Deep Learning Methods for Text Classification

Garrett Hoffman, Senior Data Scientist @ StockTwits
Sentiment Analysis Symposium 2018
March 26, 2018

Talk Overview

- Learning Distributed Representations of Words with Word2Vec
- Recurrent Neural Networks and their Variants
- Convolutional Neural Networks for Language Tasks
- Practical Considerations for Modeling with Your Data

Learning Distributed Representations of Words with Word2Vec

Sparse Representation

A sparse, or one hot, representation is where we represent a word as a vector with a 1 in the position of the words index and 0 elsewhere

Sparse Representation

Let's say we have a vocabulary of 10,000 words

V = [a, aaron, ..., zulu, <UNK>]

Man (5,001) = [0 0 0 0 ... 1 ... 0 0]

Sparse Representation

Let’s say we have a vocabulary of 10,000 words

V = [a, aaron, …., zulu, <UNK>]

Man (5,001) = [0 0 0 0 … 1 … 0 0]

Woman (9,800) = [0 0 0 0 0 … 1 … 0]

Sparse Representation

Let's say we have a vocabulary of 10,000 words

V = [a, aaron, …., zulu, <UNK>]

Man (5,001) = [0 0 0 0 … 1 … 0 0]

Woman (9,800) = [0 0 0 0 0 … 1 … 0]

King (4,914) = [0 0 0 … 1 … 0 0 0]

Queen (7,157) = [0 0 0 0 … 1 … 0 0]

Sparse Representation

Let’s say we have a vocabulary of 10,000 words

V = [a, aaron, …., zulu, <UNK>]

Man (5,001) = [0 0 0 0 … 1 … 0 0]

Woman (9,800) = [0 0 0 0 0 … 1 … 0]

King (4,914) = [0 0 0 … 1 … 0 0 0]

Queen (7,157) = [0 0 0 0 … 1 … 0 0]

Great (3,401) = [0 … 1 … 0 0 0 0 0]

Wonderful (9,805) = [0 0 0 0 0 … 1 … 0]

Sparse Representation Drawbacks

- The size of our representation increases with the size of our vocabulary

Sparse Representation Drawbacks

- The size of our representation increases with the size of our vocabulary
- The representation doesn’t provide any information about how words relate to each other

Sparse Representation Drawbacks

- The size of our representation increases with the size of our vocabulary
- The representation doesn’t provide any information about how words relate to each other

  - E.g. “Metis is great!” vs. “Metis is wonderful!”

Distributed Representation

A distributed representation is where we represent a word as a prespecified number of latent features that each correspond to some semantic or syntactic concept

Distributed Representation

	Gender
Man	-1.0
Woman	1.0
King	-0.97
Queen	0.98
Great	0.02
Wonderful	0.01

Distributed Representation

	Gender	Royalty
Man	-1.0	0.01
Woman	1.0	0.02
King	-0.97	0.97
Queen	0.98	0.99
Great	0.02	0.15
Wonderful	0.01	0.05

Distributed Representation

	Gender	Royalty	...	Polarity
Man	-1.0	0.01	...	0.02
Woman	1.0	0.02	...	-0.01
King	-0.97	0.97	...	0.01
Queen	0.98	0.99	...	-0.02
Great	0.02	0.15	...	0.89
Wonderful	0.01	0.05	...	0.94

Word2Vec

One method used to learn these distributed representations of words using the Word2Vec algorithm

