



Deep Learning for Sequences

CMSC 389A: Lecture 8

Sujith Vishwajith

University of Maryland

Agenda

1. Applications
2. Word Embeddings
3. Recurrent Neural Network
4. Long Short-term Memory (LSTM)
5. Announcements

Applications

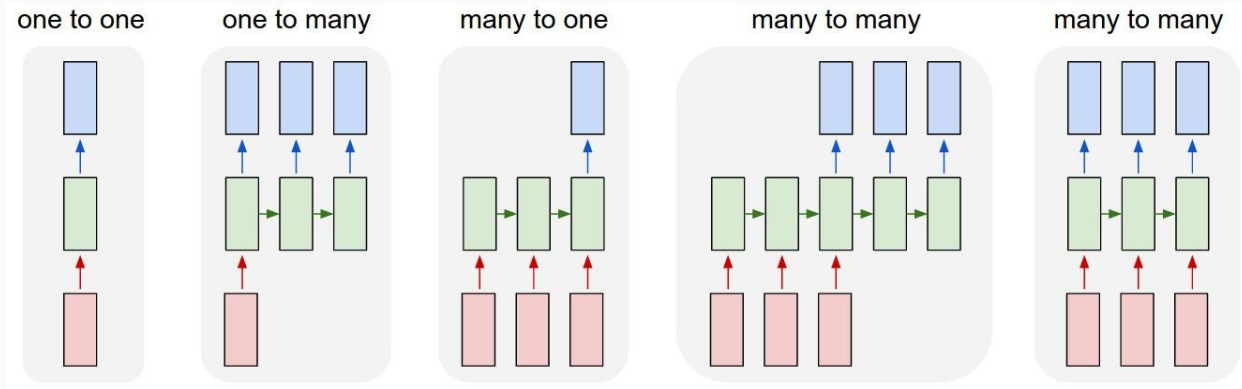
Applications

Deep learning for sequence based tasks are used everywhere!

Due to recent improvement in RNNs such as the LSTM model and computing power.

Takes advantage of the fact that neural networks are excellent at finding patterns.

Allows for automatic learning compared to statistical based natural language methods.



From Andrej Karpathy.

Applications (cont.)

1. Text Classification
2. Language Translation
3. Image Captioning
4. Question Answering
5. Sequence Prediction

User Reviews

★★★★★★★ **Amazing!**
16 December 2006 | by [jellenellie818](#) (United States) - [See all my reviews](#)

I have been a fan of Will Smith for years and I have to say this may be his best film yet! "The Pursuit of Happiness" is just a wonderful (based on a true) story, full of adventure, hope, and pain. I saw the movie last night in a packed theater. Big Willie Weekend has returned, and for good reason! It's a great movie to see during the holidays and definitely a tear-jerker! Perfect for a date, a night out with friends, or even with family. If you ever thought Will Smith really couldn't act (and shame on you!), you'll think otherwise once you see this movie. You can really feel what he's going through just by looking in his eyes. And Jaden Smith is too adorable! Their on screen chemistry is almost unbearable to watch! Go see this movie! Great acting, great directing, great writing...you won't regret it!

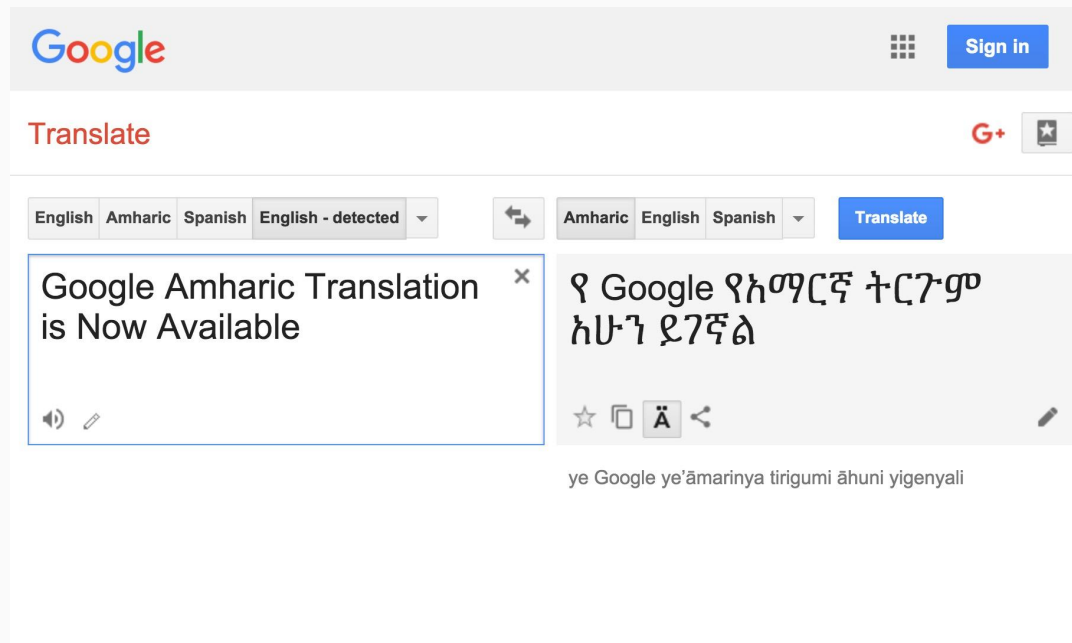
156 of 192 people found this review helpful. Was this review helpful to you?

