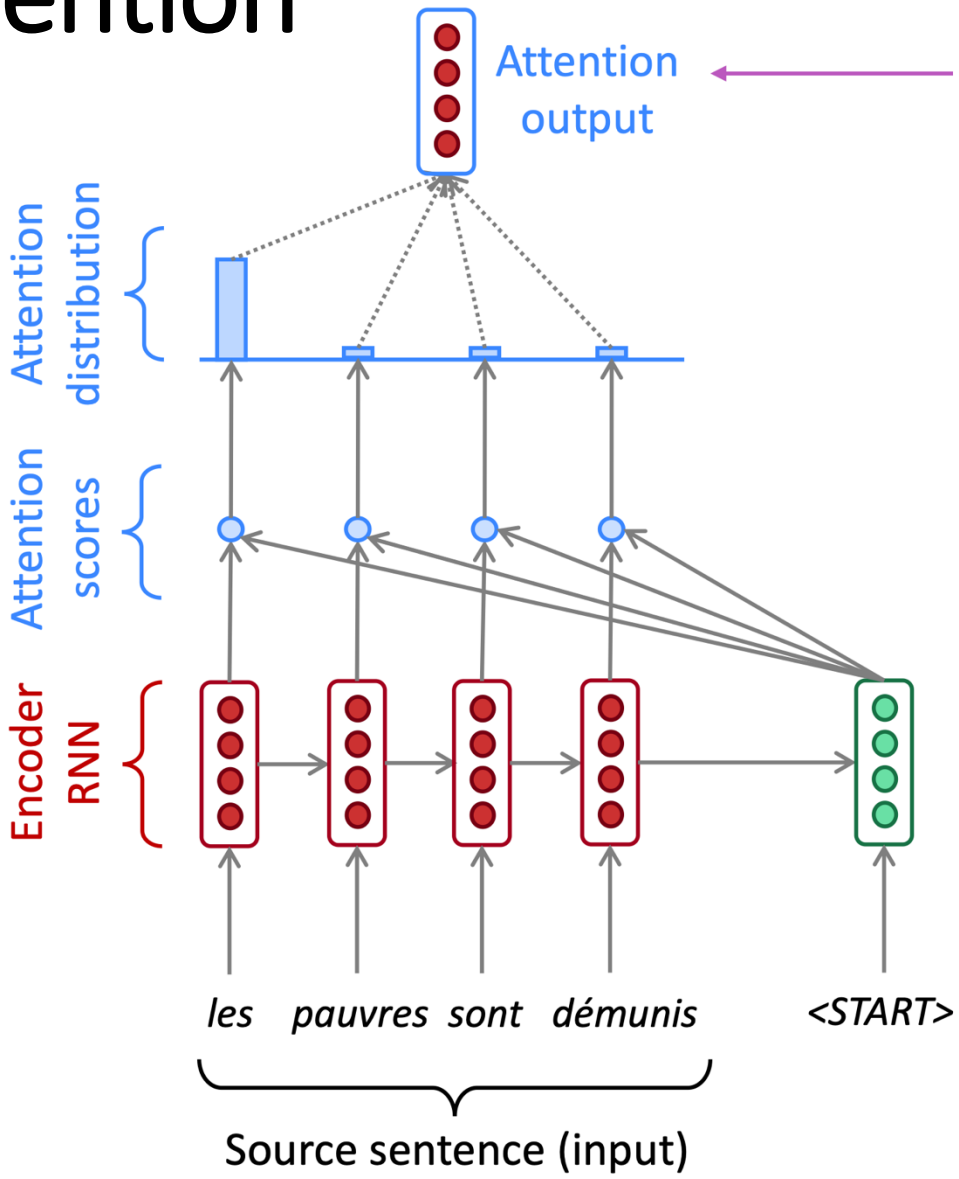
Attention


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

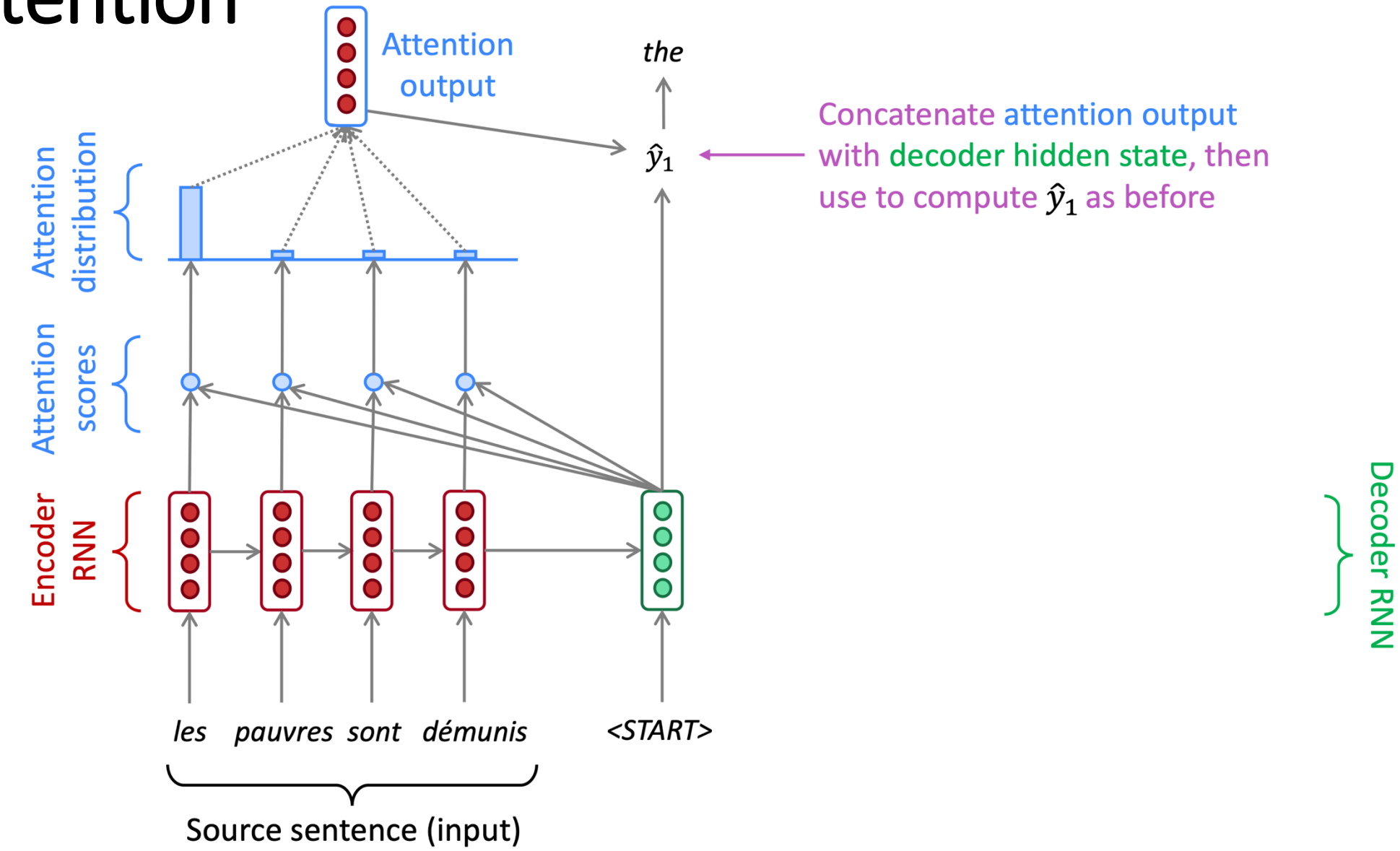