Word2Vec uses a 2-layered neural network to reconstruct the context of words

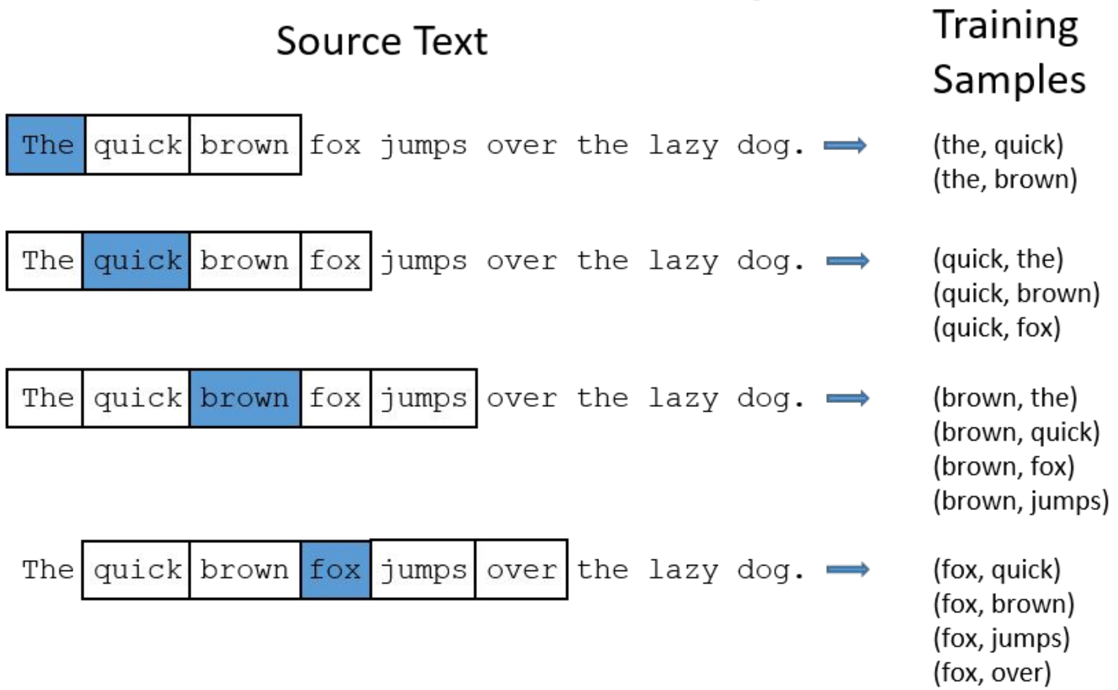
"Distributed Representations of Words and Phrases and their Compositionality", Mikolov et al. (2013)

