Intel Nervana™ AI Academy

RECURRENT NEURAL NETWORKS (RNNS)

ISSUE: VARIABLE LENGTH SEQUENCES OF WORDS

- With images, we forced them into a specific input dimension
- Not obvious how to do this with text
- For example, classify tweets as positive, negative, or neutral
- Tweets can have a variable number of words
- What to do?

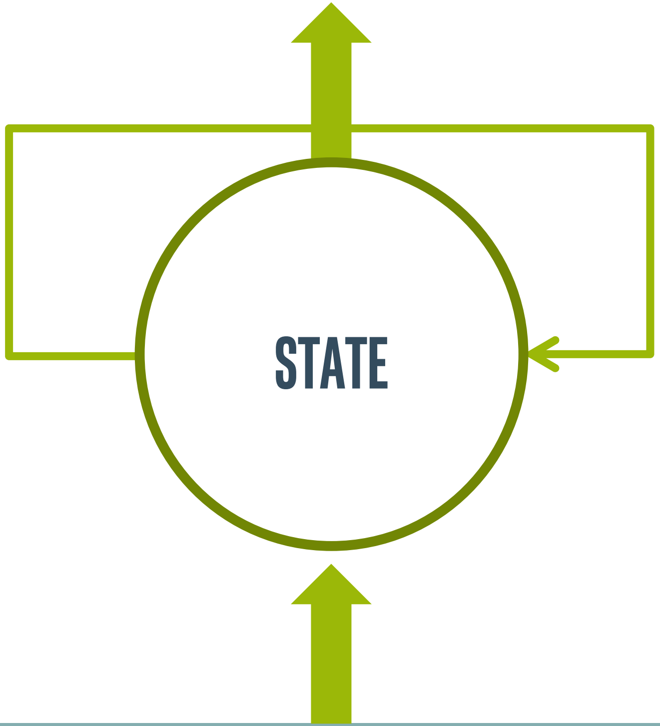
ISSUE: ORDERING OF WORDS IS IMPORTANT

- Want to do better than “bag of words” implementations
- Ideally, each word is processed or understood in the appropriate context
- Need to have some notion of “context”
- Words should be handled differently depending on “context”
- Also, each word should update the context

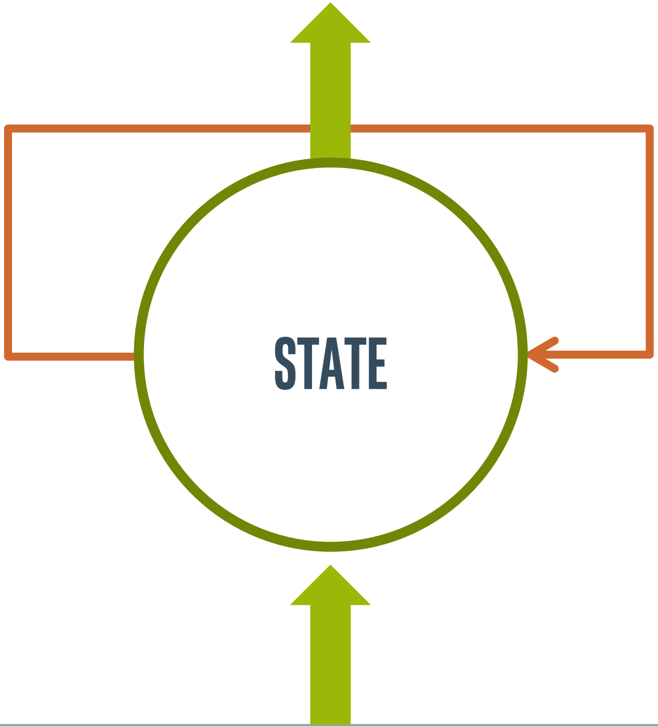
IDEA: USE THE NOTION OF "RECURRENCE"

- Input words one by one
- Network outputs two things:
  - Prediction: What would be the prediction if the sequence ended with that word
  - State: Summary of everything that happened in the past
- This way, can handle variable lengths of text
- The response to a word depends on the words that preceded it

