CS60075
Natural Language Processing
Autumn 2020

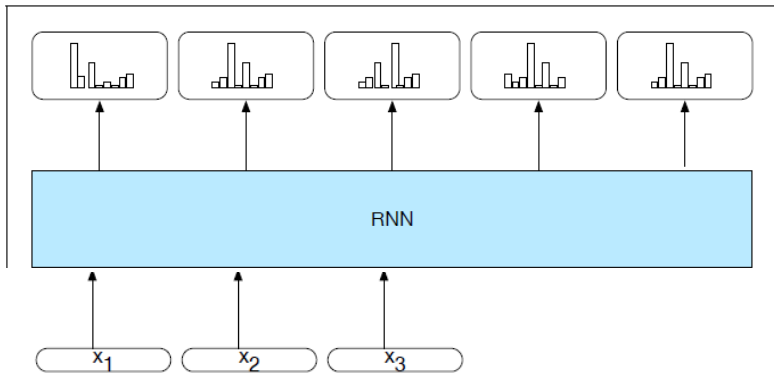
Module 6:

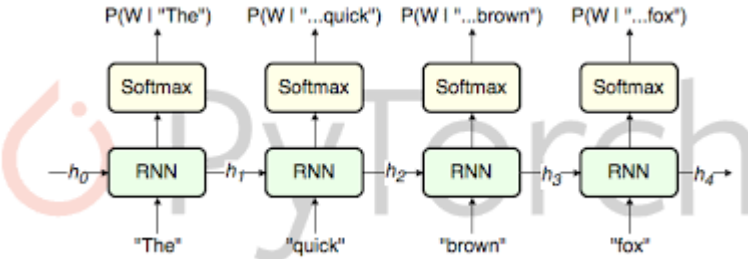
RNN Part 2

14 October 2020

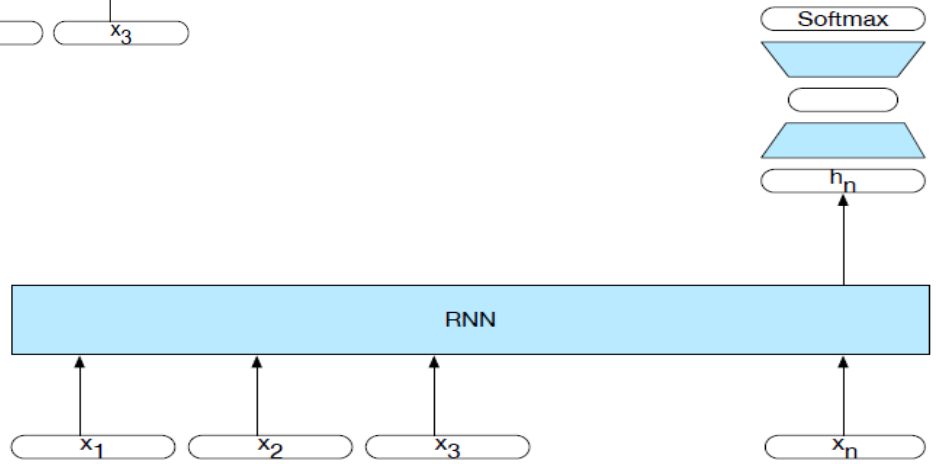
RNN Applications

- Language Modeling

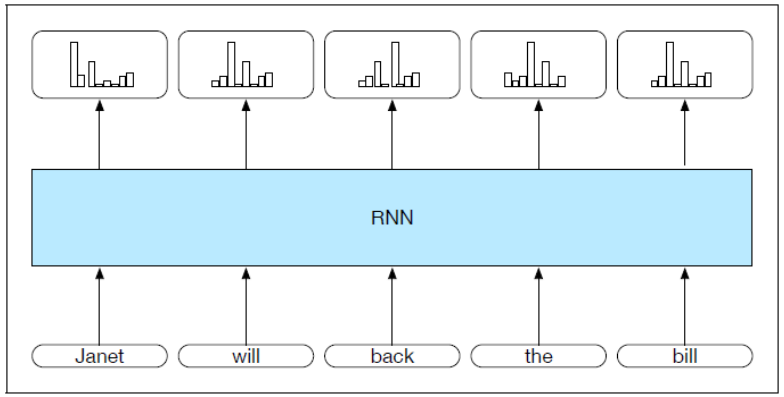




- Sequence Classification (Sentiment, Topic)



- Sequence to Sequence



Generating text with neural language models

Model Completion (Machine-Written, 10 Tries): The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic."

Dr. Pérez believes that the unicorns may have originated in Argentina, where the animals were believed to be descendants of a lost race of people who lived there before the arrival of humans in those parts of South America.

While their origins are still unclear, some believe that perhaps the creatures were created when a human and a unicorn met each other in a time before human civilization. According to Pérez, "In South America, such incidents seem to be quite common."

However, Pérez also pointed out that it is likely that the only way of knowing for sure if unicorns are indeed the descendants of a lost alien race is through DNA. "But they seem to be able to communicate in English quite well, which I believe is a sign of evolution, or at least a change in social organization," said the scientist.