“

you shall know a word by the company it keeps

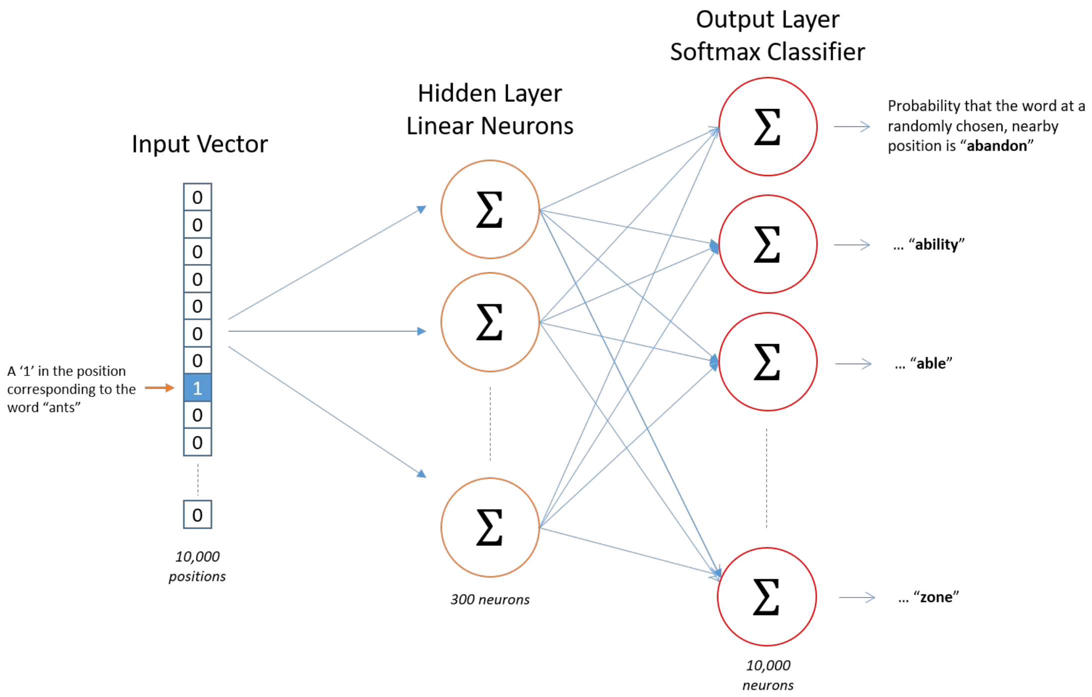
- *J.R. Firth*

Word2Vec - Generating Data



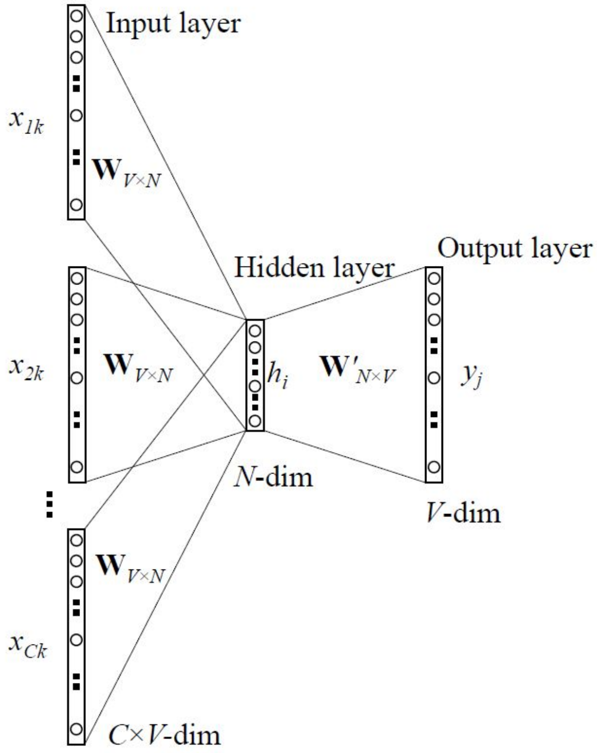
[McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model.]

