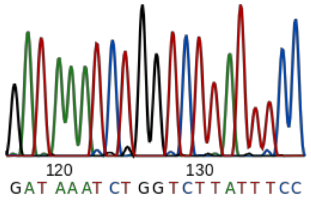
RNNs / LSTMs

Why Recurrent Neural Networks (RNNs)?

- Humans don't start their thinking from scratch every second.
- As you read this essay, you understand each word based on your understanding of previous words.
- You don't throw everything away and start thinking from scratch again.
- Not all problems can be converted into one with fixed length inputs and outputs



DNA Sequencing

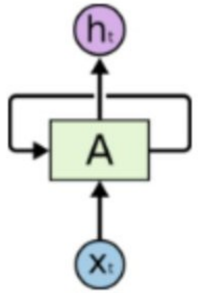
Speech recognition

Video analysis

Adapted from : http://colah.github.io/

RNNs

- RNNs address this issue!
- Recurrent Neural Networks take the previous output or hidden states as inputs. The composite input at time t has some historical information about the happenings at time T < t

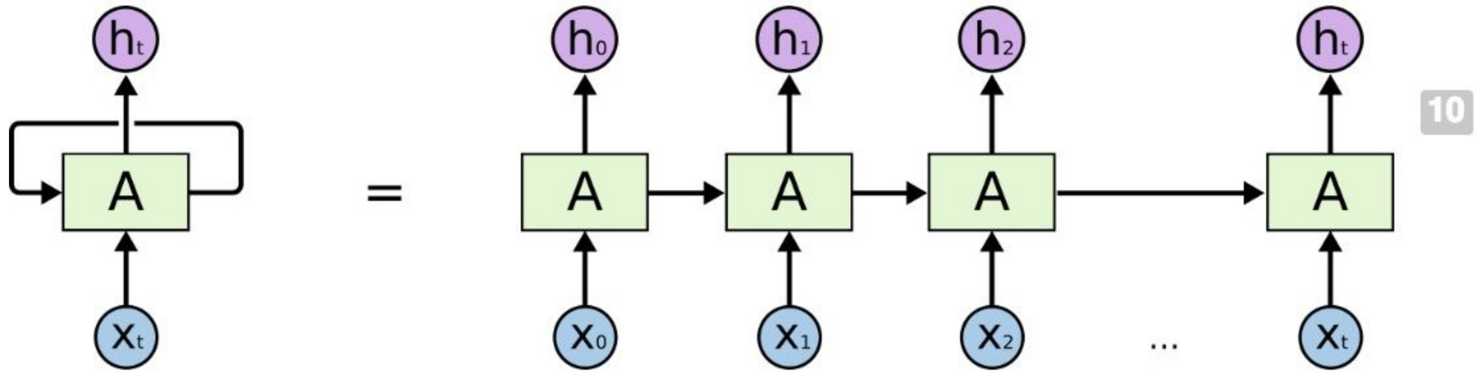


Recurrent Neural Networks have loops.

In the diagram, a chunk of neural network, A, looks at some input xt and outputs a value ht. A loop allows information to be passed from one step of the network to the next.

Adapted from : http://colah.github.io/

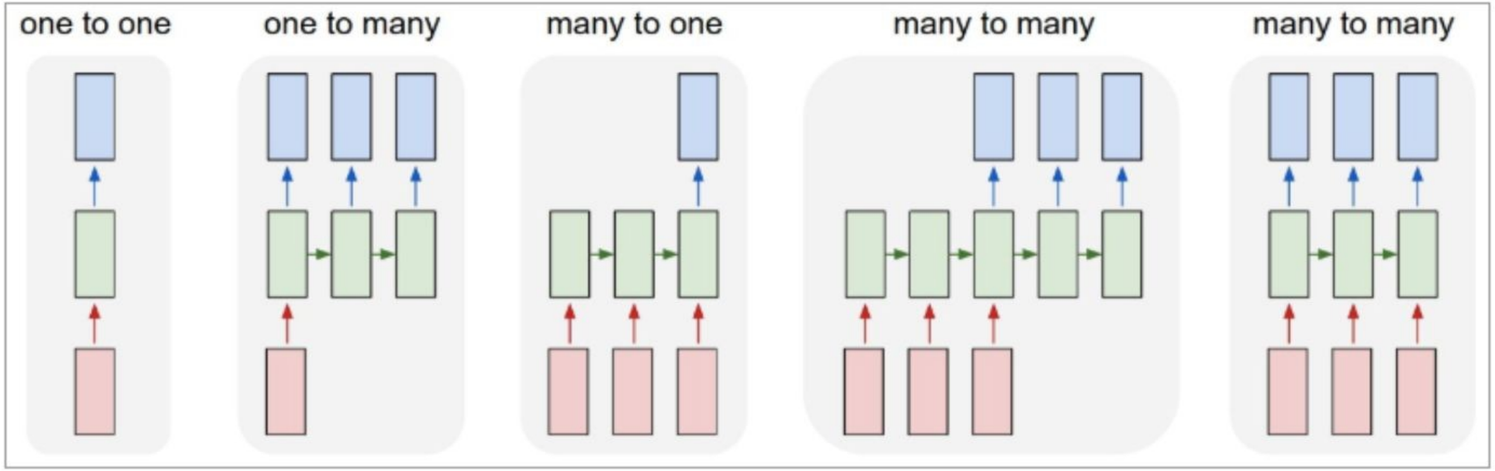
RNNs



An unrolled recurrent neural network.