Generation with Neural Language Models

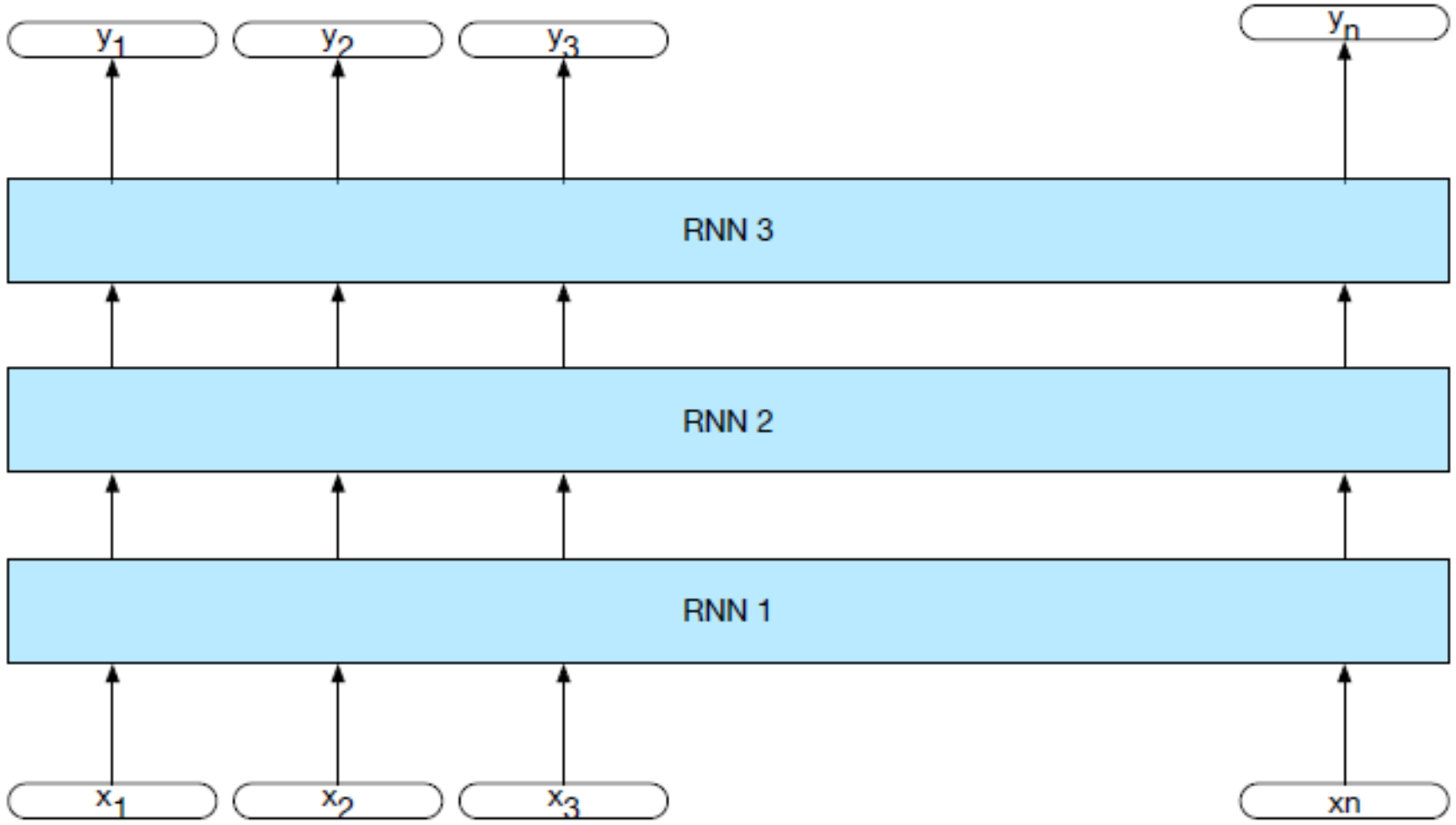
beginning of sentence marker

end of sentence marker

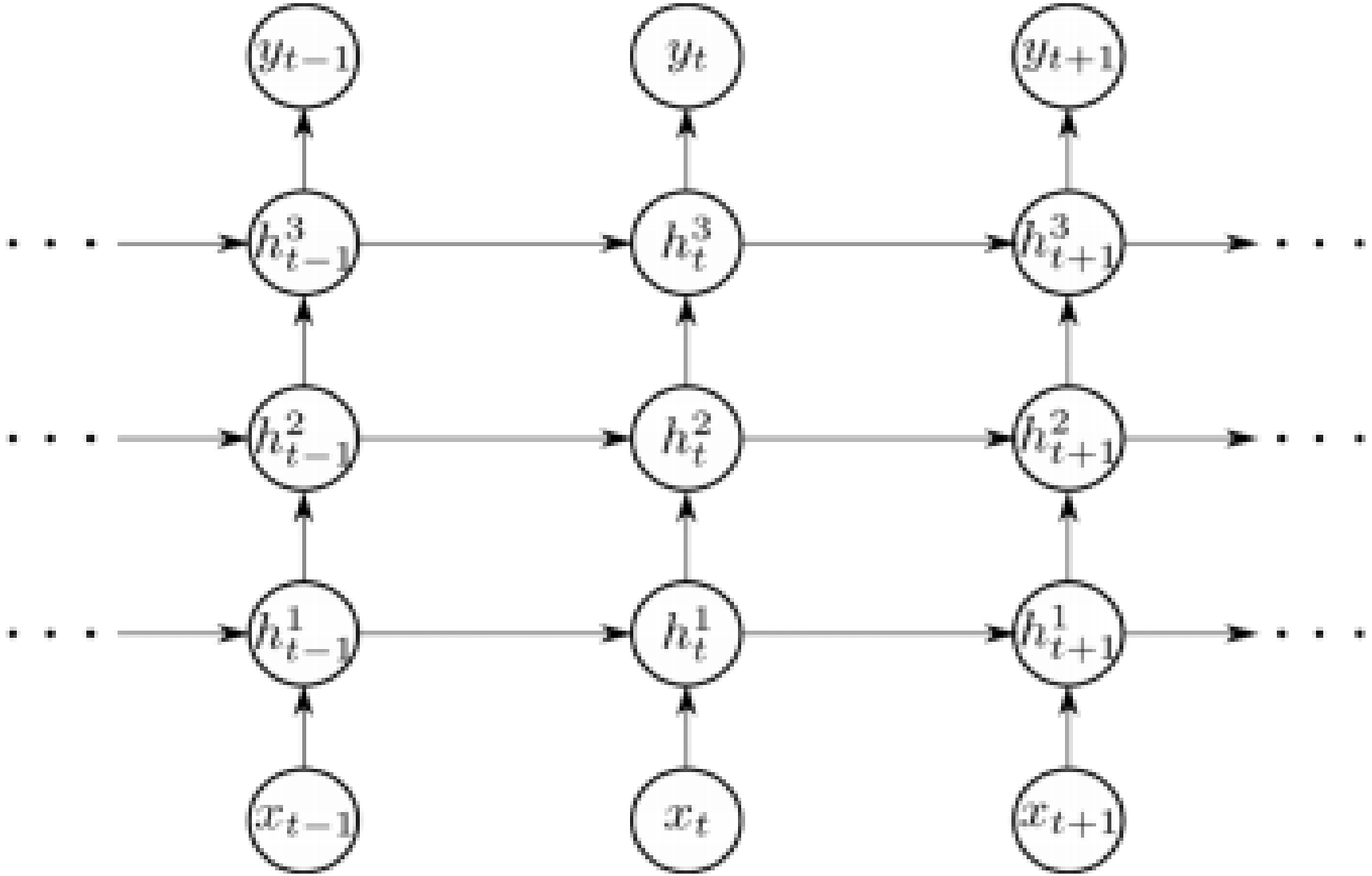
- Autoregressive Generation:
- Word generated at each timestep is conditioned on the word generated previously by the model

Deep Networks: Stacked RNNs

Use the sequence of outputs from one RNN as an input sequence to another one



Stacked RNN



Bidirectional RNNs

In SRN, h_t represents everything the network knows about the sequence up to that point in the sequence.

In text-based applications the entire input sequence is available all at once.

Train SRN on the input sequence in reverse. h'_t represents information about sequence to the right of the current.

Outputs of both networks are concatenated (or element wise addition or multiplication)