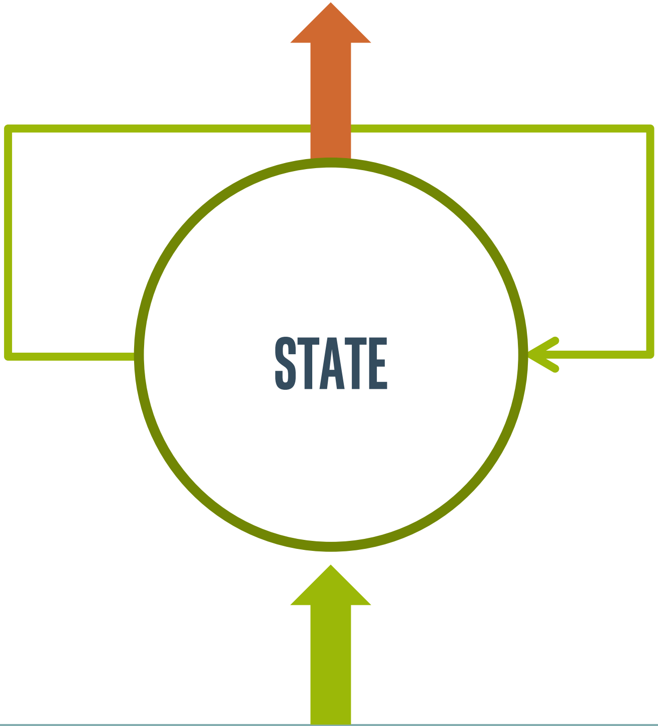
IDEA: USE THE NOTION OF "RECURRENCE"



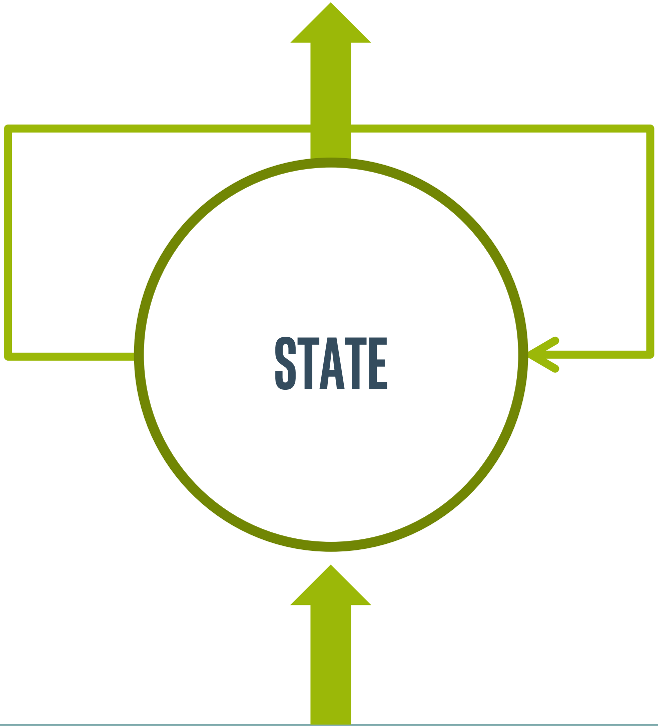
IDEA: USE THE NOTION OF "RECURRENCE"