Word2Vec - Skip-gram Network Architecture

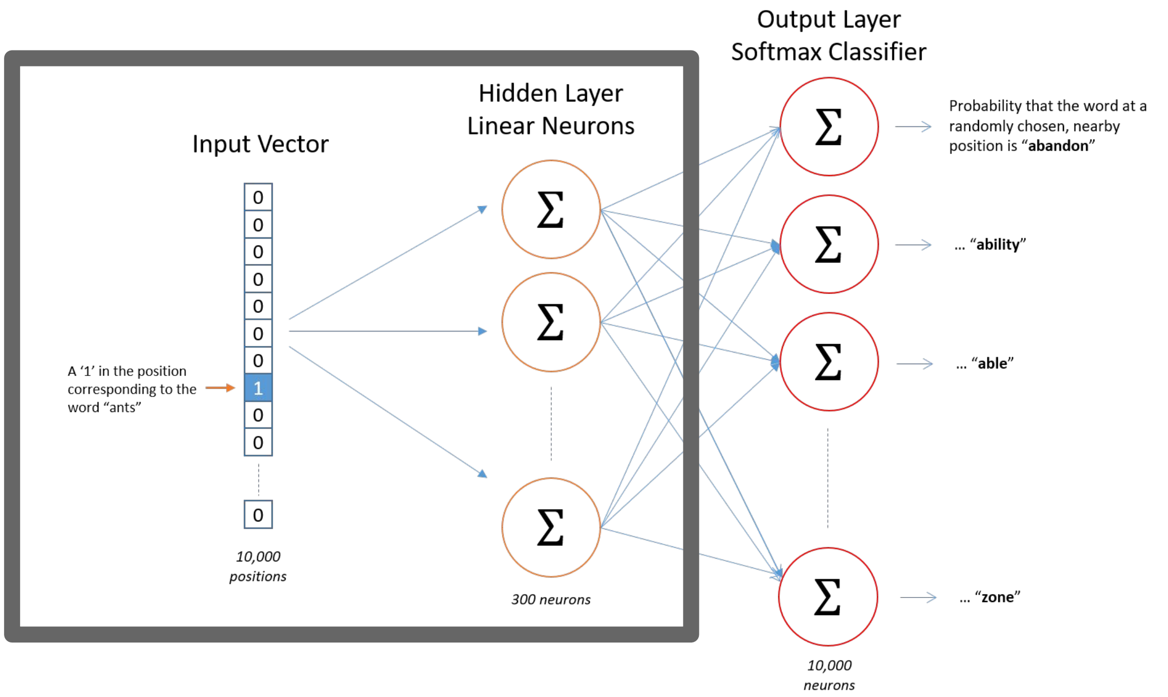


[McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model.]

Word2Vec – CBOW Network Architecture

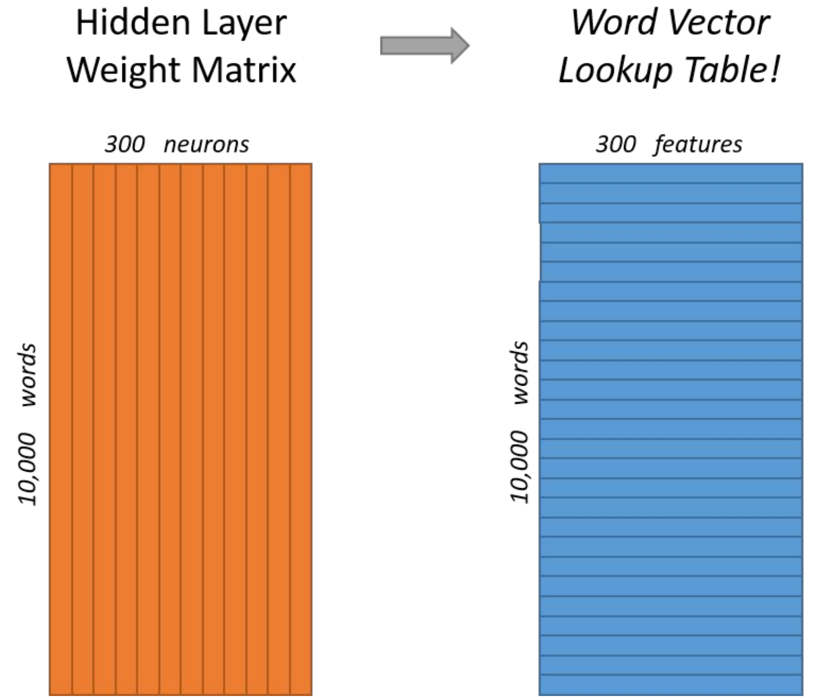


Word2Vec - Skip-gram Network Architecture



McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model.