[Review this title](#) | [See all 574 user reviews »](#)

Sentiment Analysis of Movie Reviews

Applications (cont.)

1. Text Classification
- 2. Language Translation**
3. Image Captioning
4. Question Answering
5. Sequence Prediction



Google Translate uses RNNs.

Applications (cont.)

1. Text Classification
2. Language Translation
- 3. Image Captioning**
4. Question Answering
5. Sequence Prediction

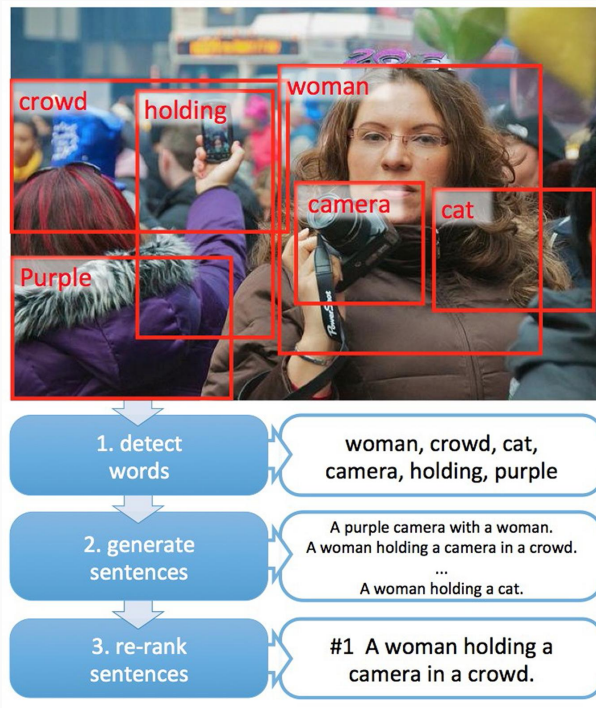



Image Captioning

Applications (cont.)

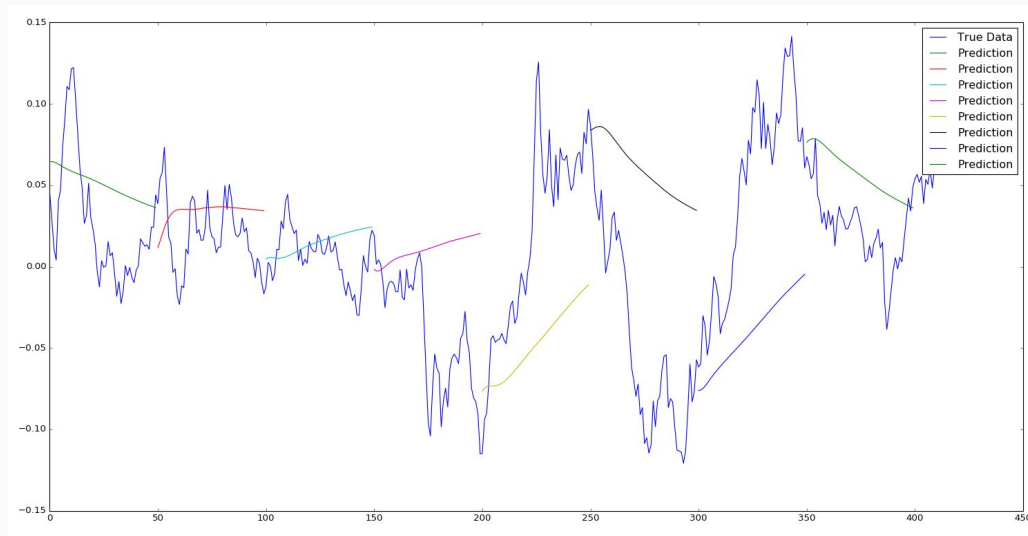
1. Text Classification
2. Language Translation
3. Image Captioning
- 4. Question Answering**
5. Sequence Prediction