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

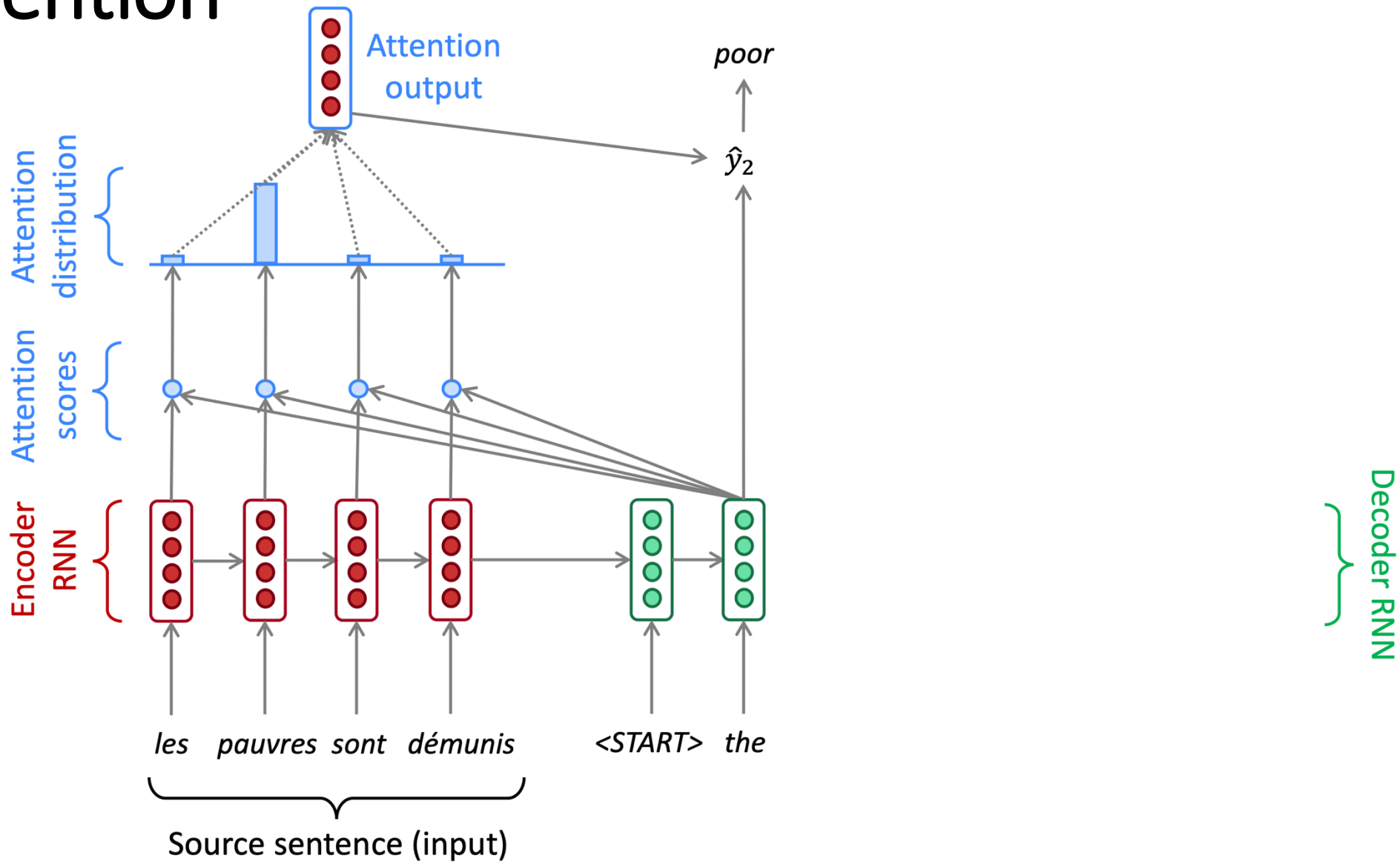
The attention output mostly contains information the hidden states that received high attention.