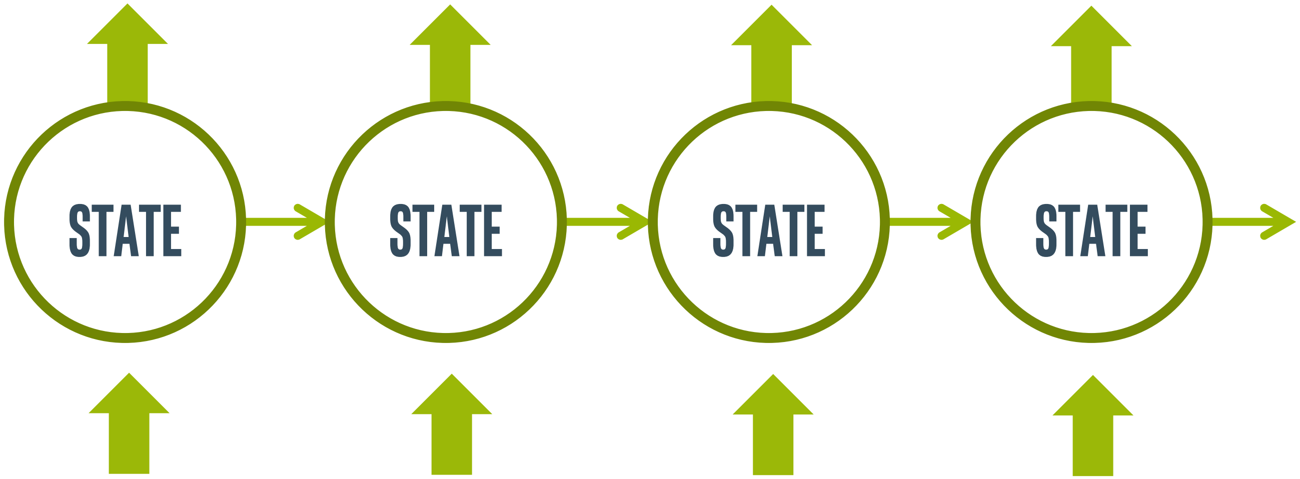
IDEA: USE THE NOTION OF “RECURRENCE”



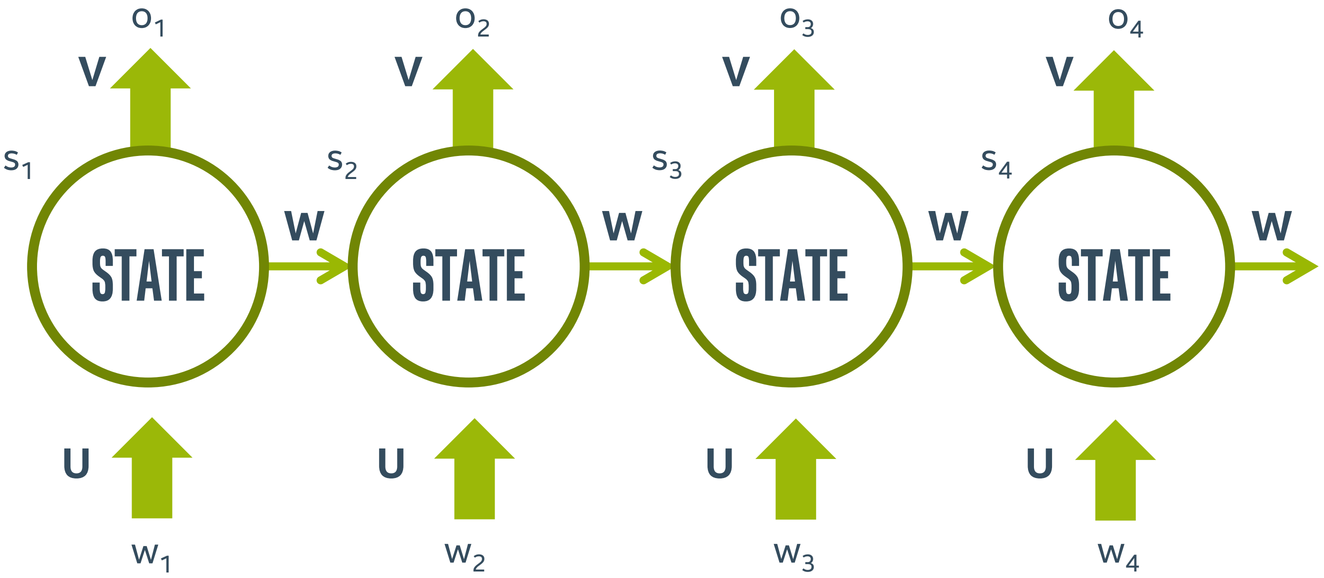
IDEA: USE THE NOTION OF "RECURRENCE"