Word2Vec - Embedding Layer



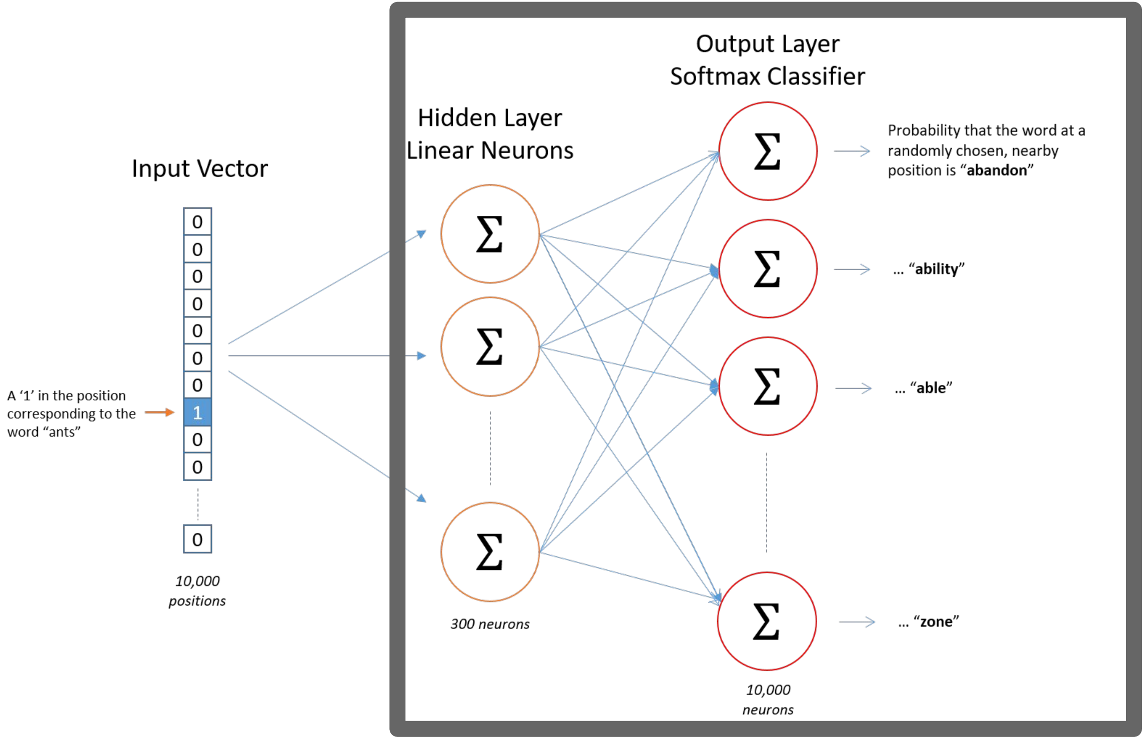
McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model.

Word2Vec - Embedding Layer

$$\begin{bmatrix} 0 & 0 & 0 & \boxed{1} & 0 \end{bmatrix} \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix}$$

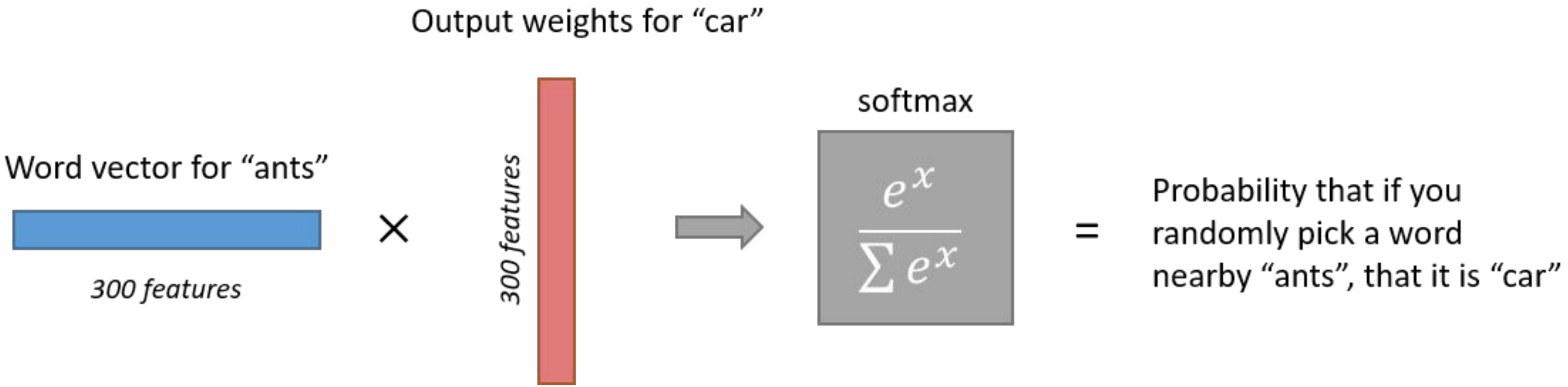
[McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model.]