 <p>What vegetable is on the plate? Neural Net: broccoli Ground Truth: broccoli</p>	 <p>What color are the shoes on the person's feet ? Neural Net: brown Ground Truth: brown</p>	 <p>How many school busses are there? Neural Net: 2 Ground Truth: 2</p>	 <p>What sport is this? Neural Net: baseball Ground Truth: baseball</p>
 <p>What is on top of the refrigerator? Neural Net: magnets Ground Truth: cereal</p>	 <p>What uniform is she wearing? Neural Net: shorts Ground Truth: girl scout</p>	 <p>What is the table number? Neural Net: 4 Ground Truth: 40</p>	 <p>What are people sitting under in the back? Neural Net: bench Ground Truth: tent</p>

Visual Question Answering

Applications (cont.)

1. Text Classification
2. Language Translation
3. Image Captioning
4. Question Answering
5. **Sequence Prediction**



Stock Price Prediction

Word Embeddings

Word Embeddings

Problem: We can't input words into a neural network since it isn't a number.

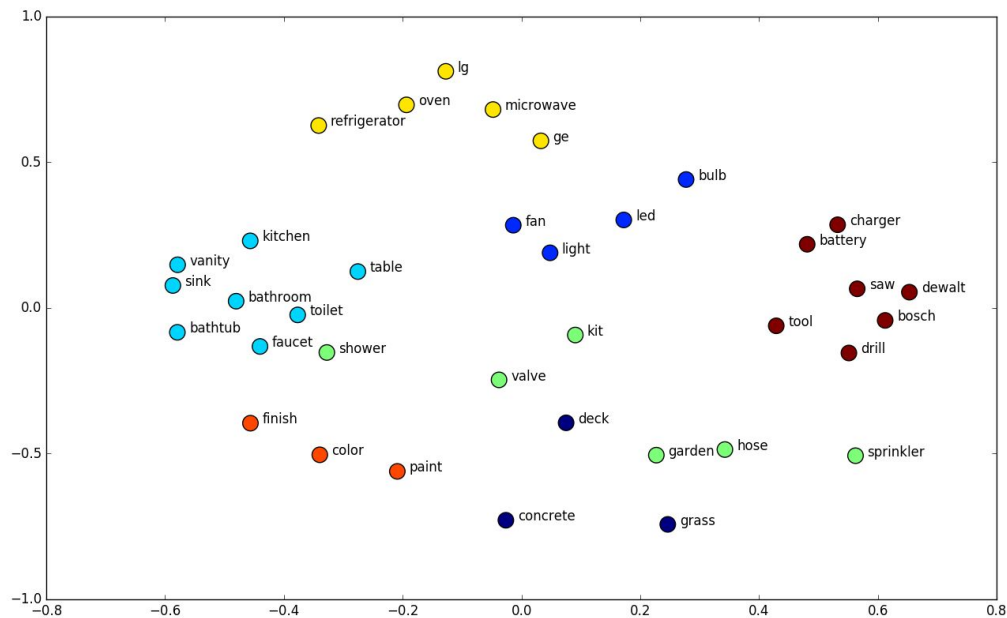
Solution: Can we convert words into a set of representative numbers that we can input?

Mapping words into a vector of real numbers that captures features of the word.

By comparing two words vectors we can also compare their relationships.

One of the most popular models is Word2Vec (but others exist e.g. GloVe).

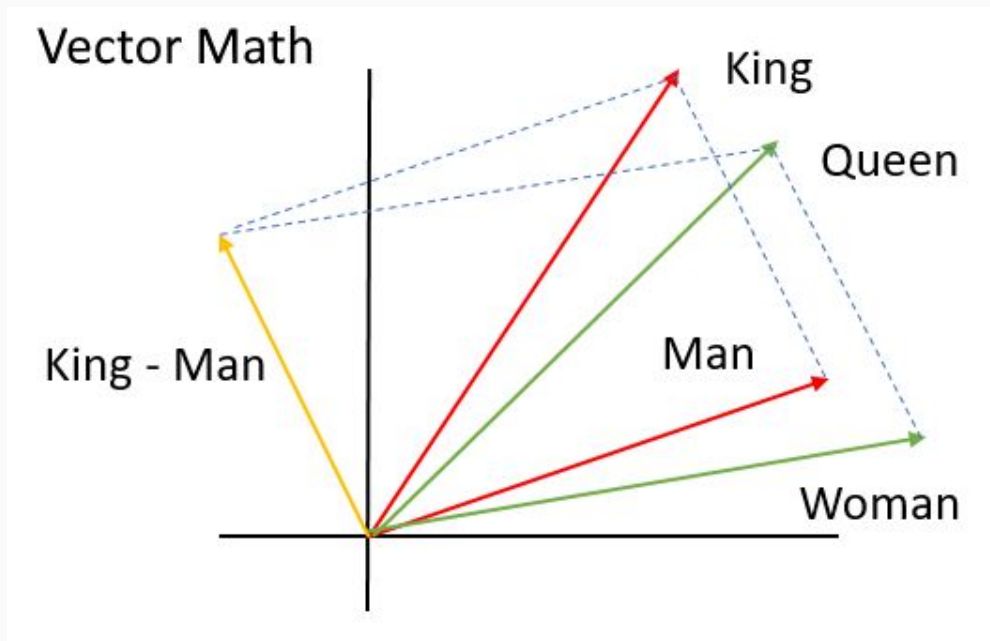
Word Embeddings (cont.)