Decoder RNN

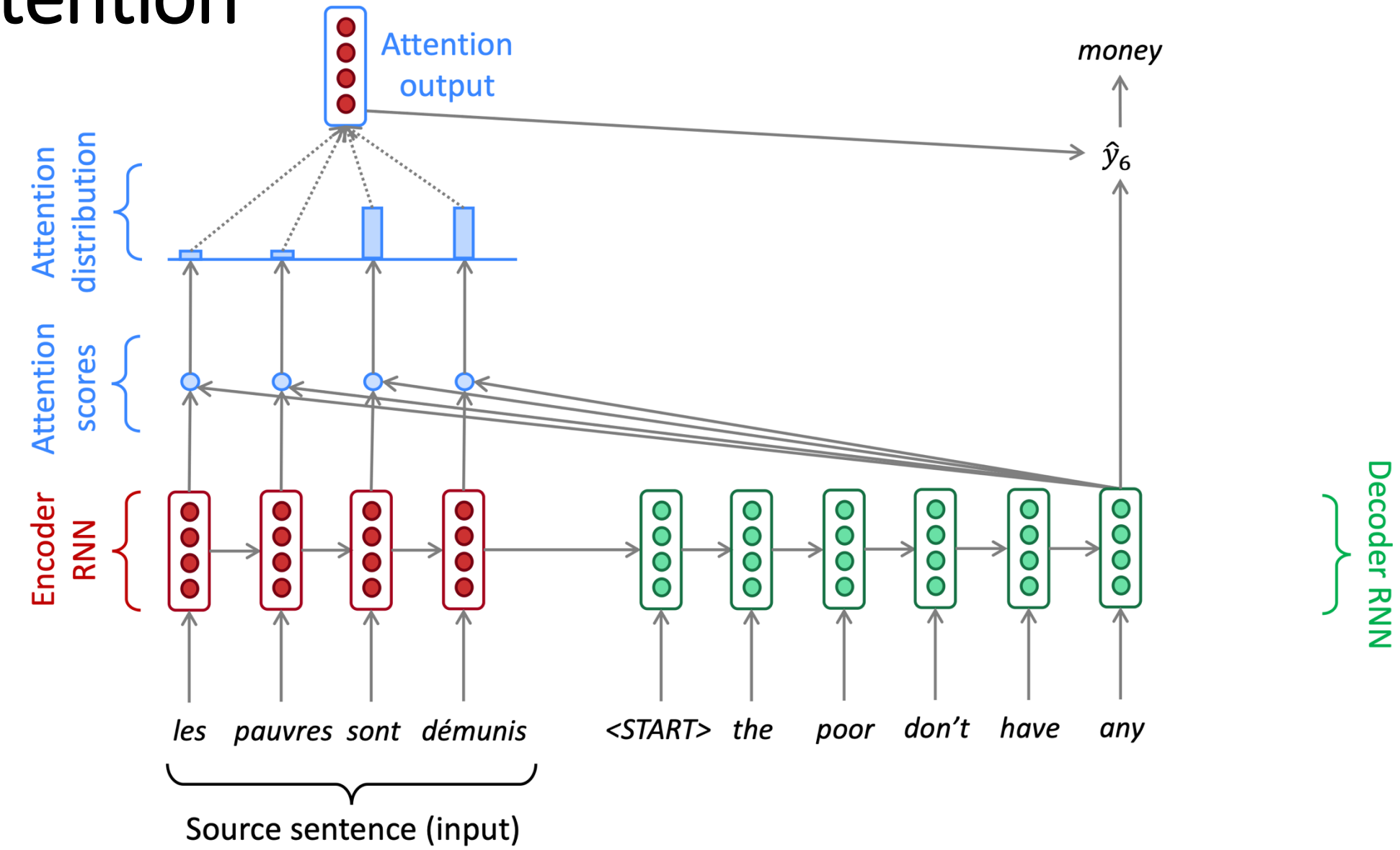
Attention


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 Equations

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores e^t for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

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

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

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

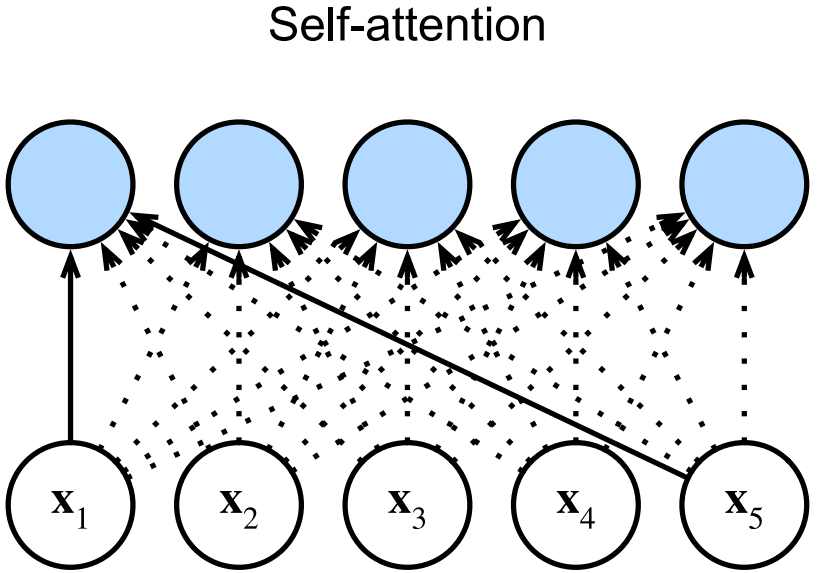
- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

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

Transformers

- Composition of **self-attention layers**

- **Intuition**
  - Want sparse connection structure of CNNs
  - 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