Word2Vec - Skip-gram Network Architecture



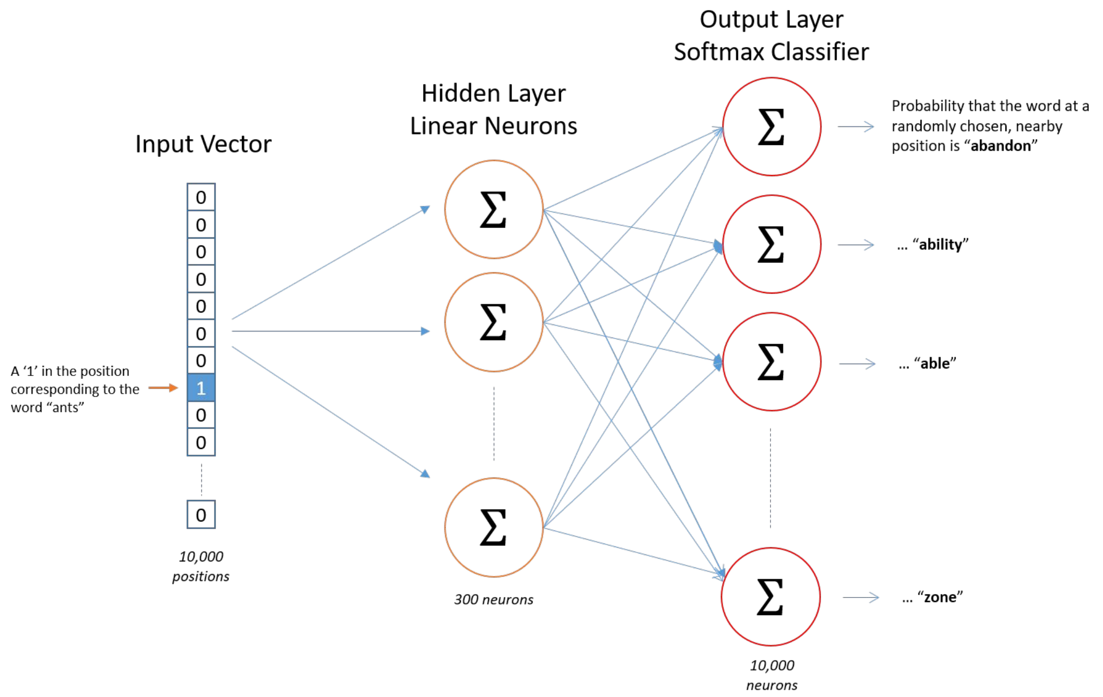
McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model.

Word2Vec - Output Layer



[McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model.]

Word2Vec - Intuition



[McCormick, C. (2017, January 11). Word2Vec Tutorial Part 2 - Negative Sampling.](#)

Word2Vec - Negative Sampling

In our output layer we have 300 x 10,000 = 3,000,000 weights, but given that we are predicting a single word at a time we only have a single “positive” output out of 10,000 output.

McCormick, C. (2017, January 11). Word2Vec Tutorial Part 2 - Negative Sampling.

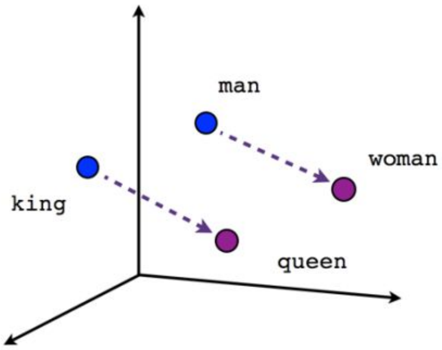
Word2Vec - Negative Sampling

In our output layer we have 300 x 10,000 = 3,000,000 weights, but given that we are predicting a single word at a time we only have a single “positive” output out of 10,000 output.