Embeddings of similar words are grouped together.

Word Embeddings (cont.)

We can also derive relationships between words which is interesting!



$$\text{King} - \text{Man} + \text{Woman} = \text{Queen}$$

Word2Vec

Many ways to compute embeddings but Word2Vec uses a neural network.

Two architectures for Word2Vec depending on task:

1. **Continuous Bag-of-Words Model (CBOW)**

- Predicts a word given its context.
- E.g. Context = "he drives his _ to work" -> Word = "car"

2. **Skip-Gram Model**

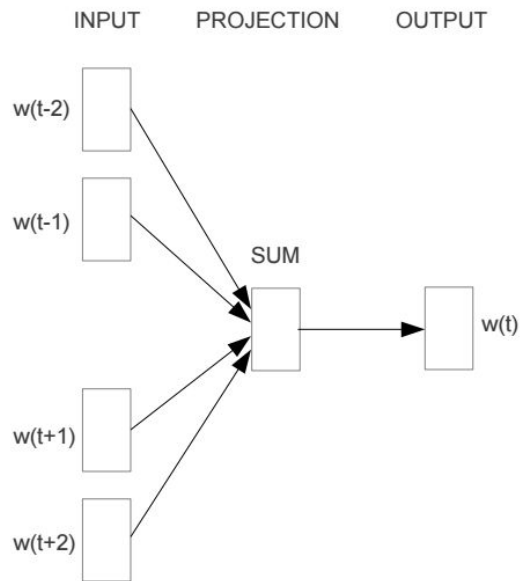
- Predicts a context given a word.
- E.g. Word = "car" -> Context = "what you drive to work"

Skip Gram is usually better for limited training data but takes longer to train.

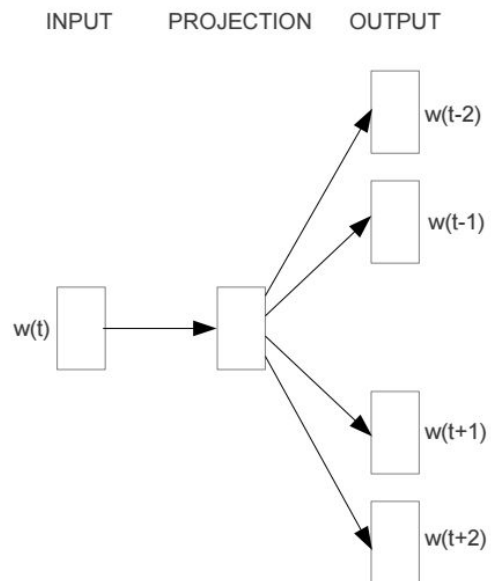
*You shall know a word by the
company it keeps.*

- Firth (1957)

Word2Vec (cont.)



CBOW



Skip-gram

CBOW vs Skip Gram Architectures

Word2Vec (cont.)

There is no need to train models for your own data.

You can load and use pre-trained Word2Vec models available online which are trained on enormous datasets for days.

One example is Google's Word2vec models trained on all the articles in Google News.

Another example is a Word2Vec model trained on all of Wikipedia.

Recurrent Neural Networks (RNN)

Recurrent Neural Networks

Fully connected feedforward networks take in a fixed size input and have no notion of order in time.

But sequences (e.g. text) aren't a fixed size and order matters.

We need a way to remember and reason about what came before an item in a sequence.

Solution: Recurrent Neural Networks (RNN).

RNNs are “recurrent” because they perform similar computation for every item in a sequence but the output depends on the previous sequence items' computations.

Intuition: Encoding a memory about what you have calculated so far.