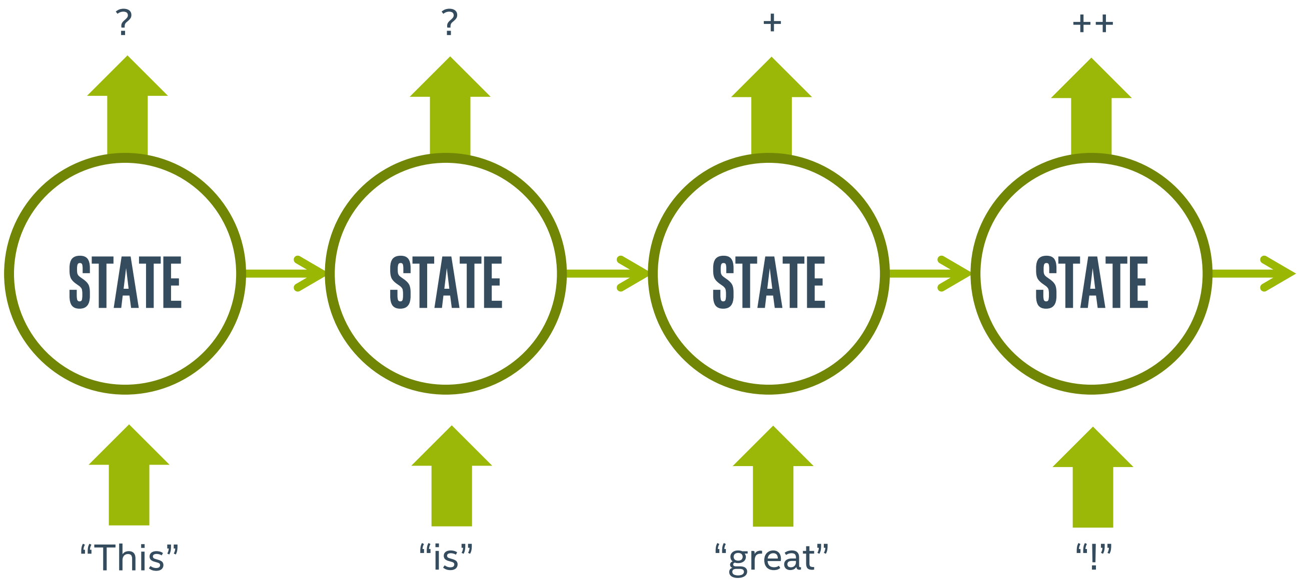
“UNROLLING” THE RNN



"UNROLLING" THE RNN

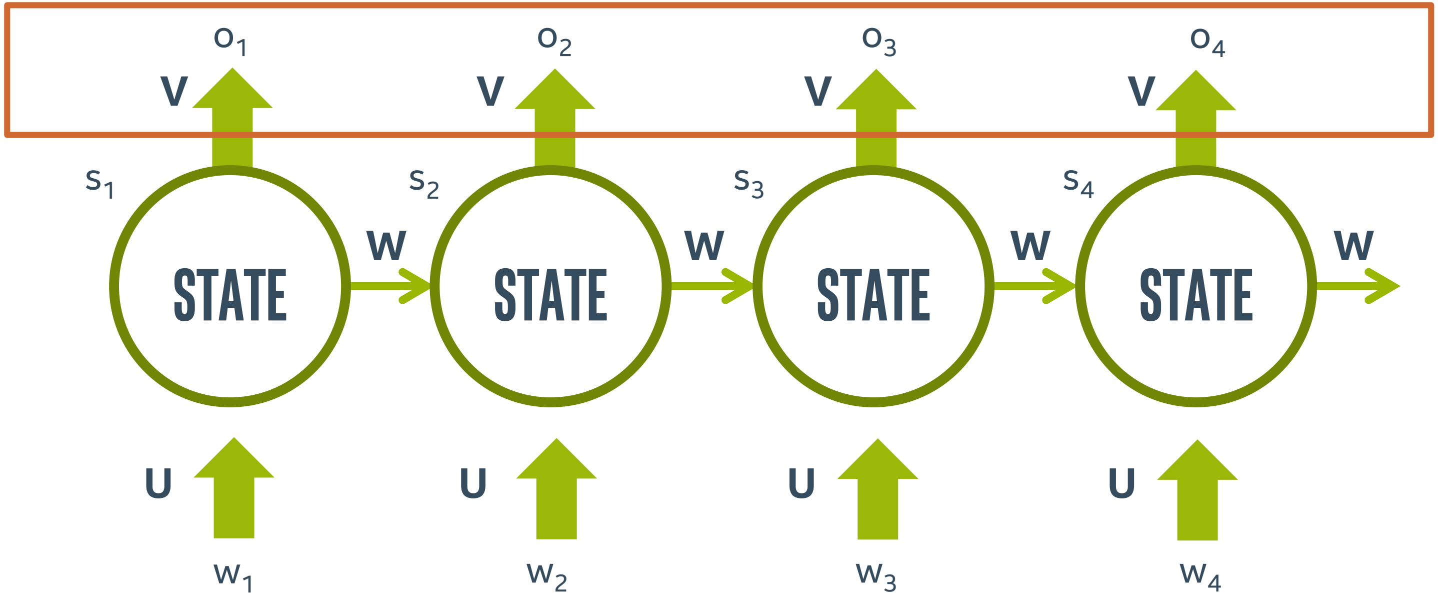


“UNROLLING” THE RNN