$$h_t^{forward} = \text{SRN}^{forward}(x_1 : x_t)$$

Result of a function of the inputs up to ***t***

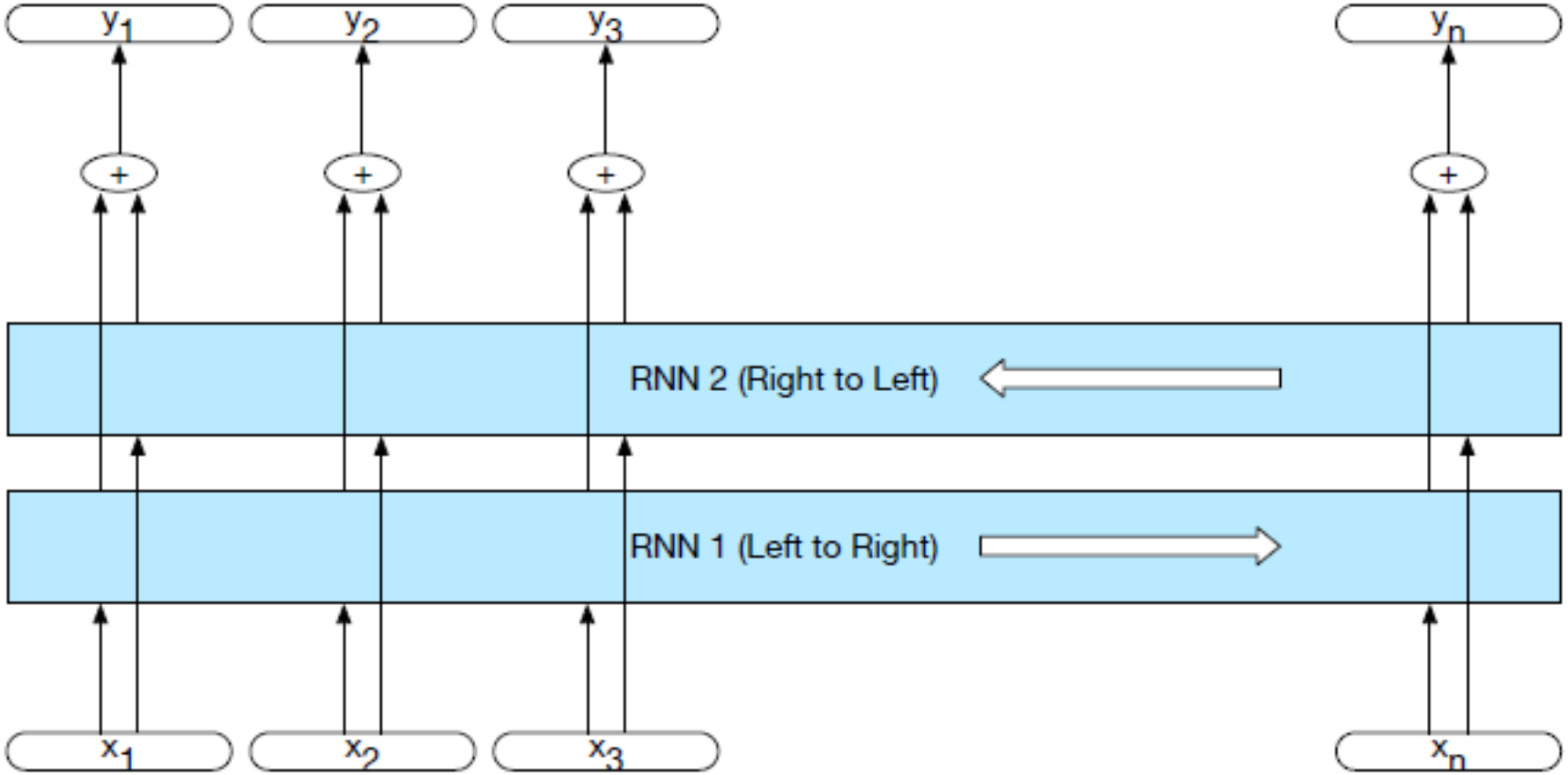
$$h_t^{backward} = \text{SRN}^{backward}(x_n : x_t)$$

Result of a function of the inputs to right

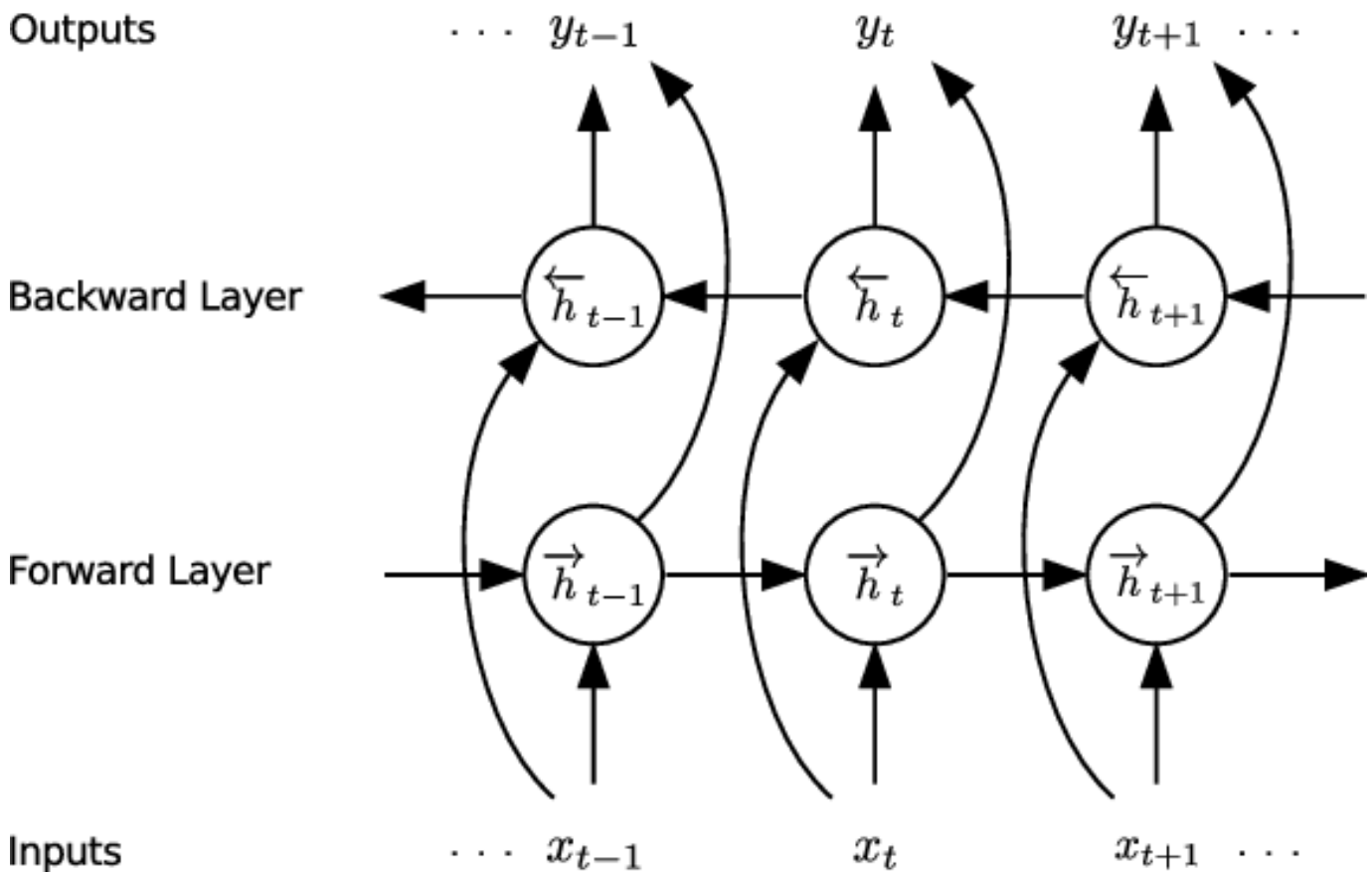
$$h_t = [h_t^{forward}; h_t^{forward}]$$

Bidirectional RNNs

- Consists of **two independent RNNs**
- outputs of the two networks are combined to capture **both the left and right contexts** of an input at each point in time.