Adapted from : http://colah.github.io/

RNNs offer a lot of flexibility

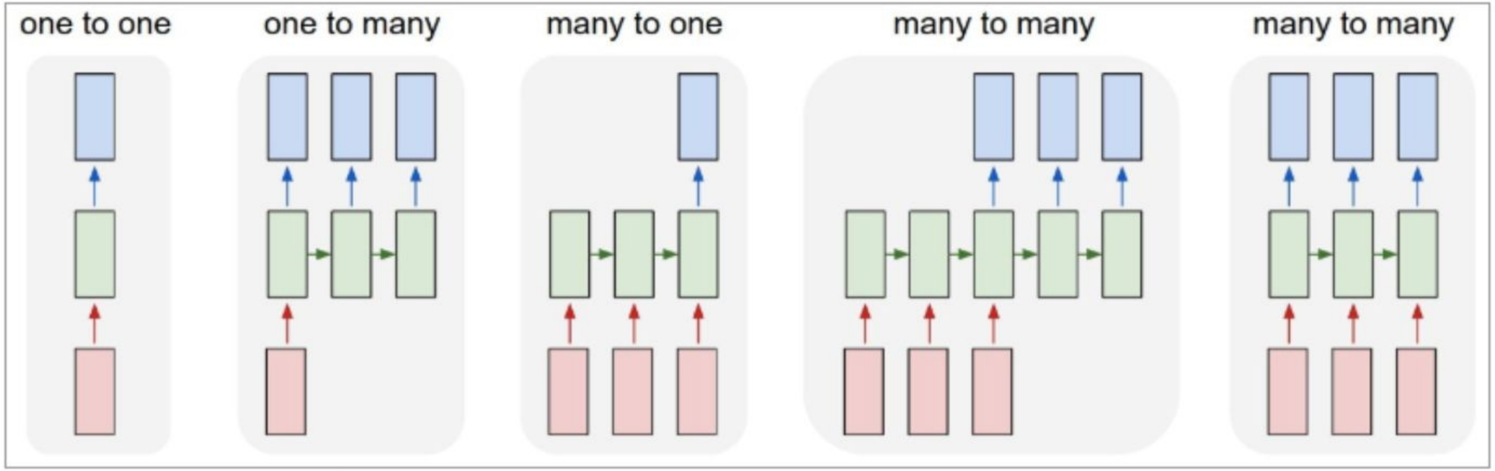


Vanilla Neural Networks

Each rectangle is a vector and arrows represent functions (eg. Matrix multiplication)

Adapted from : http://colah.github.io/

RNNs offer a lot of flexibility

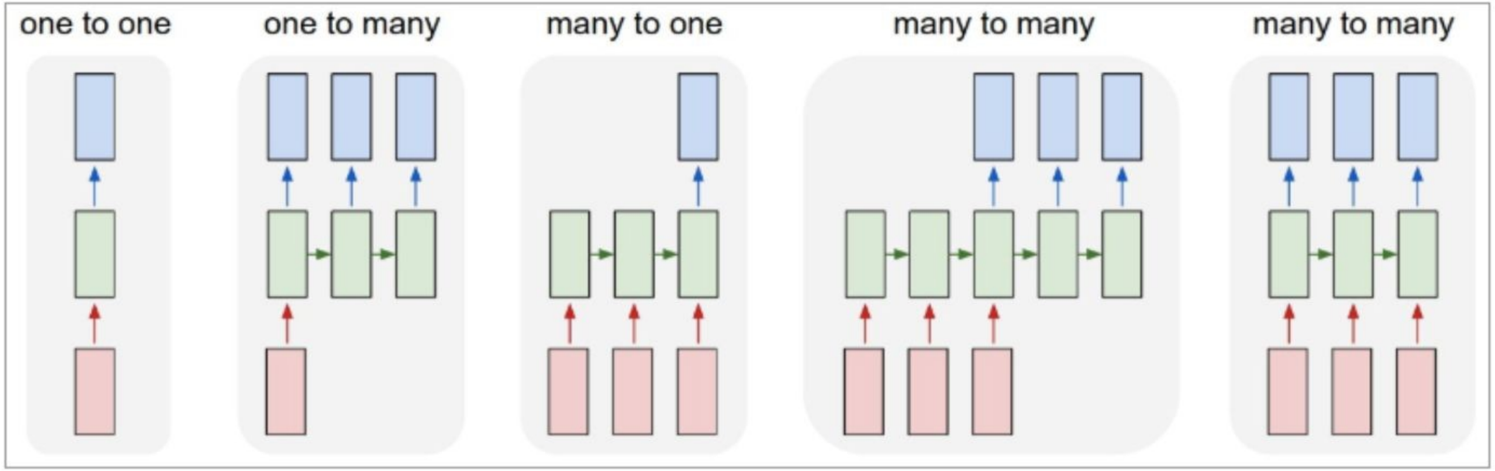


e.g. **Image Captioning**
image -> sequence of words

Each rectangle is a vector and arrows represent functions (eg. Matrix multiplication)