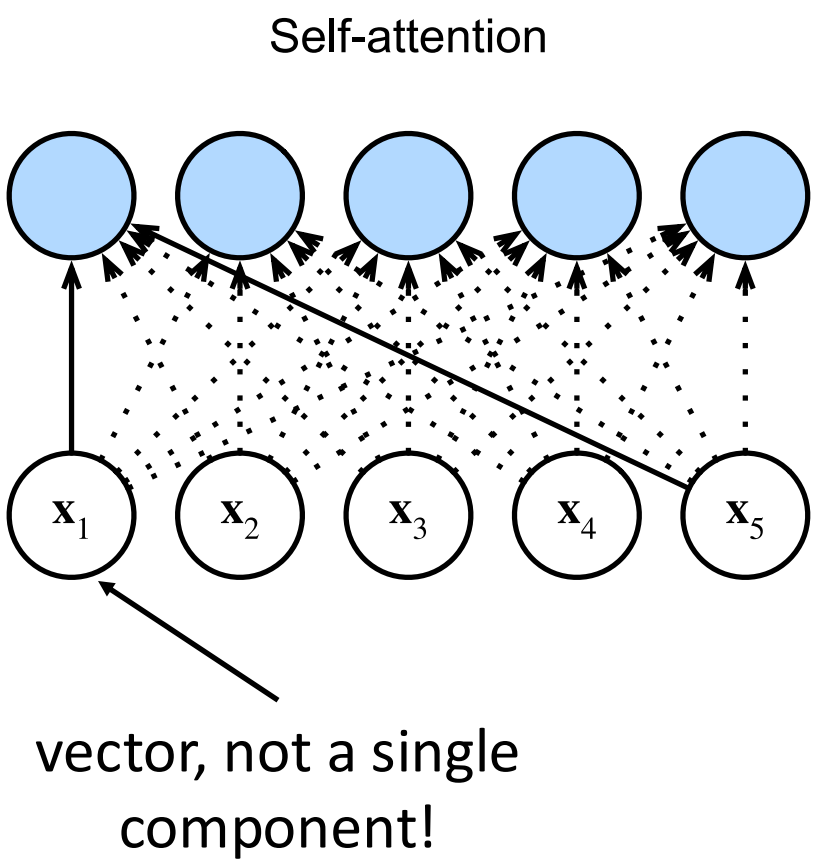


Figure credit to d2l.ai

Self-Attention Layer

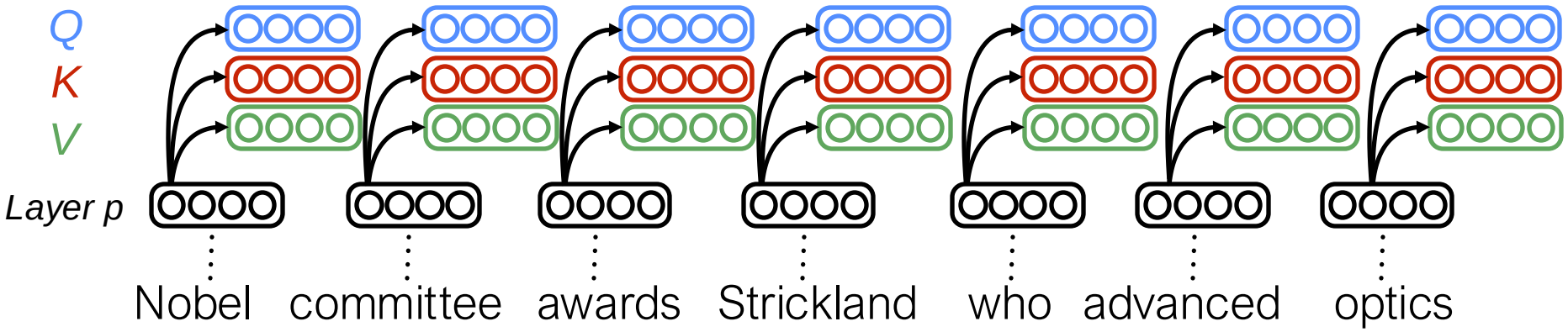
- **Self-attention layer:**

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

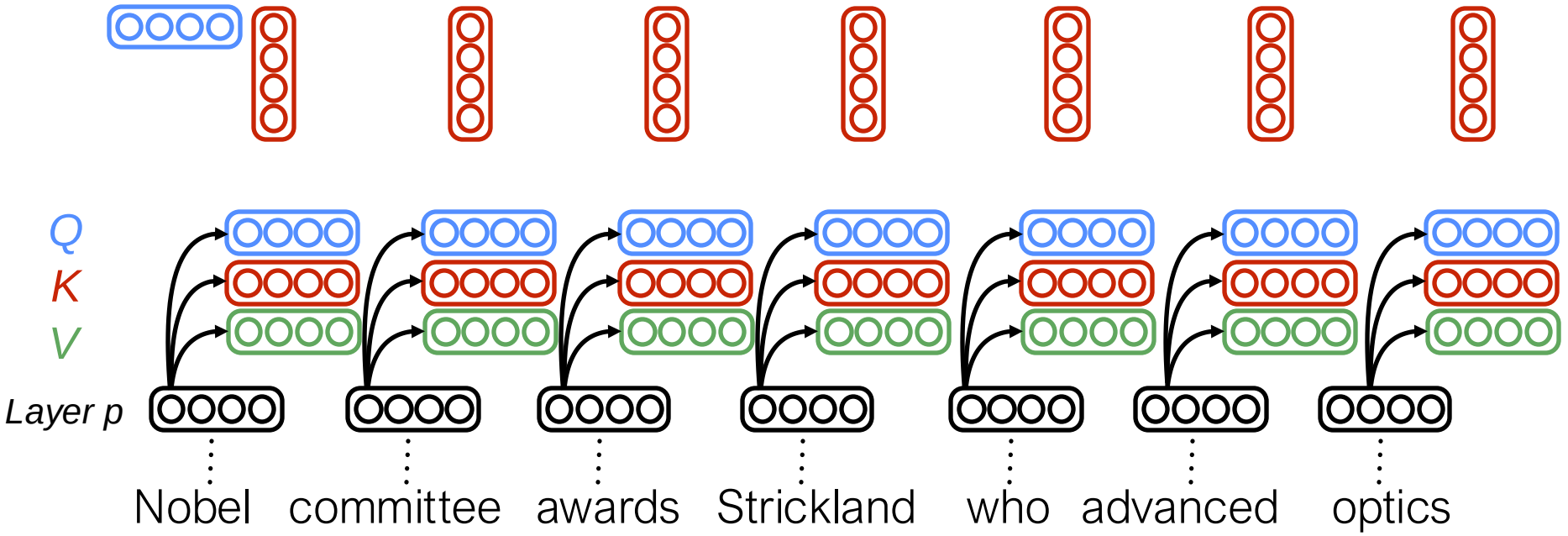
- Here, we have (learnable parameters are W_Q, W_K, and W_V):