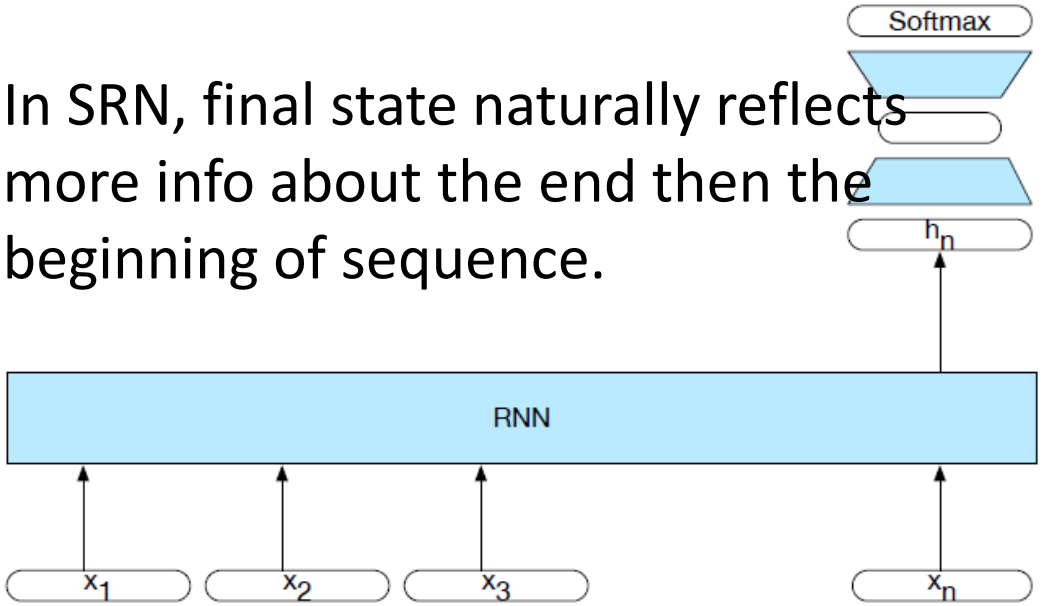


Bidirectional RNN



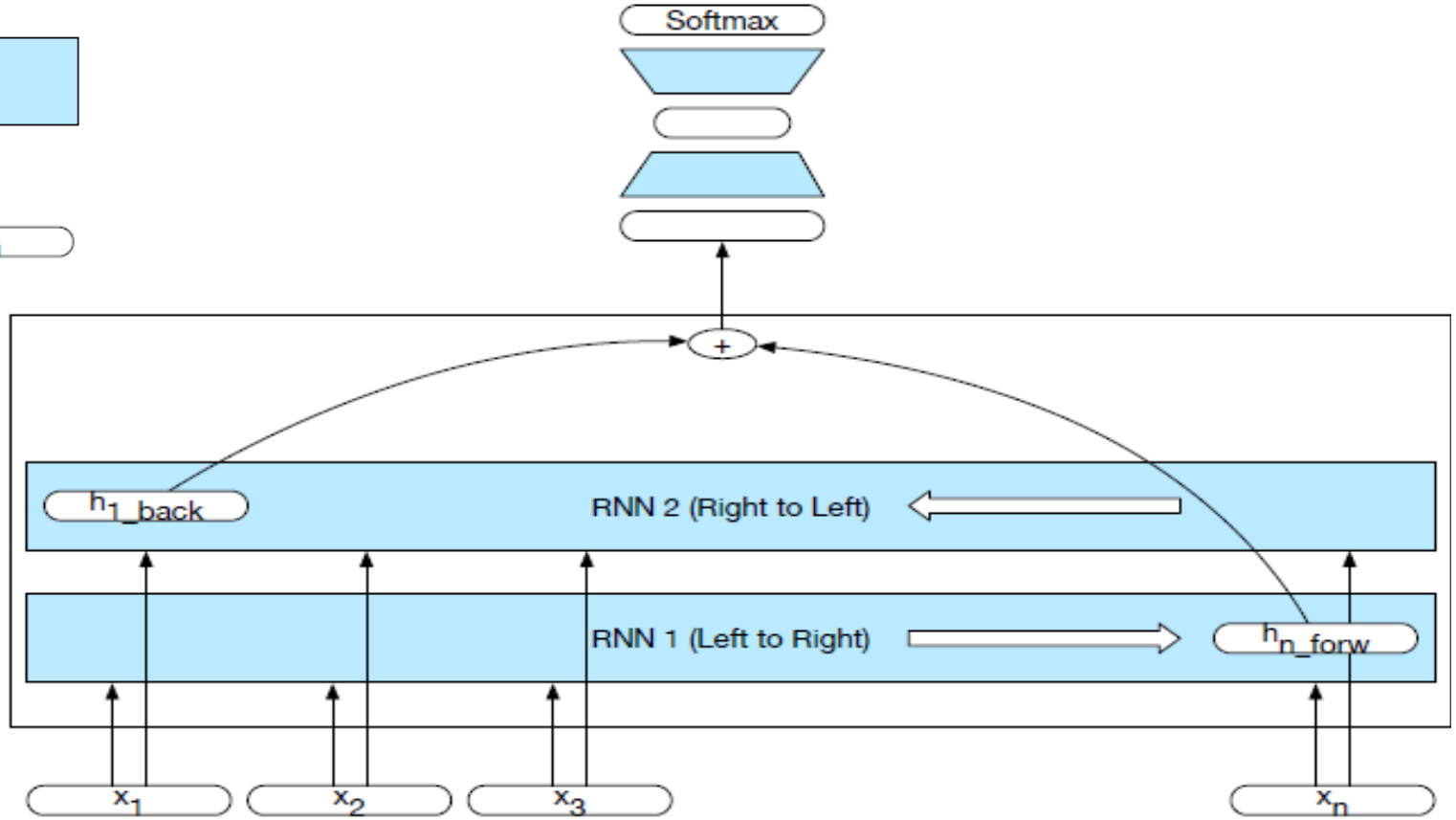
Bidirectional RNNs for Sequence Classification

In SRN, final state naturally reflects more info about the end then the beginning of sequence.