Adapted from : http://colah.github.io/

RNNs offer a lot of flexibility

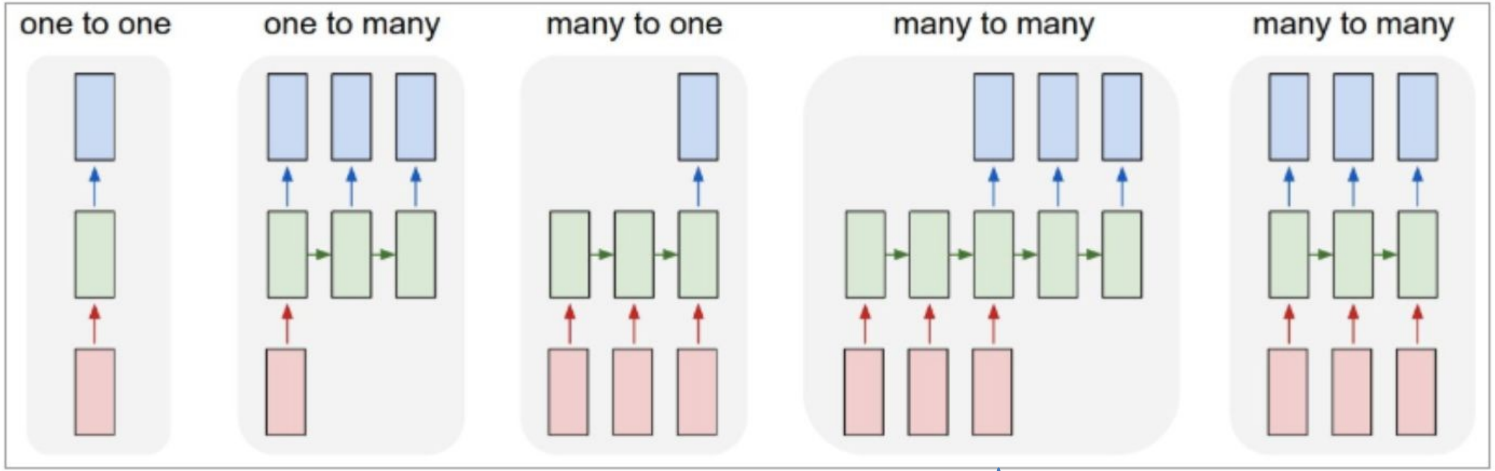


e.g. **Sentiment Classification**
sequence of words -> sentiment

Each rectangle is a vector and arrows represent functions (eg. Matrix multiplication)