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

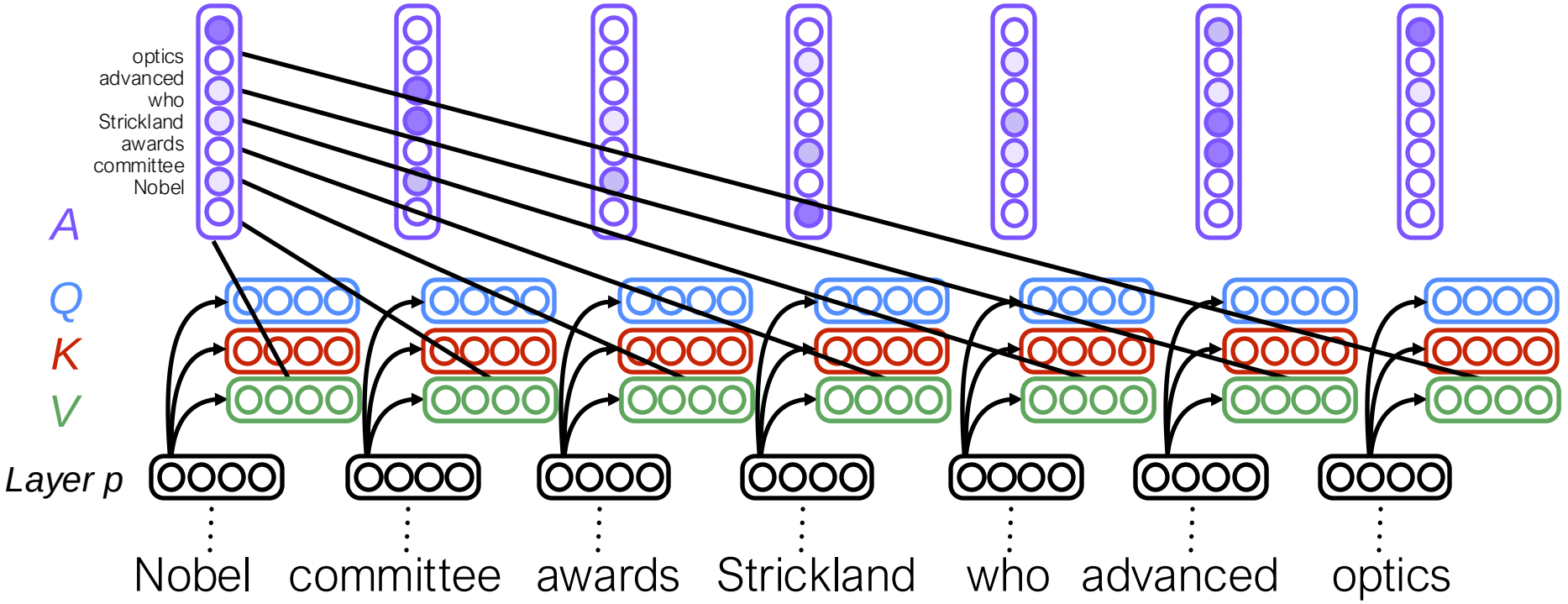
Self-Attention Layer



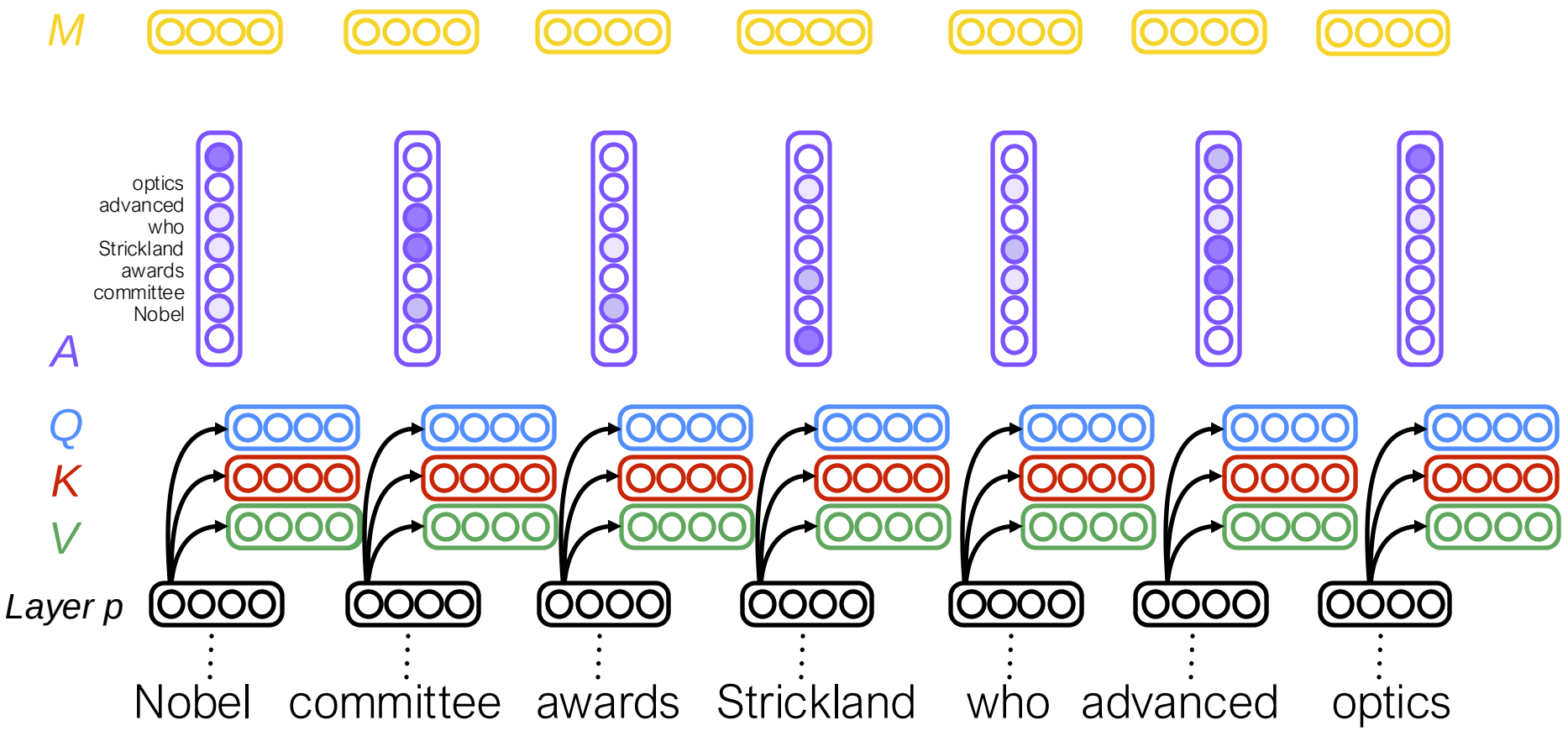
Self-Attention Layer



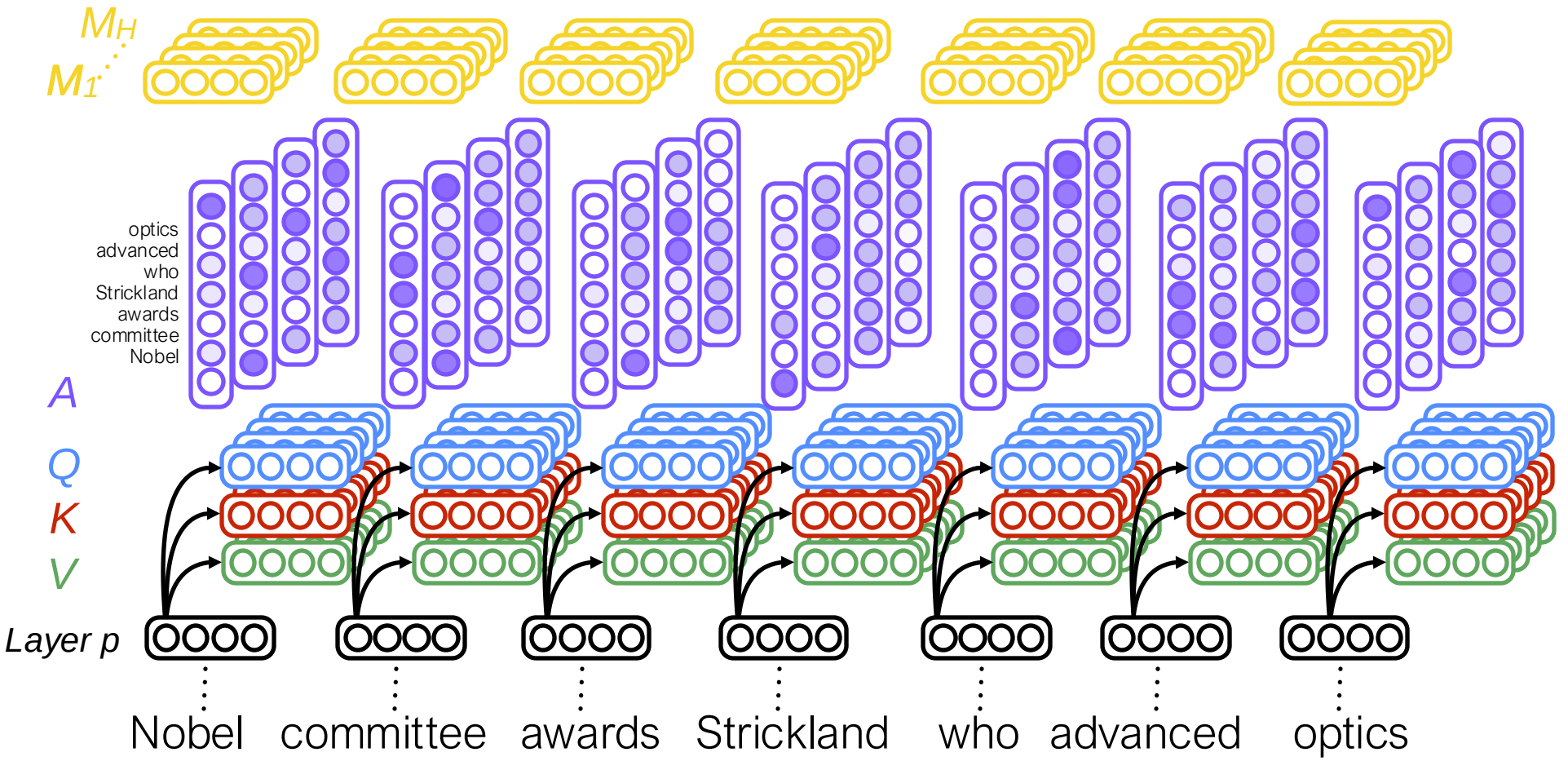
Self-Attention Layer



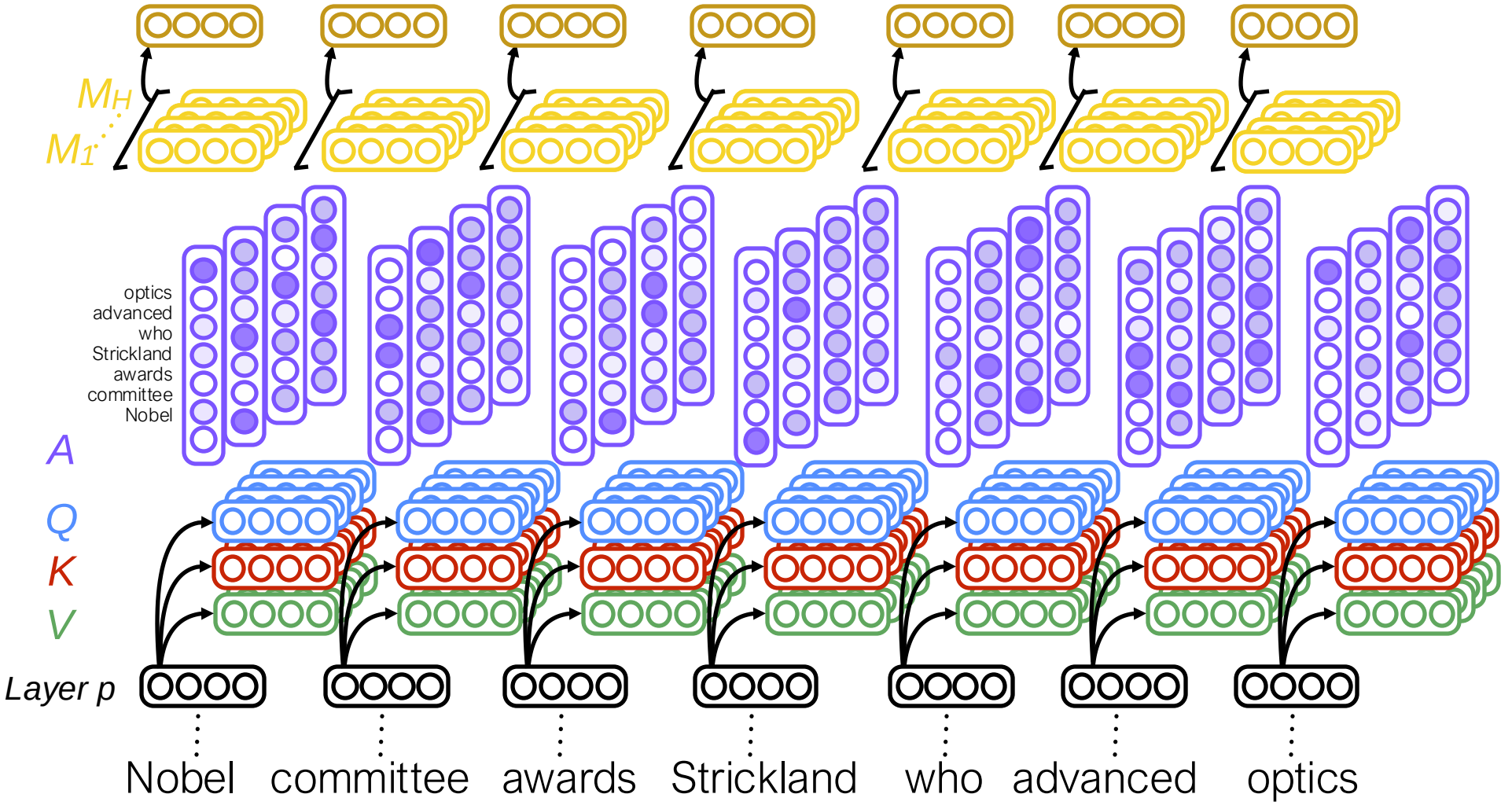
Self-Attention Layer



Multi-Head Self-Attention



Multi-Head Self-Attention



Transformers

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

- **Examples:**
  - **BERT:** Bidirectional transformer, 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

As aliens entered our planet and began to colonized Earth, a certain group of extraterrestrials began to manipulate our society through their influences of a certain number of the elite to keep and iron grip over the populace.