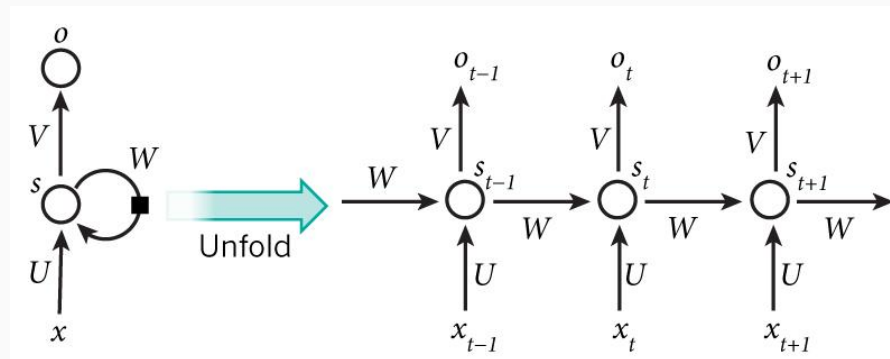
Recurrent Neural Networks (cont.)

Decision of a network at time $t-1$ affects the decision made at time t .

Two sources of input to the network: The past (output of previous state) and the present.

Sequential information (long-term dependencies) is preserved through the networks hidden state at each time step.

Unlike feed forward networks, we also use the hidden state at the previous time step when computing the output.



Recurrent Neural Networks (cont.)

$$\mathbf{h}_t = f(\mathbf{U} * \mathbf{x}_t + \mathbf{W} * \mathbf{h}_{t-1})$$

$$\mathbf{o}_t = \text{softmax}(\mathbf{V} * \mathbf{h}_t)$$

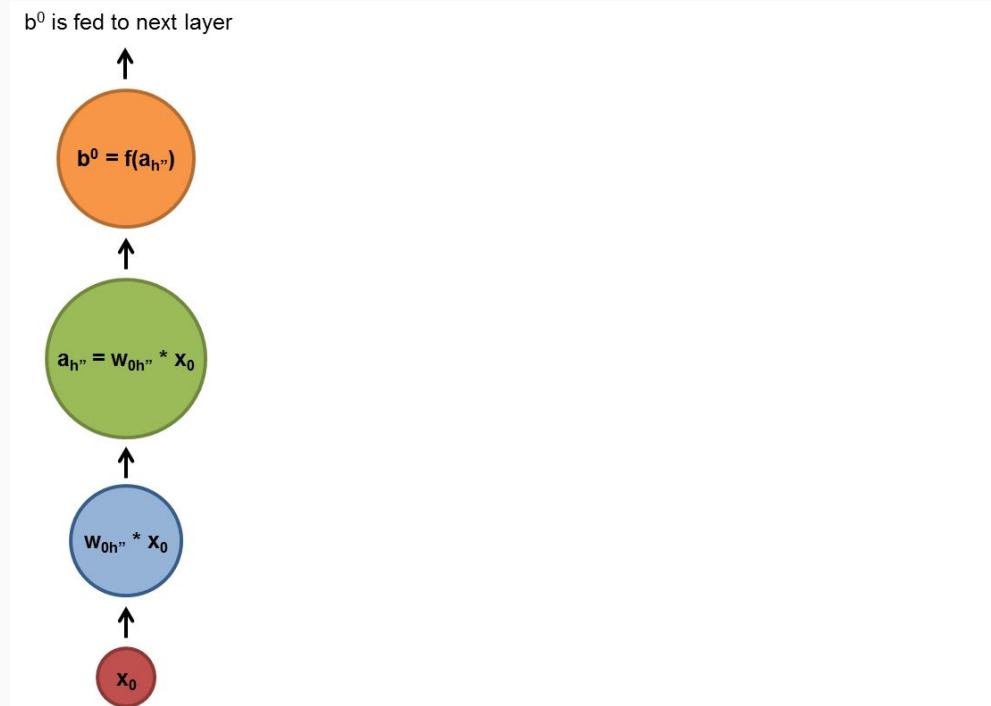
Where the hidden state \mathbf{h}_t is a function of the input \mathbf{x}_t multiplied by a weight matrix \mathbf{U} added to the previous hidden state \mathbf{h}_{t-1} multiplied by \mathbf{W} which is a hidden-to-hidden state matrix (essentially a transition matrix).

The output at the step \mathbf{o}_t is determined by the weight matrix of the hidden state \mathbf{V} . Activation function is f .

\mathbf{V} , \mathbf{W} , and \mathbf{U} are shared throughout the entire network.

Since we compute this at every time step t , every hidden state has some traces of information about all the time steps that came before it.

Recurrent Neural Networks (cont.)



Unrolling a recurrent Neural Network. From DeepLearning4j.

Recurrent Neural Networks (cont.)

Unlike feed forward neural networks, RNNs are harder to train (more unstable).