Adapted from : http://colah.github.io/

RNNs offer a lot of flexibility

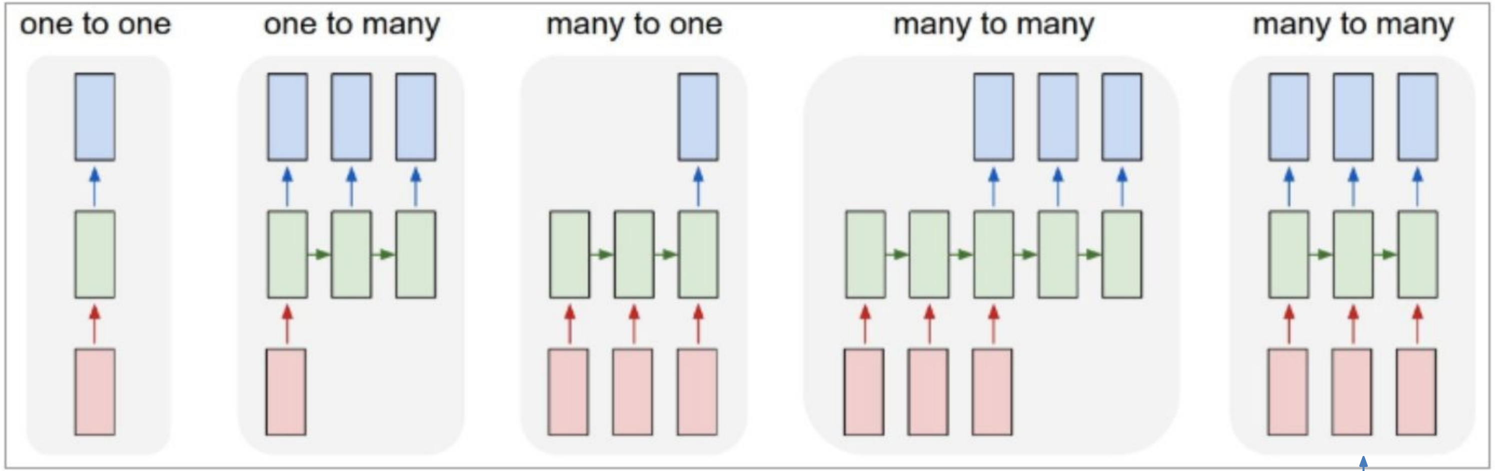


e.g. **Machine Translation**
seq of words -> seq of words

Each rectangle is a vector and arrows represent functions (eg. Matrix multiplication)

Adapted from : http://colah.github.io/

RNNs offer a lot of flexibility

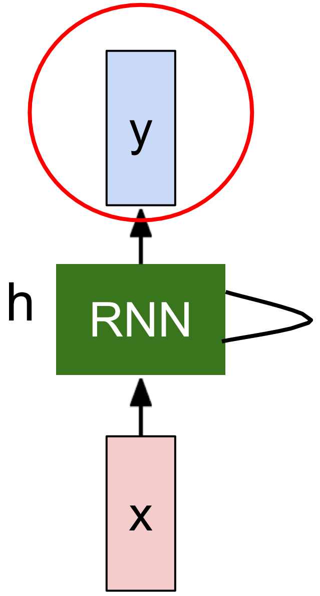


e.g. **Video classification on frame level**

Each rectangle is a vector and arrows represent functions (eg. Matrix multiplication)

Adapted from : http://colah.github.io/

RNNs



usually want to predict a vector at some time steps

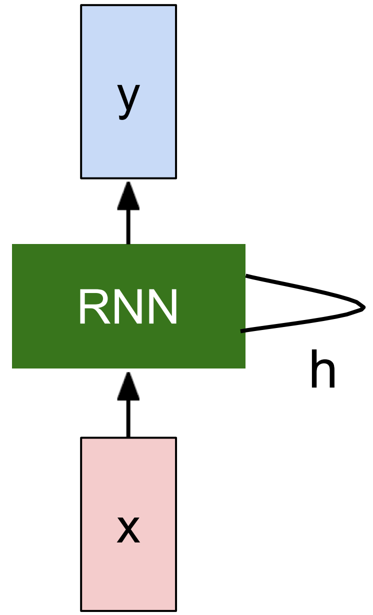
Adapted from cs231n by Fei-Fei Li

RNNs

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

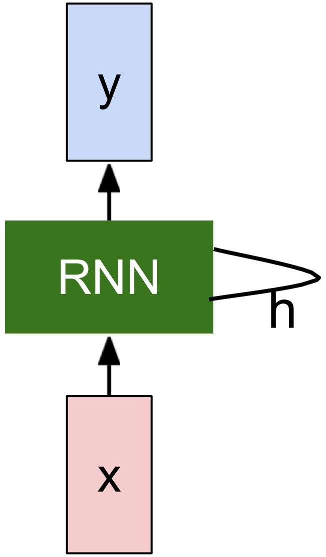
Notice: the same function and the same set of parameters are used at every time step.



Adapted from cs231n by Fei-Fei Li

(Vanilla) RNN

The state consists of a single *"hidden"* vector **h**



$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Adapted from cs231n by Fei-Fei Li

Backpropagation is used!

- Treat the unfolded network as one big feed-forward network!
- Compute gradients through the usual backpropagation
- Update shared weights

Example: Sentiment Classification

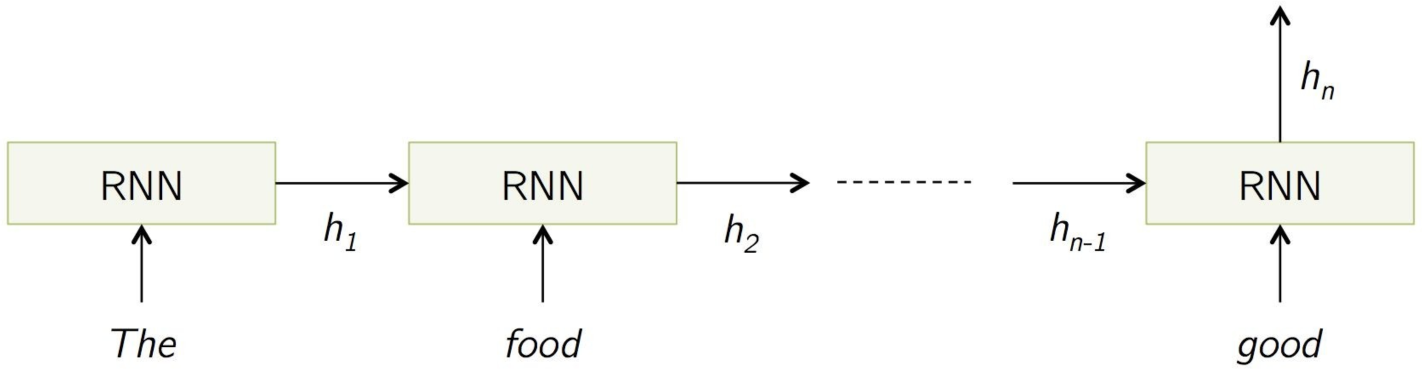
Classify a review review from Yelp or movie review from IMDB