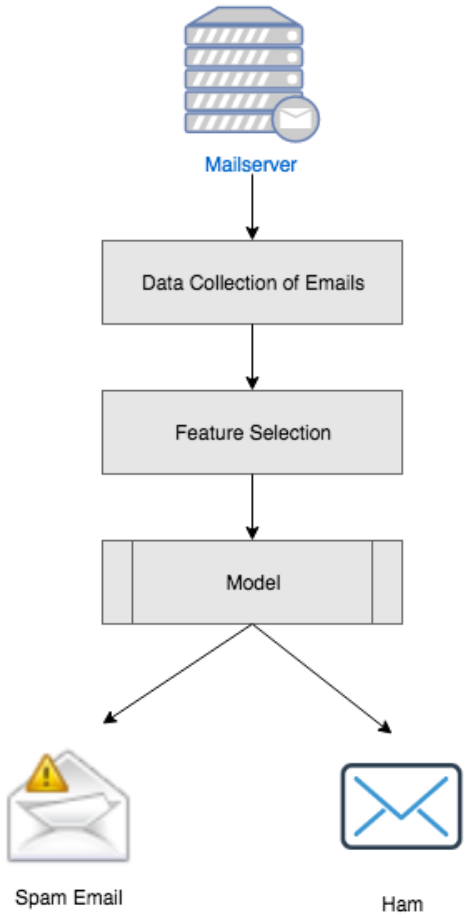
Share screenshot

https://transformer.huggingface.co/

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

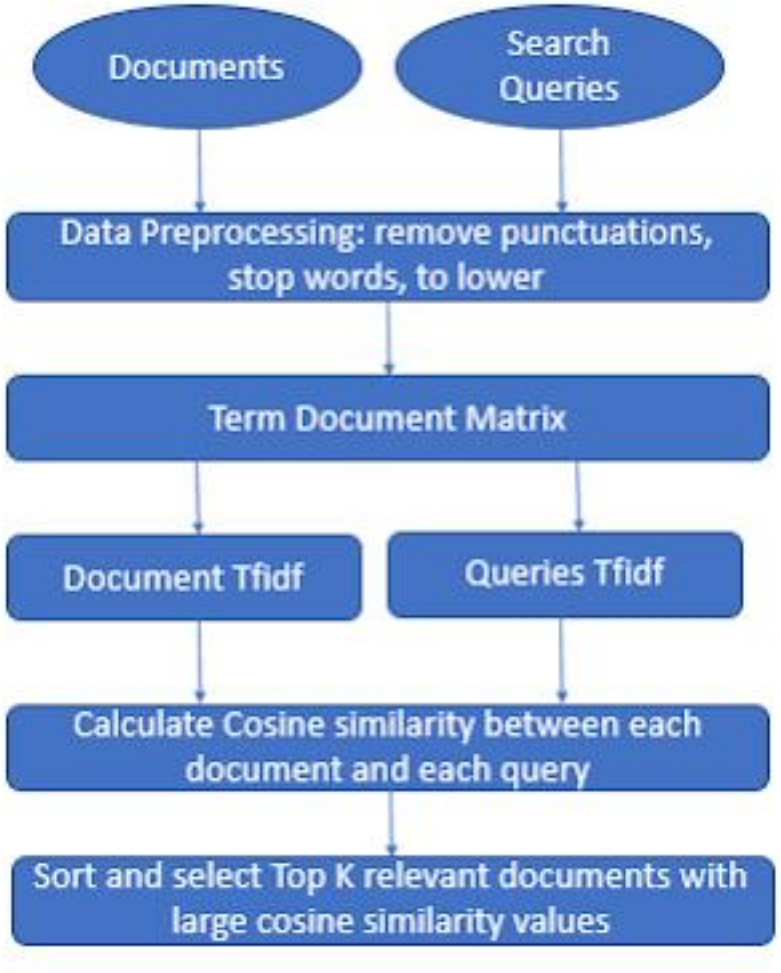
Applications: Spam Detection

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



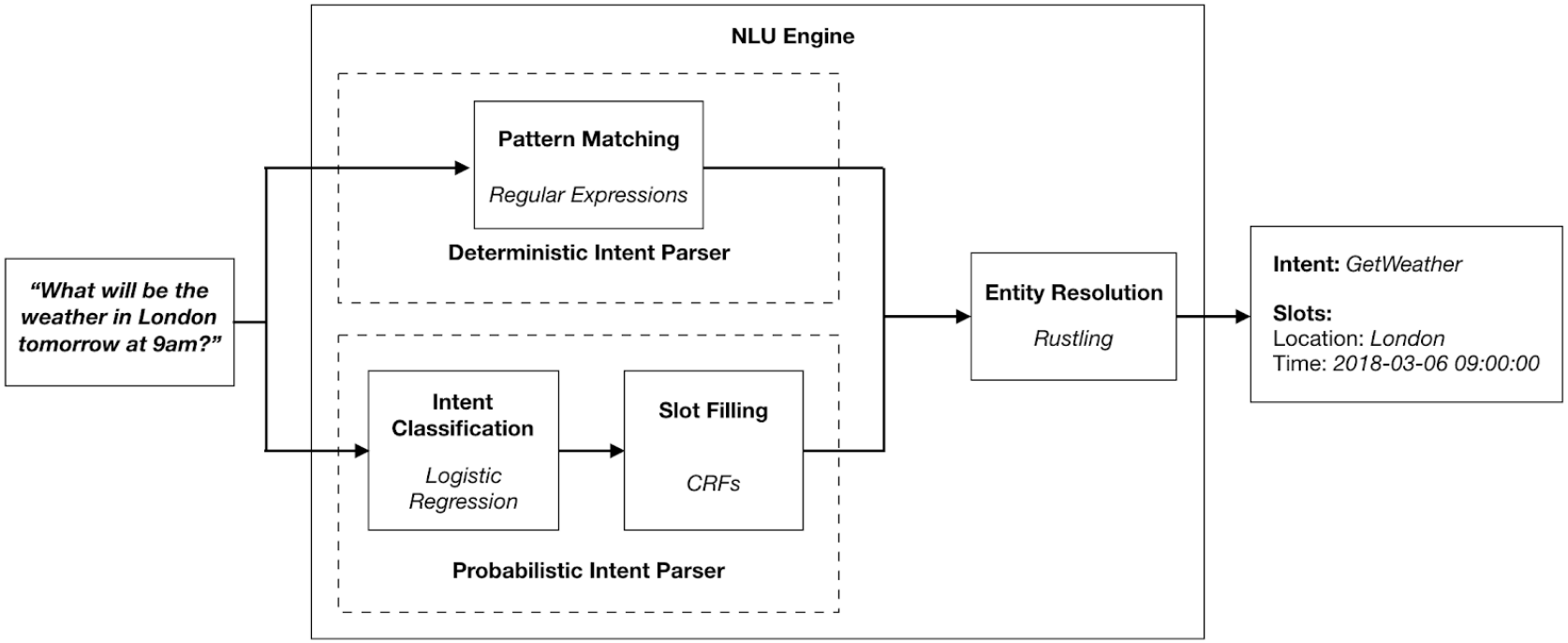
Applications: Search

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