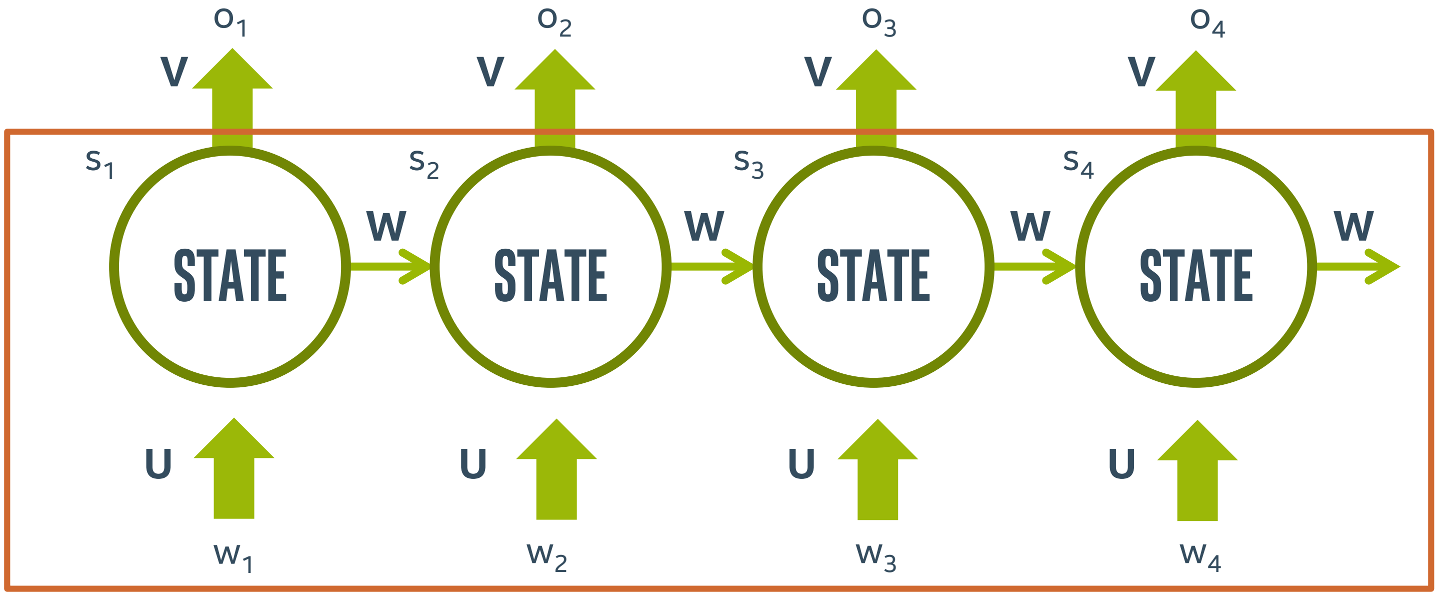
"UNROLLING" THE RNN

In Keras, this part is accomplished by a subsequent Dense layer.