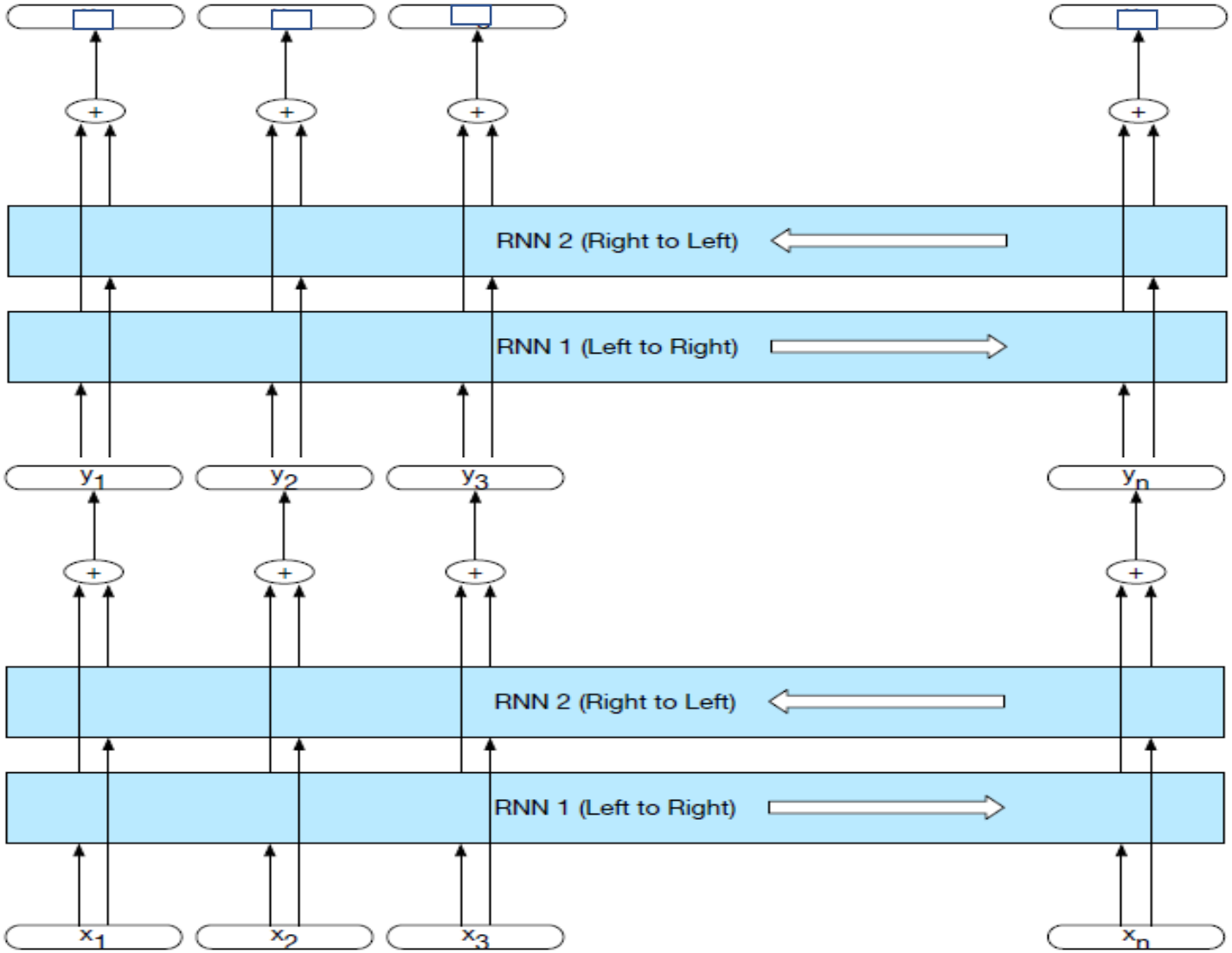


Classifier Input $= [h_{\boldsymbol{n}}^{forward}; h_{\boldsymbol{1}}^{backward}]$

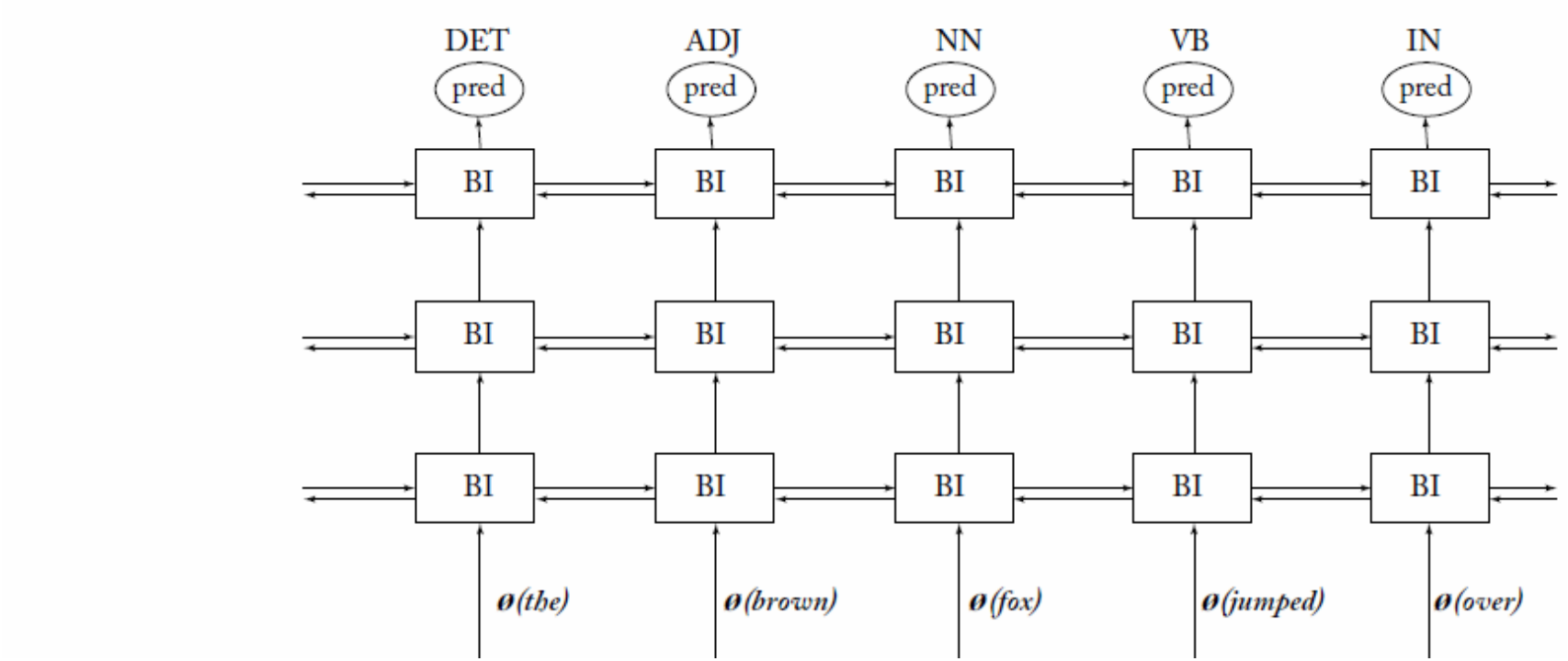
Bidirection RNNs provide a simple solution !