They use an extension of backpropagation called Backpropagation through Time (BPTT).

They are more unstable because adding the connected time steps adds more functions to calculate the derivatives of with the chain rule.

In theory RNNs should be able to capture all long term dependencies but in practice they don't always work well.

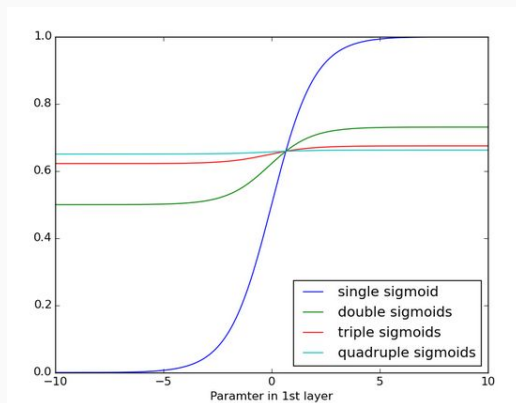
Long Short-Term Memory (LSTM)

Long Short-Term Memory Models (LSTM)

Inspiration: Due to the computation occurring over a long period of time in RNNs, they suffer from the vanishing gradient problem.

Since information in the network passes through several steps of computation, a small change at the end of the network can cause an explosive (solvable) or negligible change in the beginning of the network making it unstable. (Think of compound interest)

LSTMs were developed to solve the vanishing gradient problem in RNNs.



From DeepLearning4j.

LSTM (cont.)

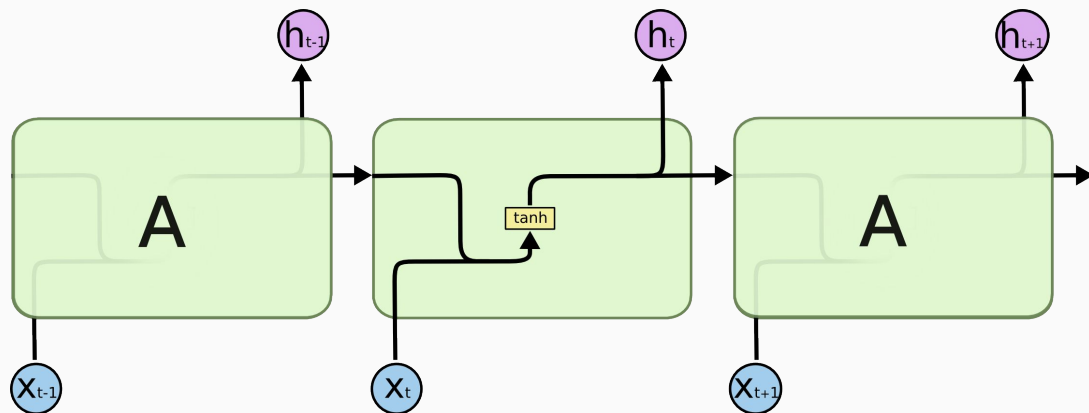
LSTMs were invented by Hochreiter and Schmidhuber to solve the vanishing gradient problem.

Aims to solve the long term dependency gap that RNNs can't handle well.

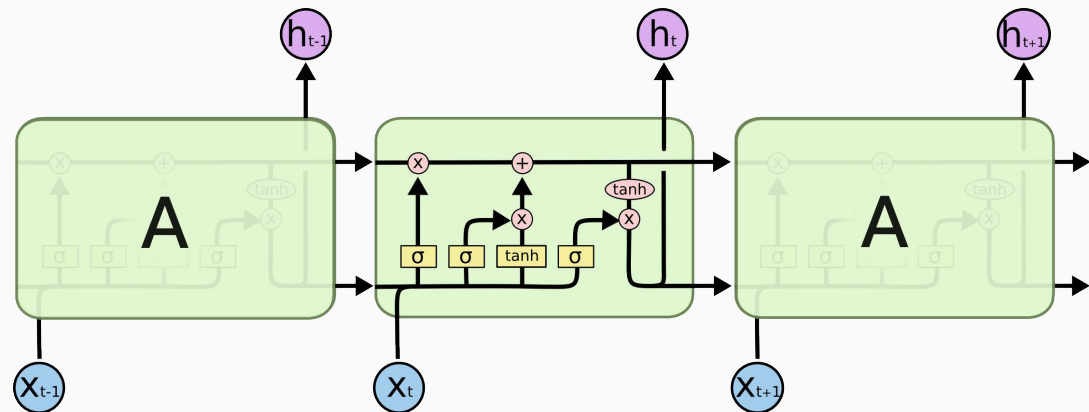
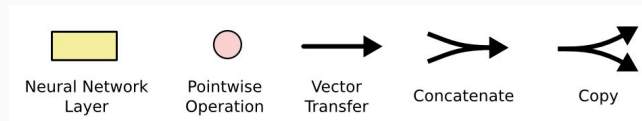
For example “the clouds are in the *sky*” is easy but “I grew up in France... I speak fluent *French*” is much more difficult.