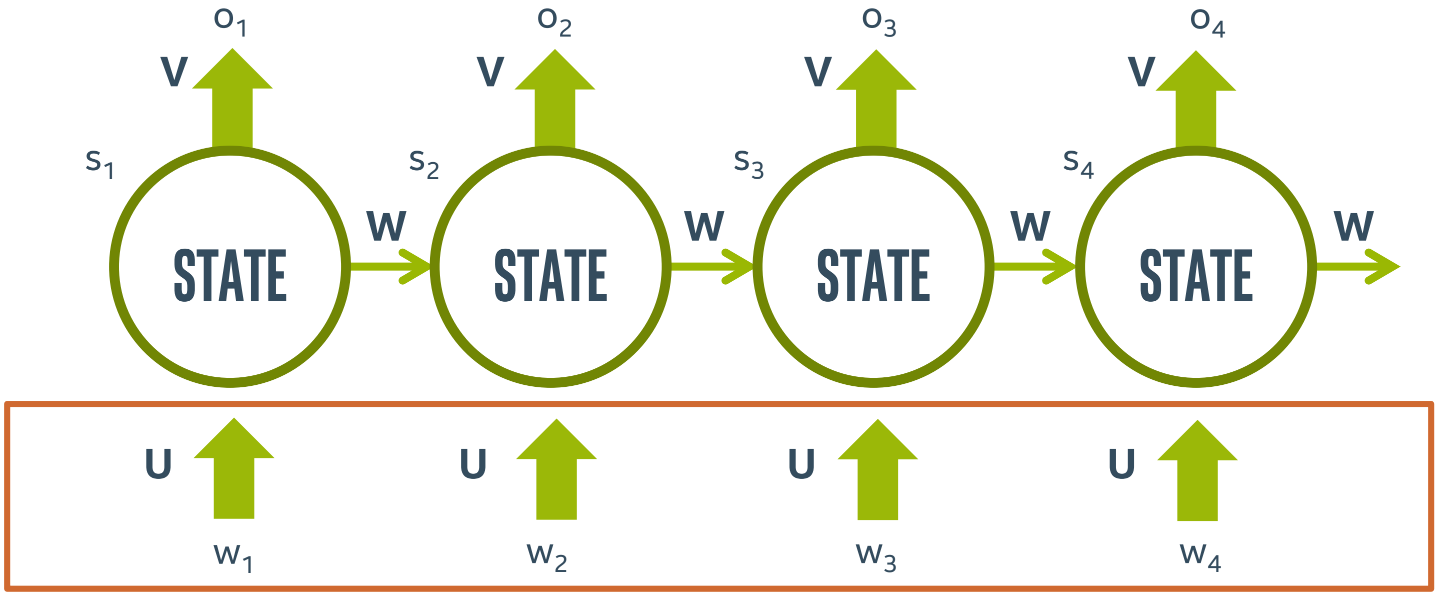
"UNROLLING" THE RNN

This part is the core RNN.