…..As positive or negative

Inputs – one or more sentences consisting of multiple words
Output – Positive / Negative classification

Eg : ”The food is really good”
“It was a fantastic movie”

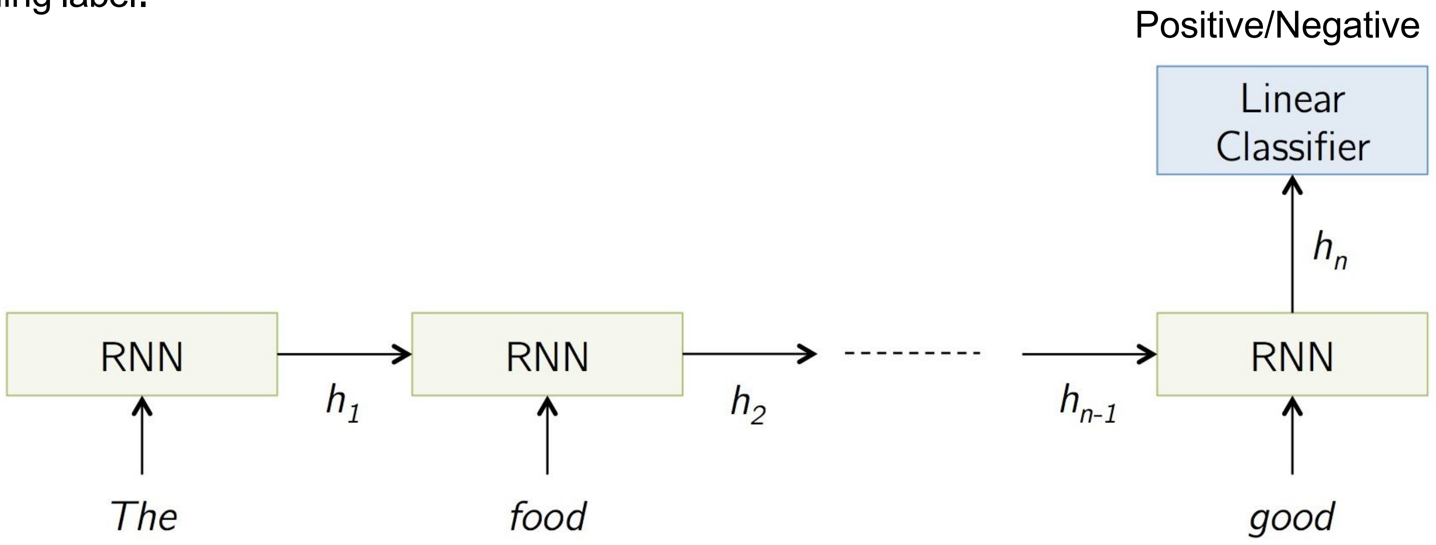
Sentiment Classification



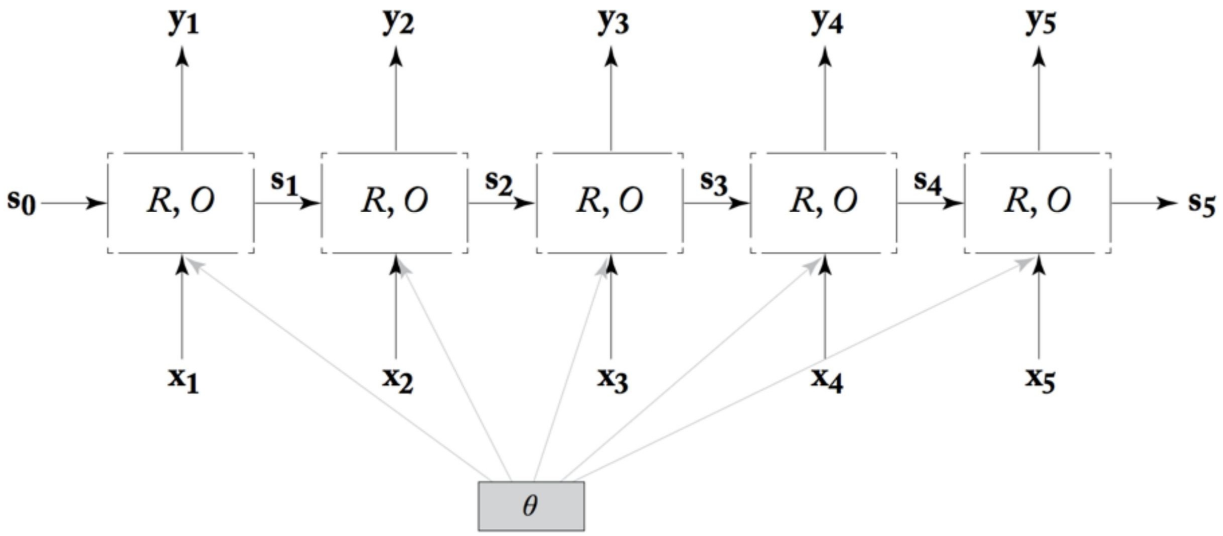
•You first change your words into vector embedding and then feed into the RNN
•The RNN that receives a sequence of vectors as input and considers the order of the vectors to generate prediction.