Instead of a single neural network layer in an RNN, and LSTM has four interacting together.

LSTM (cont.)



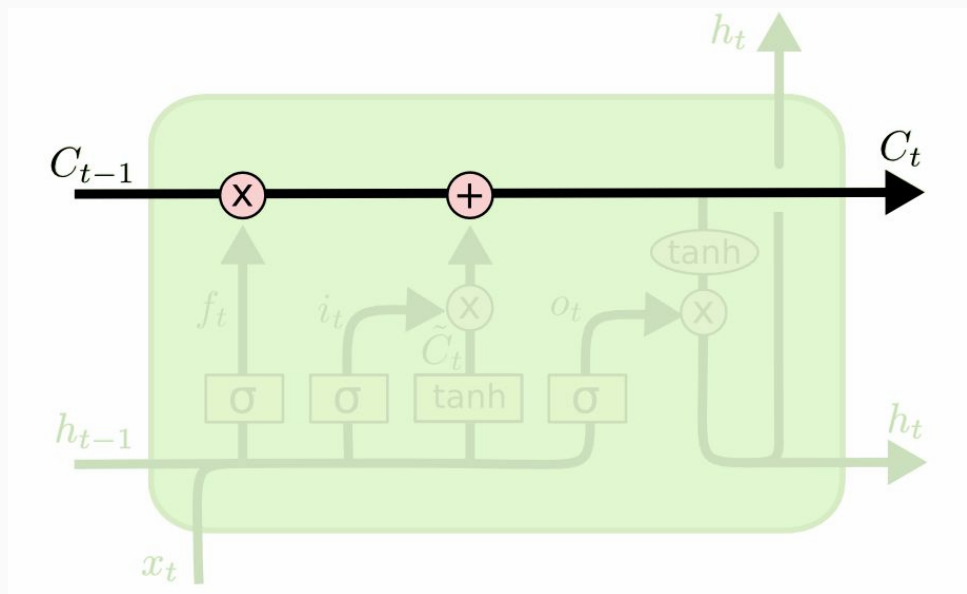
Module in an RNN has one layer.



Module in an LSTM has four layers.

LSTM (cont.)

Cell state progresses throughout the entire network and is what contains the information about the sequence.



LSTM (cont.)

LSTM controls the information added to the cell state using three gates.

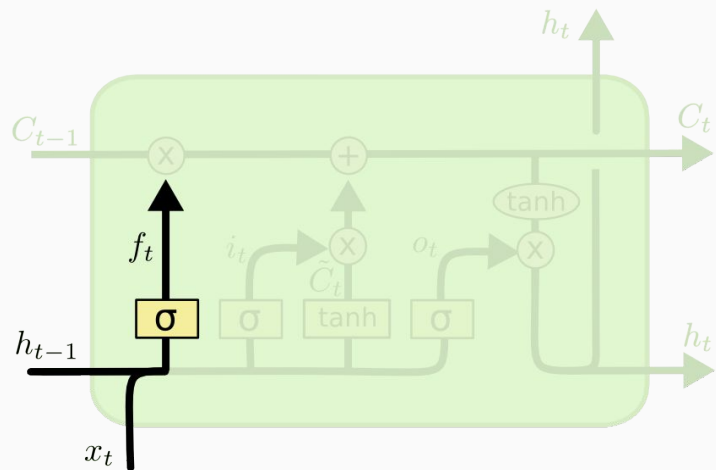
Gates are based off of the *sigmoid* function indicating how much of the information to let through (where 0 means none up to 1 which means let all the data in).

This helps control the flow of data and fix the vanishing gradient problem.

LSTM (cont.)

LSTMs have forget gates which decide what information to drop from the previous layer based on \mathbf{h}_{t-1} and \mathbf{x}_t . Outputs a number between 0 and 1 for each number in \mathbf{C}_{t-1} .

Intuitively you can think of this as forgetting about an old subject when encountering a new one.



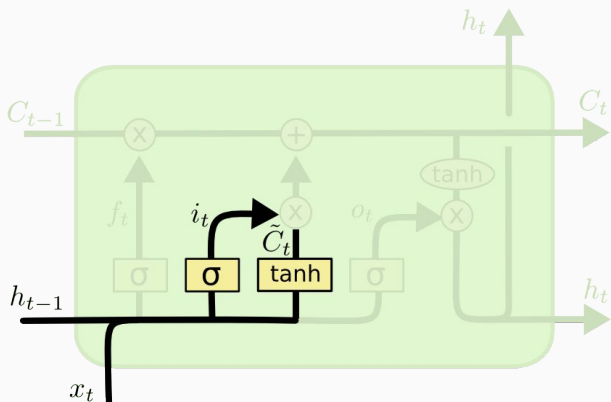
$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

LSTM (cont.)