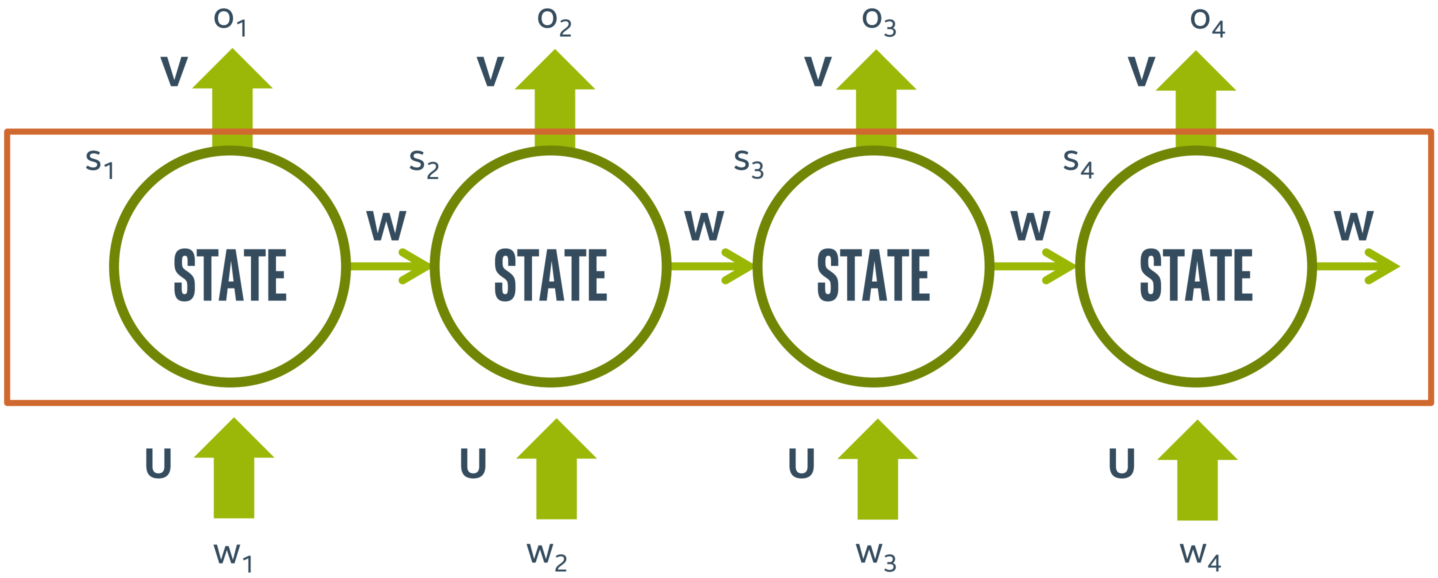
"UNROLLING" THE RNN

Keras calls this part the "kernel" (e.g. `kernel_initializer`,...).



"UNROLLING" THE RNN

Keras calls this part "recurrent" (`recurrent_initializer,...`).



MATHEMATICAL DETAILS

- w_i is the word at position i
- s_i is the state at position i
- o_i is the output at position i
- $s_i = f(Uw_i + Ws_{i-1})$ (Core RNN)
- $o_i = softmax(Vs_i)$ (subsequent dense layer)

MATHEMATICAL DETAILS

- w_i is the word at position i
- s_i is the state at position i
- o_i is the output at position i
- $s_i = f(Uw_i + Ws_{i-1})$ (Core RNN)
- $o_i = softmax(Vs_i)$ (subsequent dense layer)

In other words:

- current state = function1(old state, current input)
- current output = function2(current state)
- We learn function1 and function2 by training our network!

MORE MATHEMATICAL DETAILS

- r = dimension of input vector
- s = dimension of hidden state
- t = dimension of output vector (after dense layer)
- U is a s x r matrix
- W is a s x s matrix
- V is a t x s matrix

Note: The weight matrices U,V,W are the same across all positions.

PRACTICAL DETAILS

- Often, we train on just the ”final” output and ignore the intermediate outputs
- Slight variation called Backpropagation Through Time (BPTT) is used to train RNNs
- Sensitive to length of sequence (due to “vanishing/exploding gradient” problem)
- In practice, we still set a maximum length to our sequences
  - If input is shorter than maximum, we “pad” it
  - If input is longer than maximum, we truncate

OTHER USES OF RNNS

- We have focused on text/words as application
- But, RNNs can be used for other sequential data
  - Time-Series Data
  - Speech Recognition
  - Sensor Data
  - Genome Sequences

WEAKNESSES OF RNNS

- Nature of state transition means it is hard to keep information from distant past in current memory without reinforcement
- In the next lecture, we will introduce LSTMs, which have a more complex mechanism for updated the state

Nervana™ AI Academy