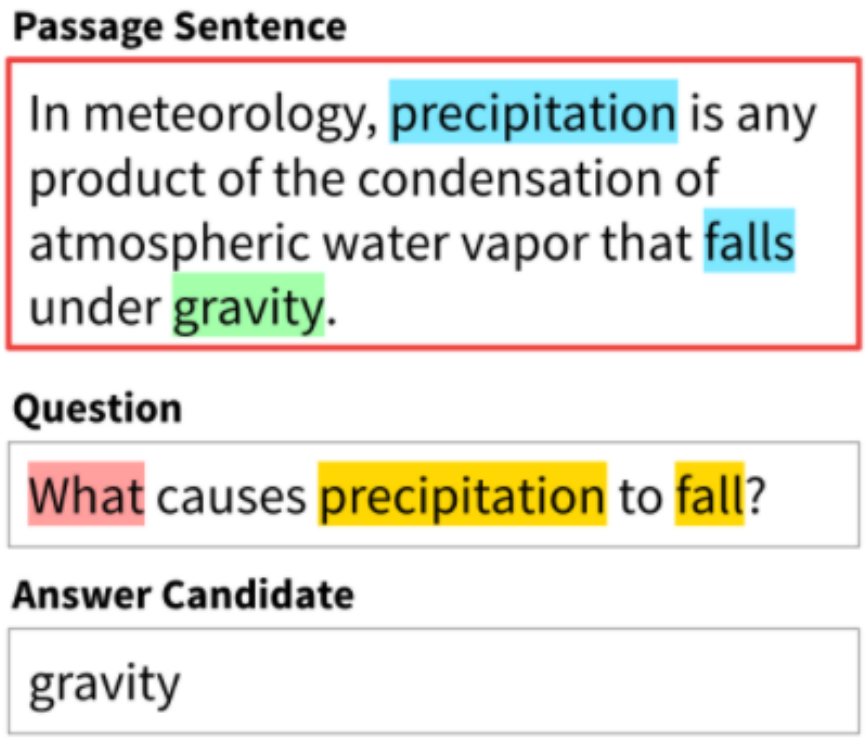
Applications: Virtual Assistants

- 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



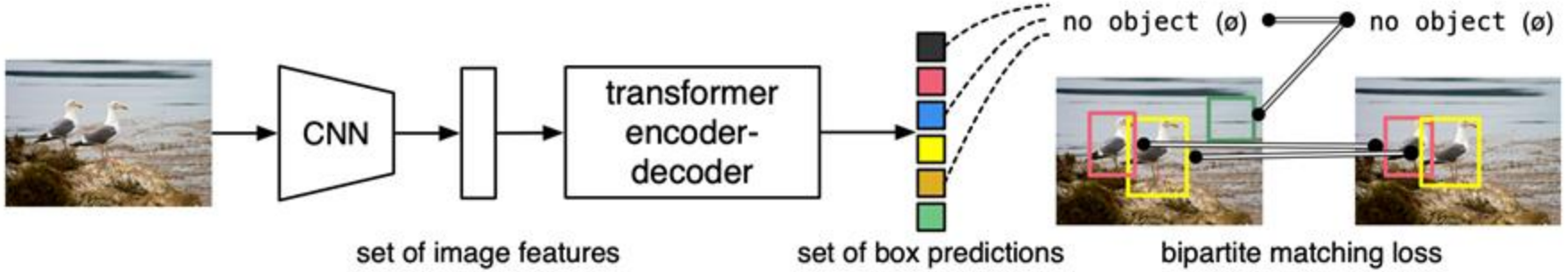
Applications: Generation

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

AI Dungeon, 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



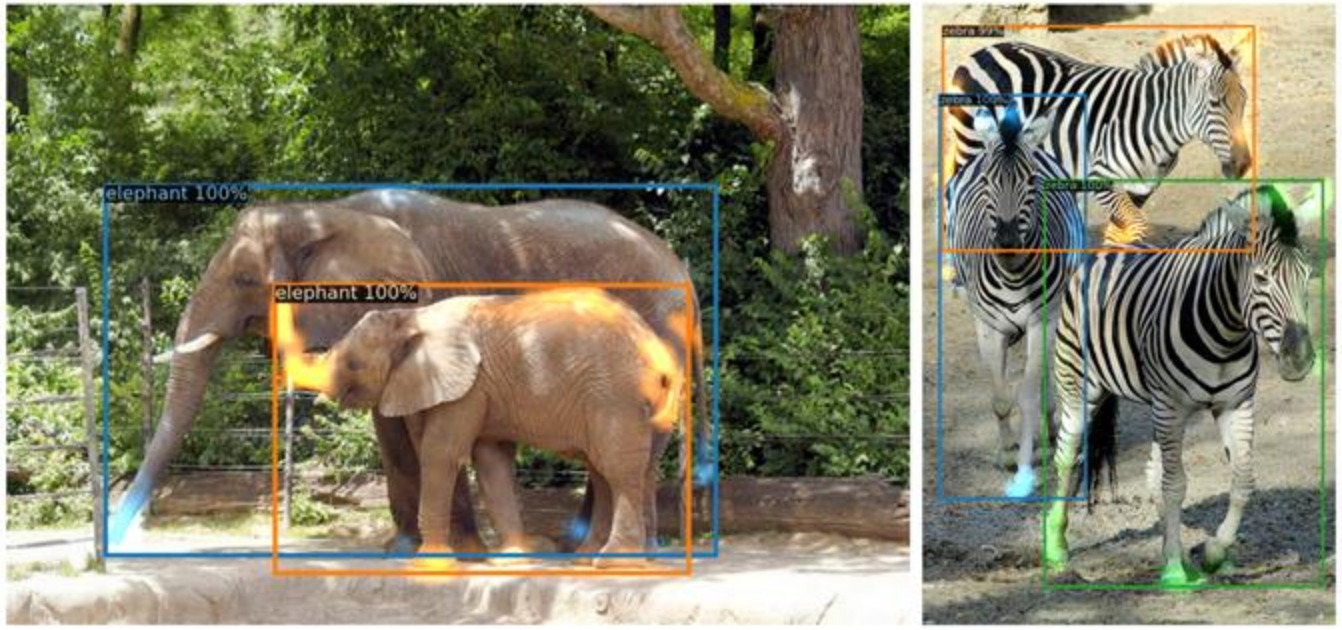


Figure credit to "End-to-End Object Detection with Transformers"