LSTMs can also decide what new information to store in \mathbf{C}_t .

It combines the output of the *sigmoid* input gate layer which decides which values to update with a *tanh* layer that creates a vector of candidate values to add to the state.

Intuitively this is like adding properties of the new subject that we want to remember (e.g. gender, plural or singular, etc.)

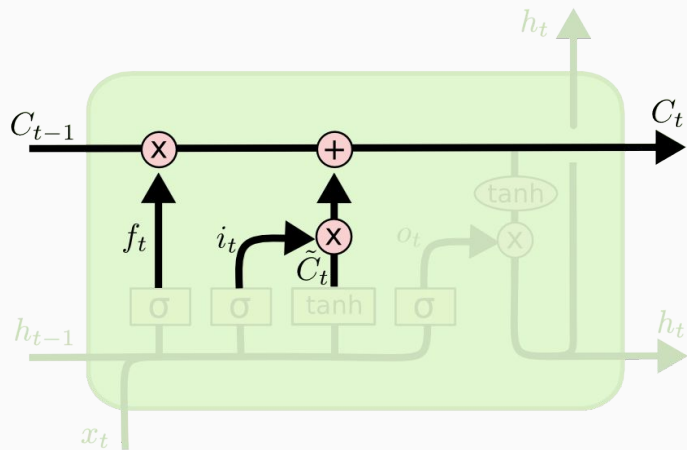


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

LSTM (cont.)

To update \mathbf{C}_{t-1} we multiply it by the forget output \mathbf{f}_t and add $\mathbf{i}_t * \tilde{\mathbf{C}}_t$.

$\mathbf{i}_t * \tilde{\mathbf{C}}_t$ scales how much we want to update new candidate values.

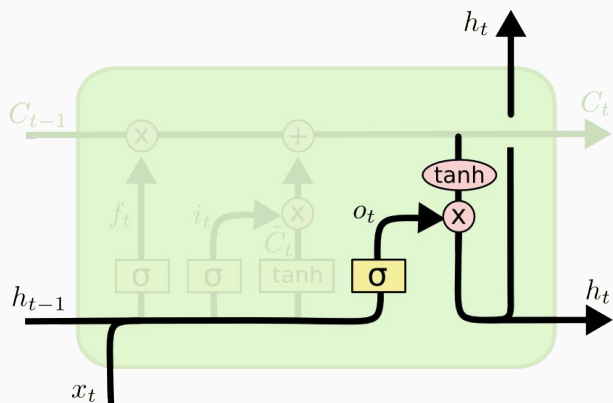


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM (cont.)

To choose what parts of the cell state we want to output as the hidden state to the next layer, we filter it using a combination of a sigmoid layer and multiply it by the *tanh* of the cell state (to squeeze values between -1 and 1).

The goal of this is to output “hidden” information related to the current context (i.e. if the subject is singular or plural, gender, etc.).



$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

LSTM (cont.)

This interaction of gates allows the vanishing gradient problem to be solved for RNNs.

LSTMs are currently the state of the art in sequence based tasks and have many extensions such as Seq-2-Seq and attention based models.

Attention based models allow the network to pick and choose what specifically to focus on in depth when performing computation.

Announcements

Announcements

Final project proposals due **April 8th**.

Practical 3 due **April 6th**.

Midterm will take place on **April 13th** and cover everything up till the lecture next week.

Please submit weekly feedbacks.

Final Project

Goal is to build a meaningful project that demonstrates your knowledge of what we have learnt. Extremely open ended and creativity is encouraged.

Requirements:

1. You must use at least one deep learning model in your project.
2. You may not repeat any of the practicals as the project nor copy projects or tutorials online (we will have people checking this seriously).
3. You must write code to perform the task and have some measurable results.

You may work in teams of up to 3.

Graded on proposal, creativity, checkpoint, final write up, code + explanations, and video presentation, and in-class presentation.

Final Project Proposal

Proposals are due by **April 8th**.

Please submit a slide deck presentation (pdf) outlining your proposal.

It should contain:

1. Team Members and Project Title (doesn't have to be final)
2. What do you want to build and why?
3. Where you will get the data?
4. What models do you think you will use?
5. How can you measure your results?

Proposal is worth 10% of your final project grade.

Questions?