Bidirectional can be stacked

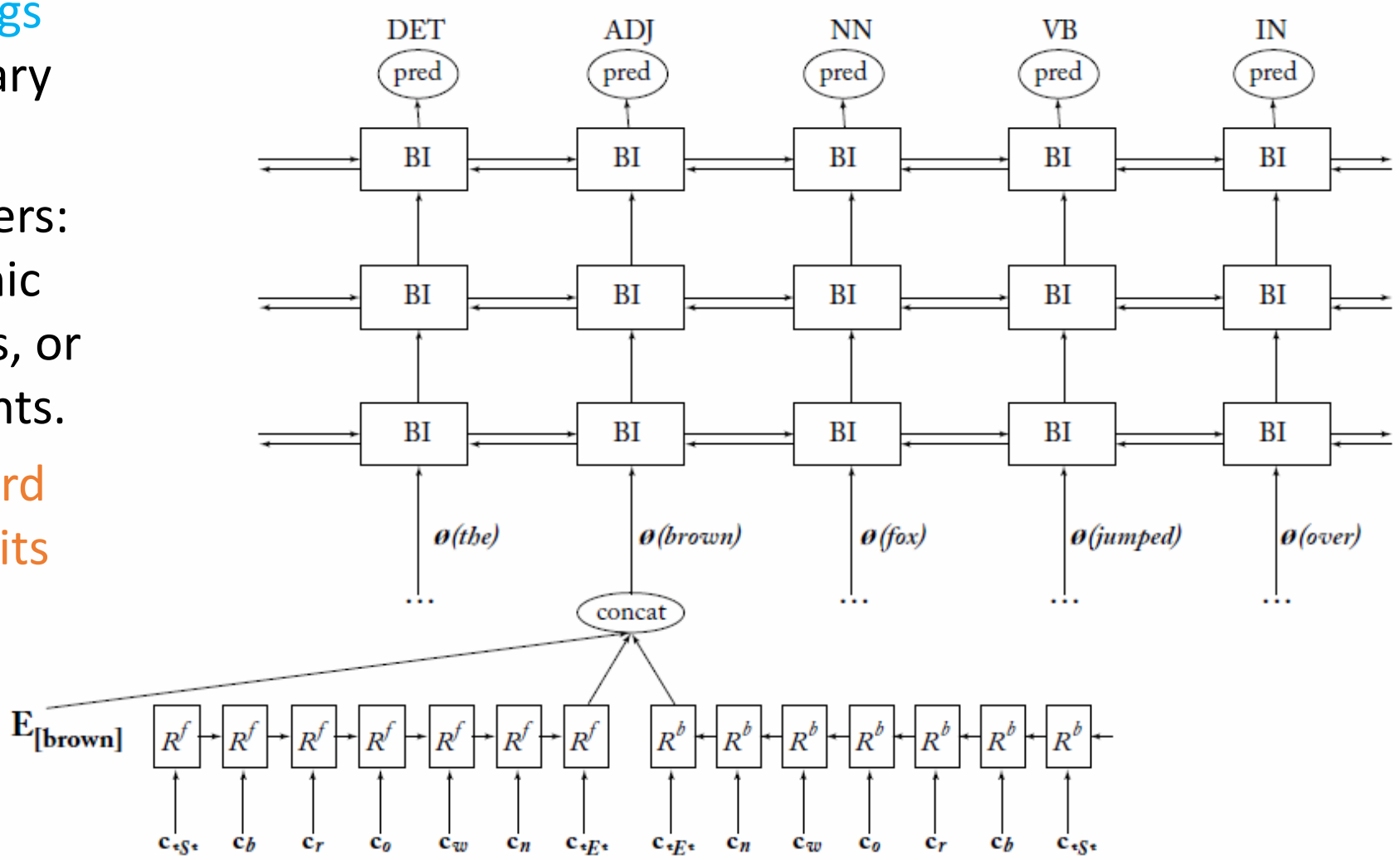


Example of stacked bi-RNN for POS tagging



Example of stacked bi-RNN for POS tagging

- Go beyond word embeddings (which suffer from vocabulary coverage)
- Words are made of characters: suffixes prefixes orthographic cues capitalization, hyphens, or digits can provide strong hints.
- Encode word using a forward RNN and reverse RNN over its characters!



Managing Context in RNNs: LSTMs and GRUs

- Information encoded in hidden states tends to be **much more influenced by the most recent parts** of the input sequence and recent decisions
- However, **long-distance information is often critical** to many language applications.

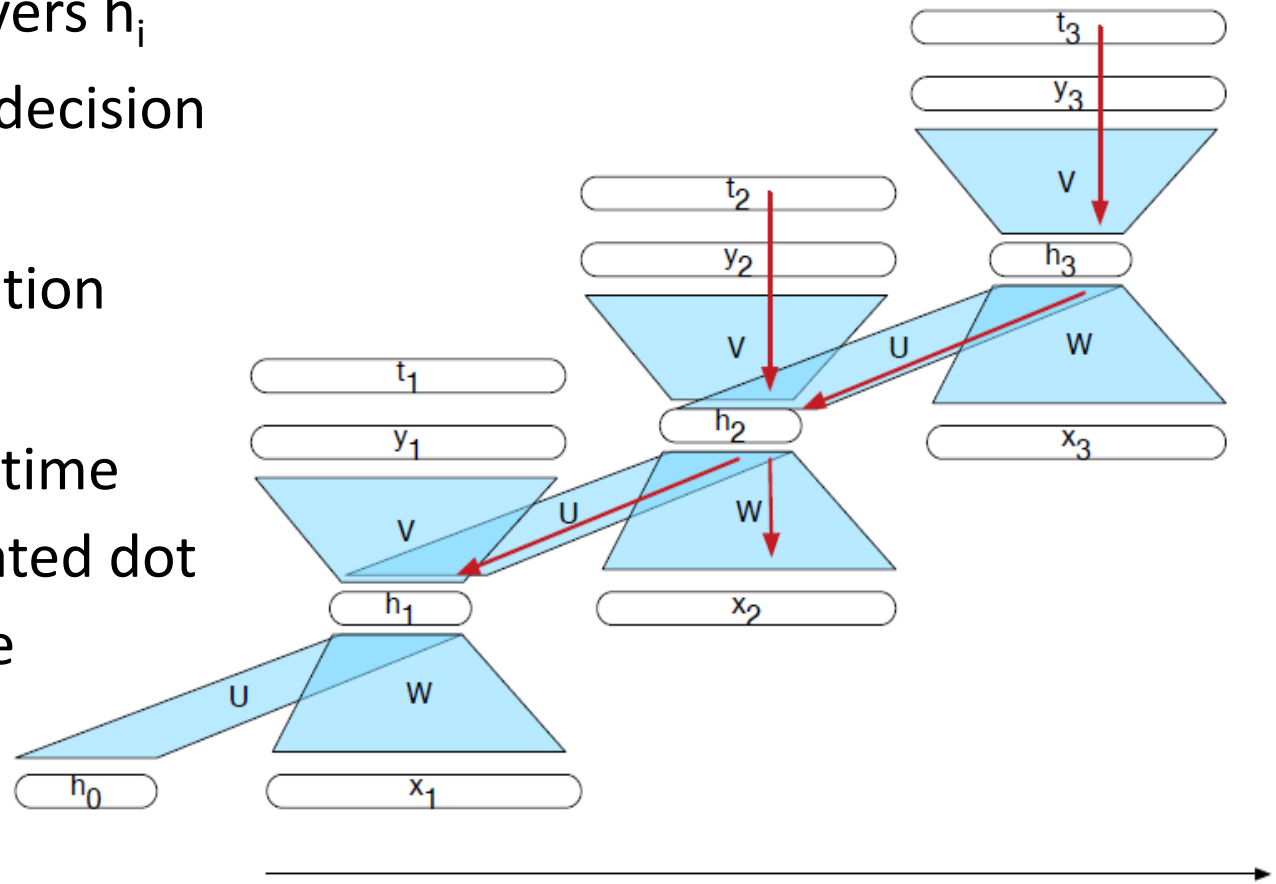
*The flights the airline **was** cancelling **were** full.*

- Assigning a high probability to ***was*** following ***airline*** is straightforward
  - since it provides a strong local context for the singular agreement.
- However, assigning an appropriate probability to ***were*** is quite difficult
  - not only because the plural flights is quite distant
  - but also because the more recent context contains singular constituents.

Inability of SRNs to carry forward critical long distance information. Hidden layers h_i

1. Provide information useful to the decision being made in the current context
2. Update and carry forward information useful for future decisions.