Sentiment Classification

We are using sigmoid because we are trying to predict if this text has positive or negative sentiment.
We don't care about the sigmoid outputs except for the very last one, we can ignore the rest. We'll calculate the cost from the output of the last step and the training label.



RNN Language Modeling



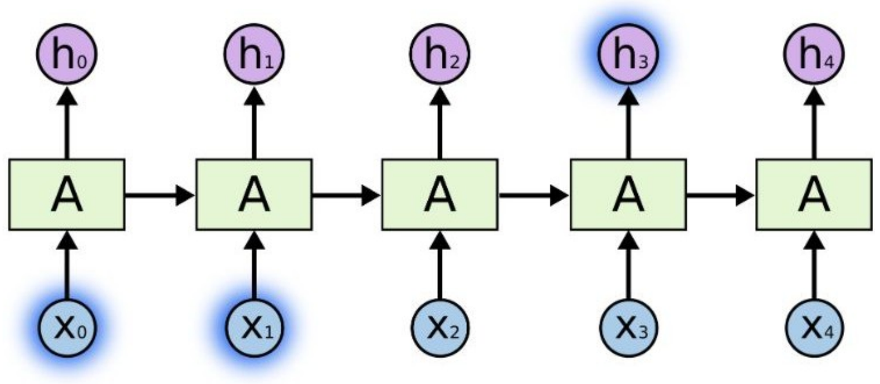
$$s_i = R(x_i, s_{i-1})$$

$$y_i = O(s_i)$$

Goldberg 2017

Problem of Long Term Dependencies

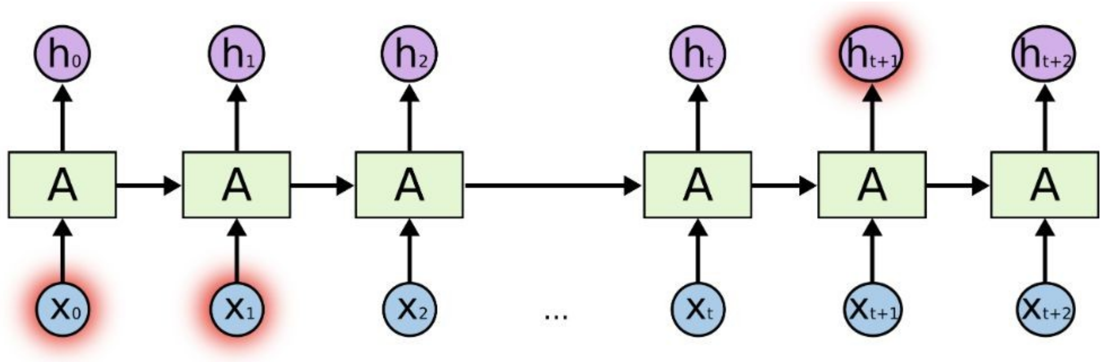
- If we are trying to predict the last word in “the clouds are in the *sky*,” we don’t need any further context – it’s pretty obvious the next word is going to be sky.



Source: http://colah.github.io/

Problem of Long Term Dependencies

- But there are also cases where we need more context. Consider trying to predict the last word in the text “I grew up in France… I speak fluent *French*.”



Source: http://colah.github.io/

Problem of Long Term Dependencies

- In theory, RNNs are absolutely capable of handling such “long-term dependencies.”

- Sadly, in practice, RNNs don’t seem to be able to learn them.

Source: http://colah.github.io/

Vanishing / Exploding Gradients

- In the same way a product of k real numbers can shrink to zero or explode to infinity, so can a product of matrices

For large values of W, the gradients will grow exponentially with time (exploding).