Backpropagation of training error back in time through the hidden layers results in repeated dot products, determined by the length of the sequence

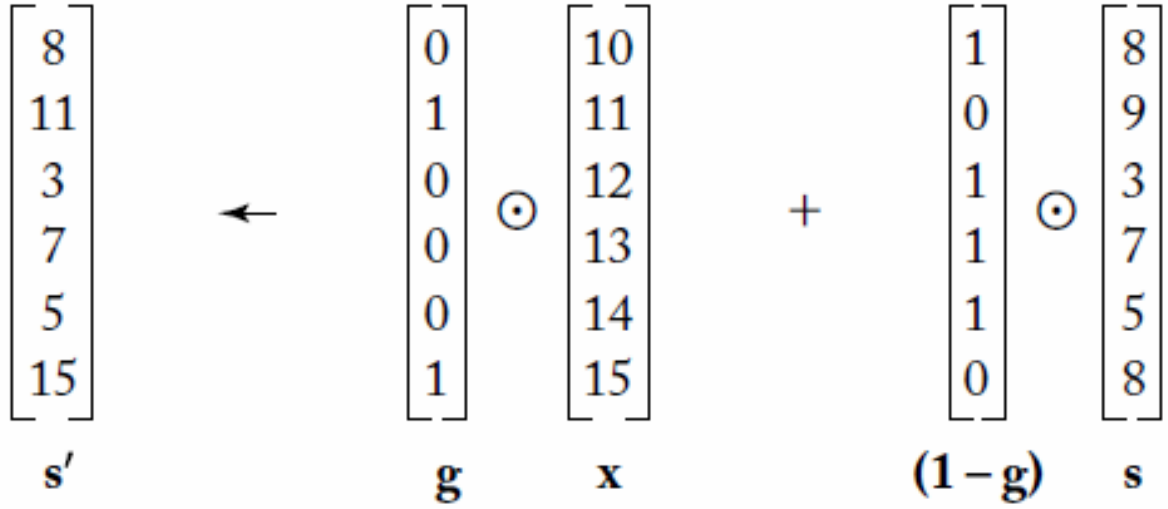


→

gradients are often either driven to zero or saturate (vanishing or exploding gradients)

LSTMs and GRUs : Gates

- Think of $\boldsymbol{h}$ as a state memory: you read the content h_{t-1} to write new content at h_t.
- At each computation tep, the entire state is read, and the entire is possibly rewritten
- Solution: provide more controlled "memory access" with gates



Using binary gate vector g to control access to memory s'.

- Gate **g** specifies what element of **x** should be copied in memory **s**.
- **(1 – g)** specifies what should be kept.

⊙ : Hadamard product
Elementwise multiplication

LSTMs and GRUs: learnable gates

These gates serve as building blocks : But they have to be differentiable.

Replace $g \in \{0,1\}^n$ with $g' \in R^n$ which is then passed through a sigmoid function $\sigma(g') \in \{0,1\}^n$

Split the hidden layer into two vectors c and h and have three learnable gates

$$g_t = tanh(U_g h_{t-1} + W_g x_t)$$

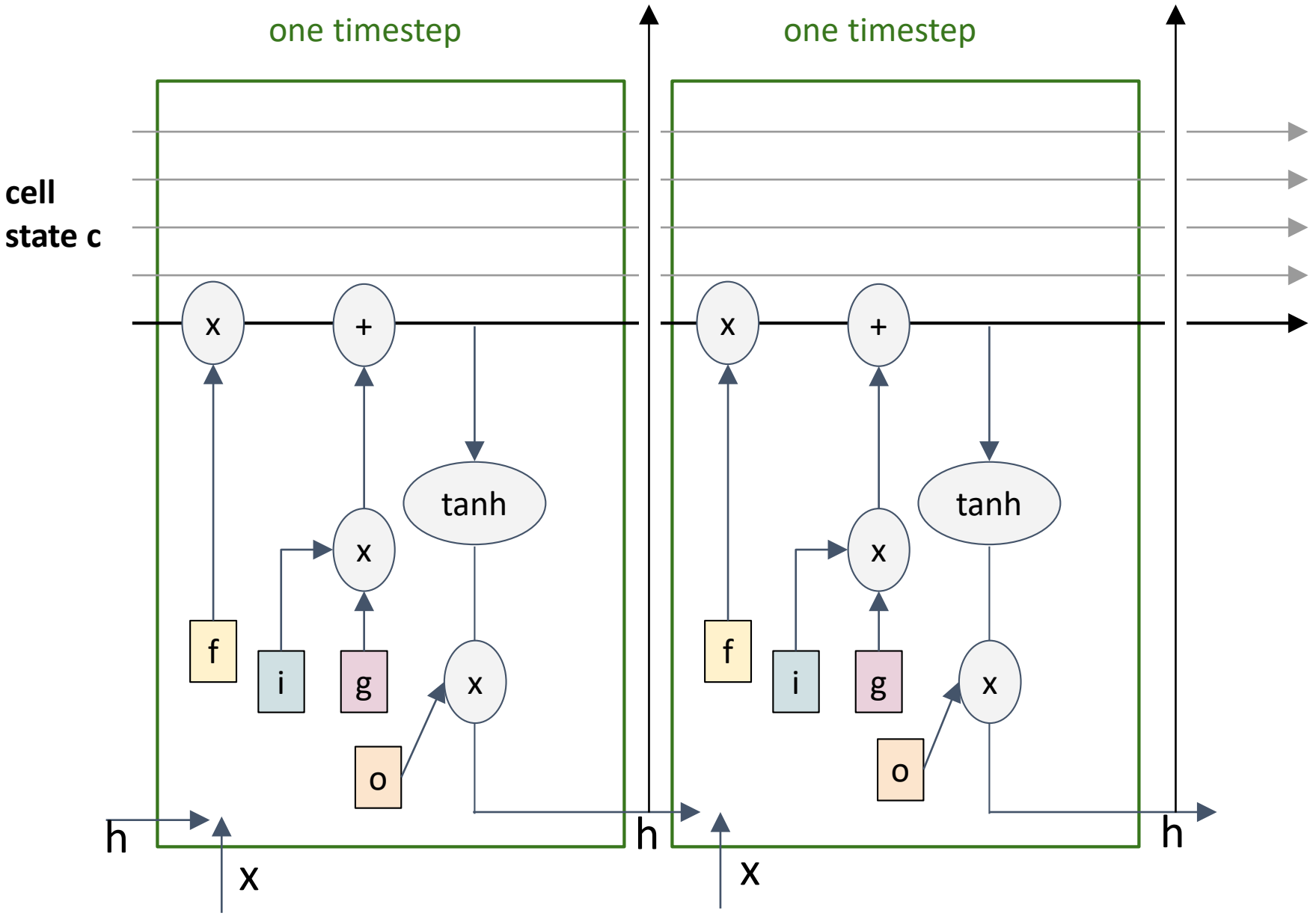
Input $$i_t = \sigma(U_i h_{t-1} + W_i x_t)$$