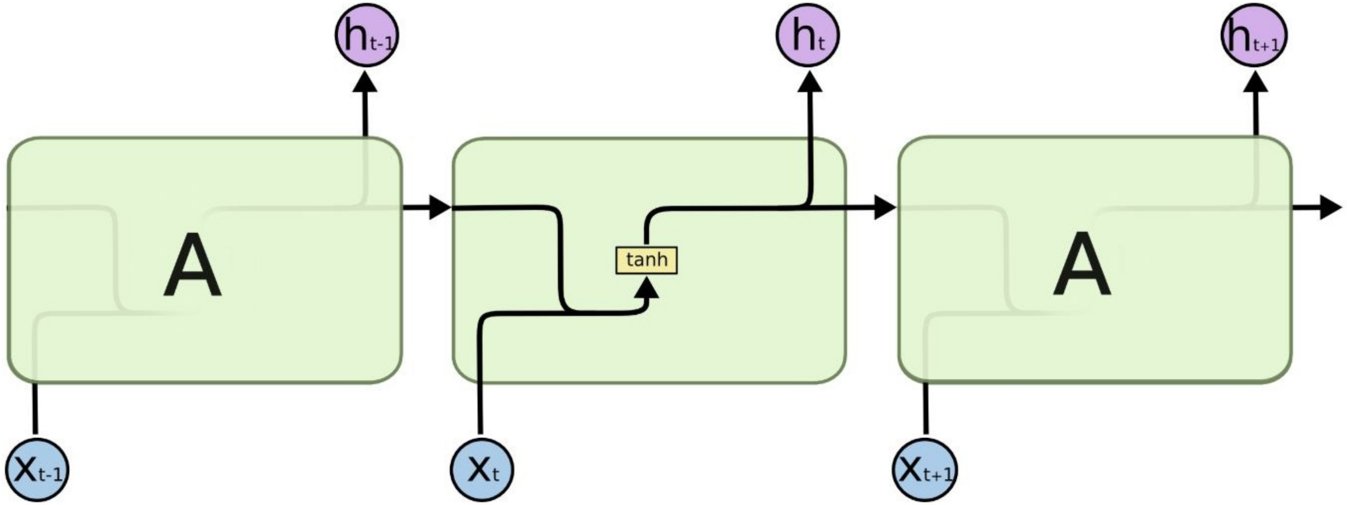
For very small values of W, the gradients will shrink exponentially with time (vanishing).

LSTMs do not have this problem!

LSTM (Long Short Term Memory)

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies.

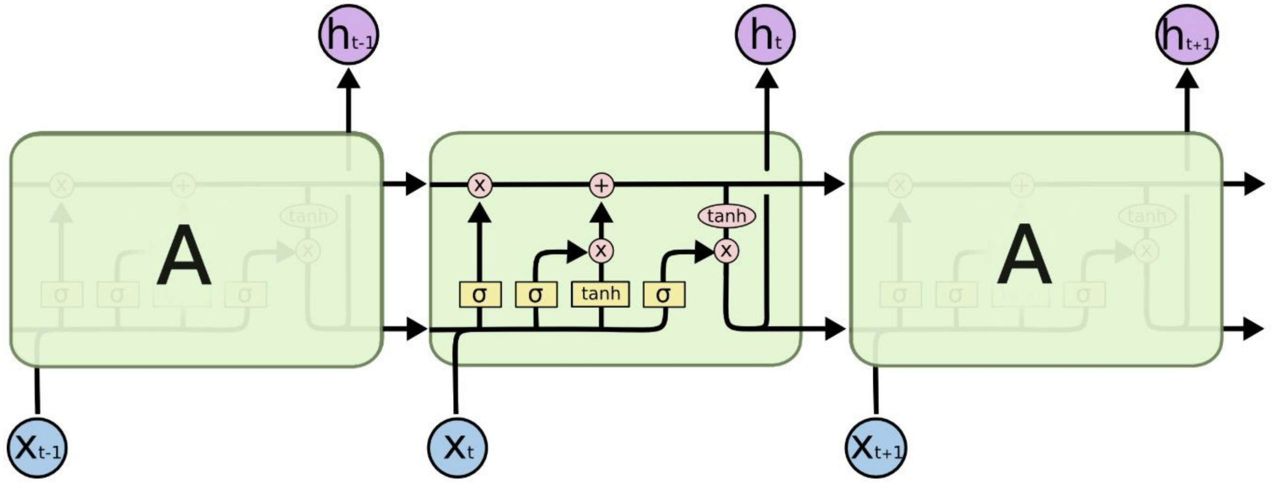
In standard RNNs, the repeating module will have a very simple structure, such as a single tanh layer.



The repeating module in a standard RNN contains a single layer.

Source: http://colah.github.io/

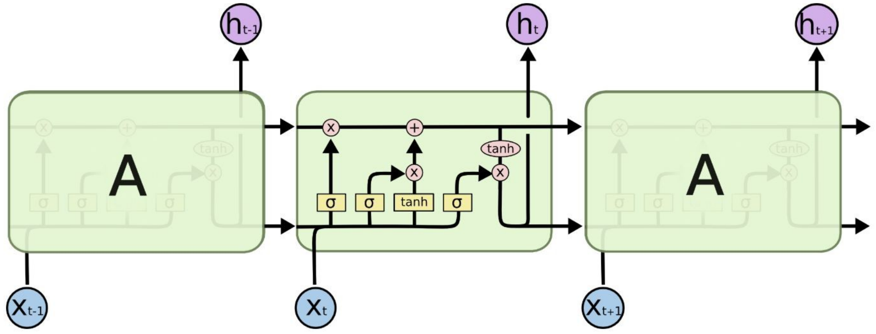
LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



The repeating module in an LSTM contains four interacting layers.

Source: http://colah.github.io/

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

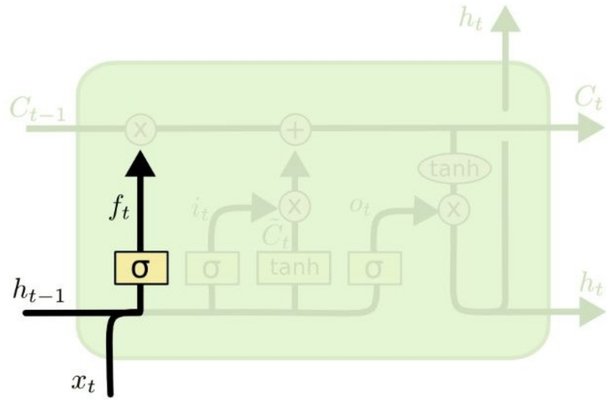


The repeating module in an LSTM contains four interacting layers.

Gates are a way to optionally let information through.

Source: http://colah.github.io/

Forget Gate

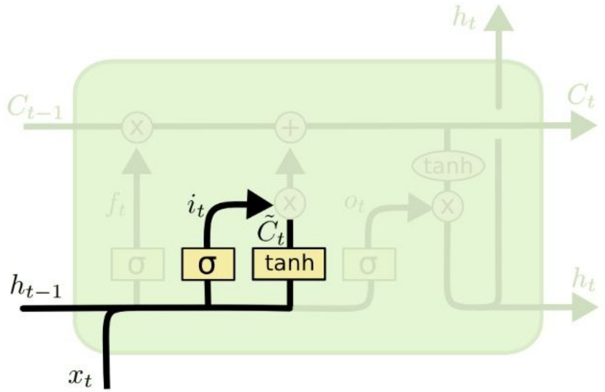


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Gates are a way to optionally let information through.

Source: http://colah.github.io/

Input Gate

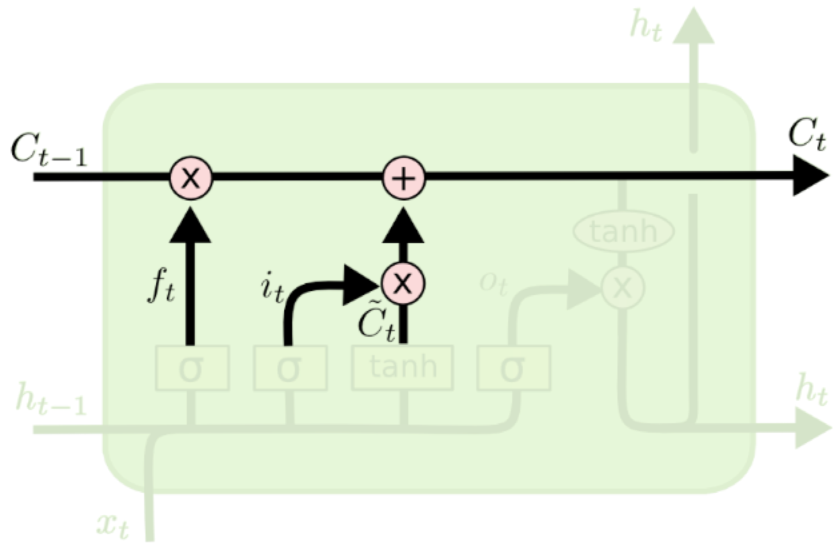


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Gates are a way to optionally let information through.

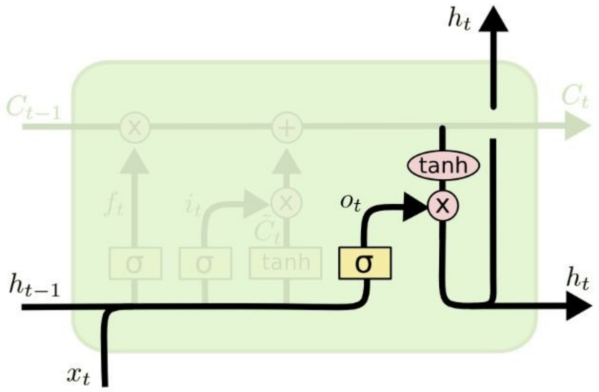
Source: http://colah.github.io/

New Cell state



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Gates are a way to optionally let information through.

Source: http://colah.github.io/

Summary

- RNNs are a widely used model for sequential data, including text
- RNNs are trainable with backpropogation
- RNNs learn complex and varied patterns in sequential data
- Vanilla RNNs are simple but don’t work very well
    - Backward flow of gradients in RNN can explode or vanish
- Common to use LSTM or GRU. Memory path makes them stably learn long-distance interactions.