Forget $$f_t = \sigma(U_f h_{t-1} + W_f x_t)$$

Output $$o_t = \sigma(U_o h_{t-1} + W_o x_t)$$

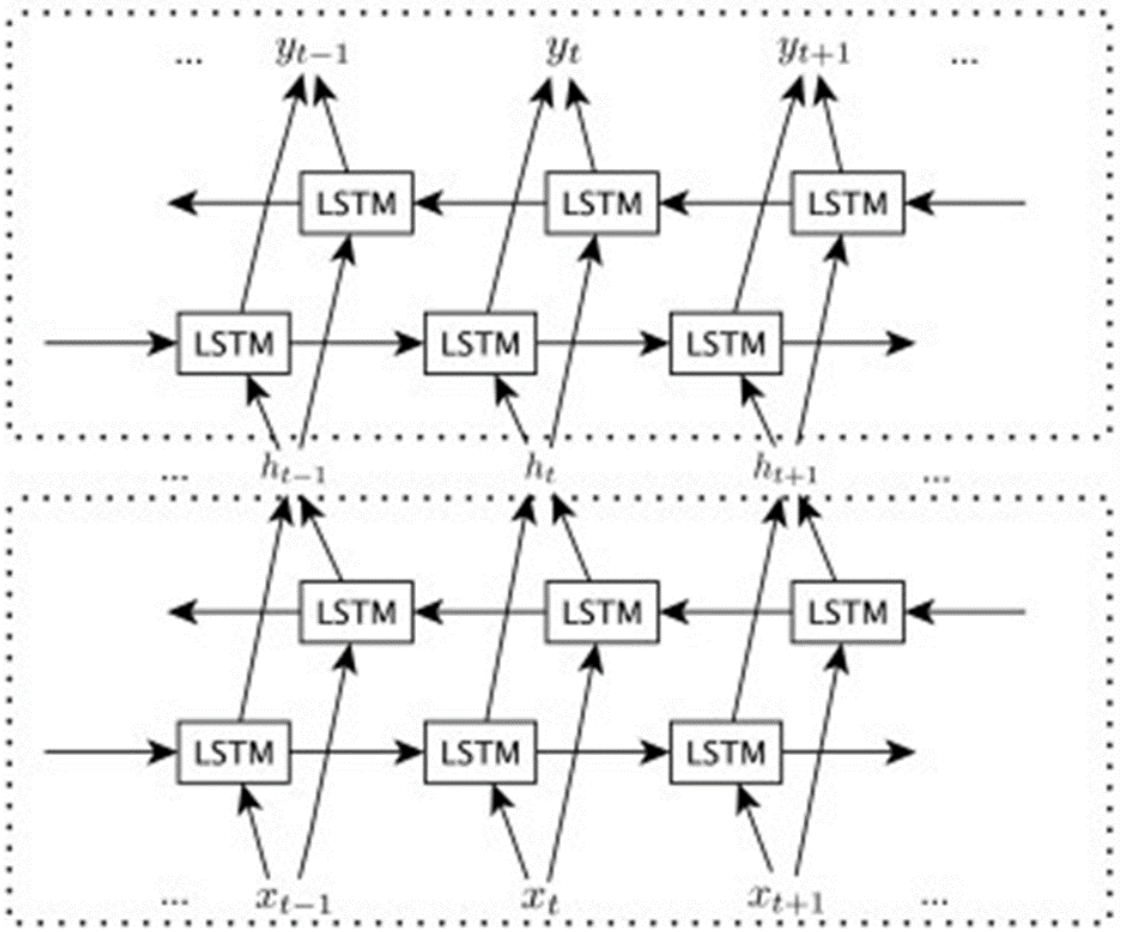
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot tanh(c_t)$$



$$
\begin{aligned}
g_t &= tanh(U_g h_{t-1} + W_g x_t) \\
i_t &= \sigma(U_i h_{t-1} + W_i x_t) \\
f_t &= \sigma(U_f h_{t-1} + W_f x_t) \\
o_t &= \sigma(U_o h_{t-1} + W_o x_t) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

Stacked bi-directional LSTM network



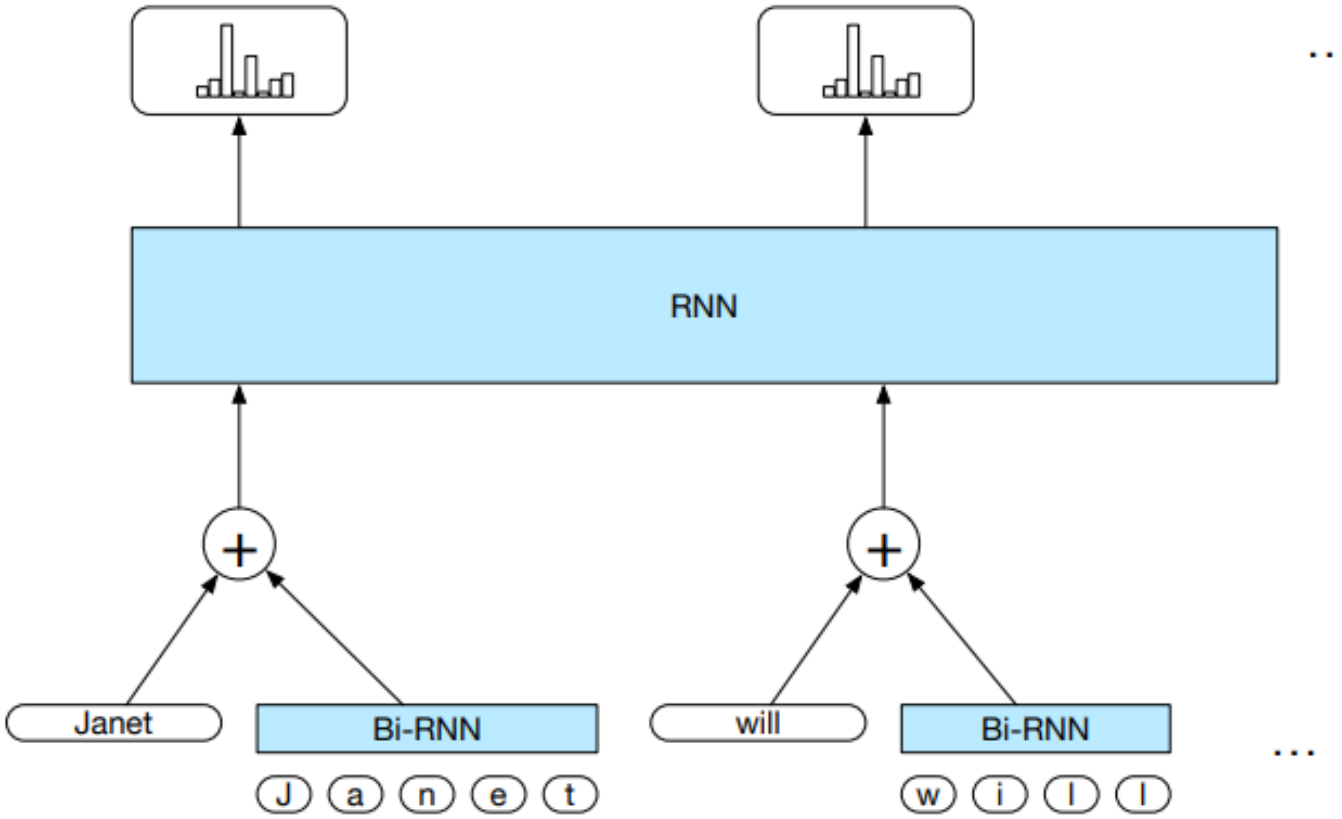
Word or Character Embeddings as inputs

Word-based embeddings are great at finding distributional similarity between words. However, there are significant issues with any solely word-based approach:

- For some languages and applications, the lexicon is simply too large
- Unknown words
- Morphological information is an important source of information for many applications. Word-based methods are blind to such regularities.

Overcoming issues related to word embeddings

Augment input word representations with embeddings derived from the characters that make up the words.



Character-level word embeddings

The character sequence for each word in the input is run through a bidirectional RNN consisting of two independent RNNs