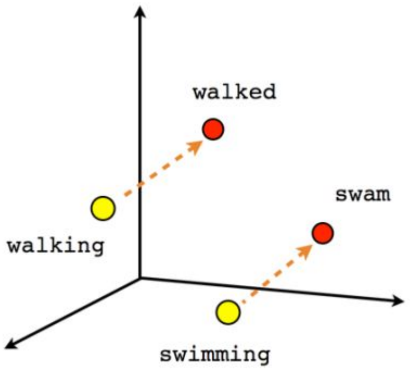
For efficiency, we will randomly update only a small sample of weights associated with “negative” examples. E.g. if we sample 5 “negative” examples to update we will only update 1,800 weights (5 “negative” + 1 “positive” * 300) weights.

[McCormick, C. (2017, January 11). Word2Vec Tutorial Part 2 - Negative Sampling.]

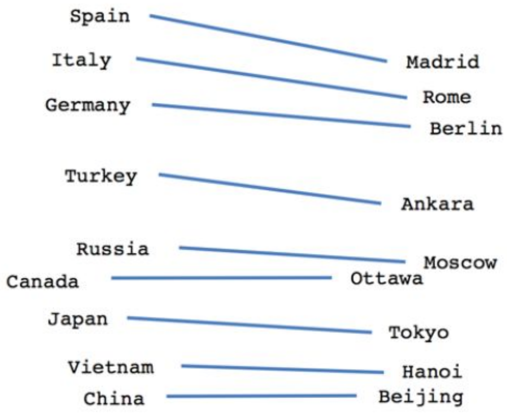
Word2Vec - Results



Male-Female



Verb tense



Country-Capital

https://www.tensorflow.org/tutorials/word2vec

Pre-Trained Word Embedding

https://github.com/Hironsan/awesome-embedding-models

```
import gensim

# Load Google's pre-trained Word2Vec model.
model =
gensim.models.KeyedVectors.load_word2vec_format('./GoogleNew
s-vectors-negative300.bin', binary=True)
```

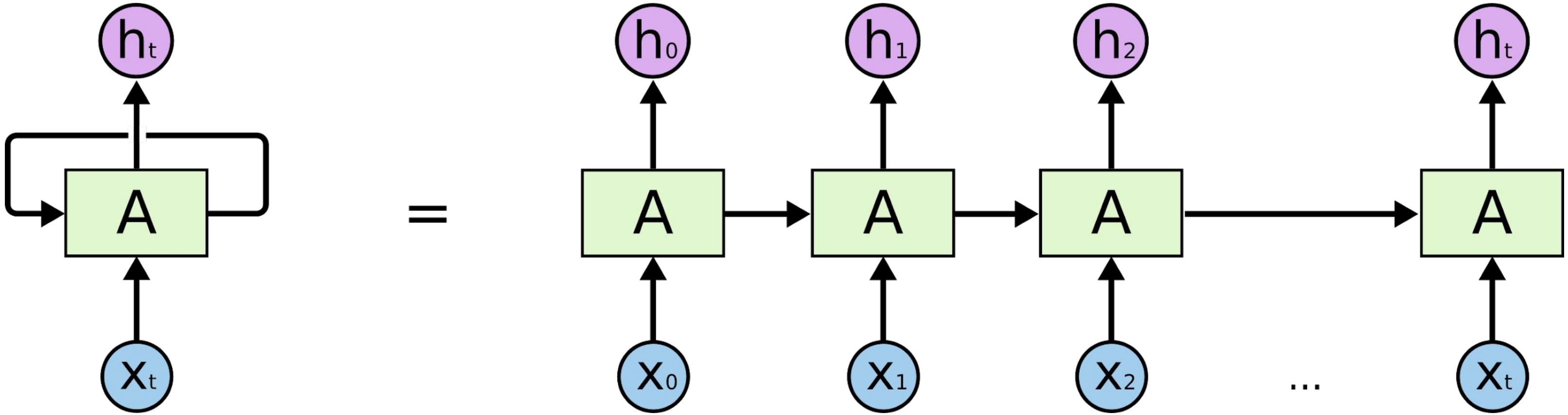
Recurrent Neural Networks and their Variants

Sequence Models

When dealing with text classification models, we are working with sequential data, i.e. data with some aspect of temporal change

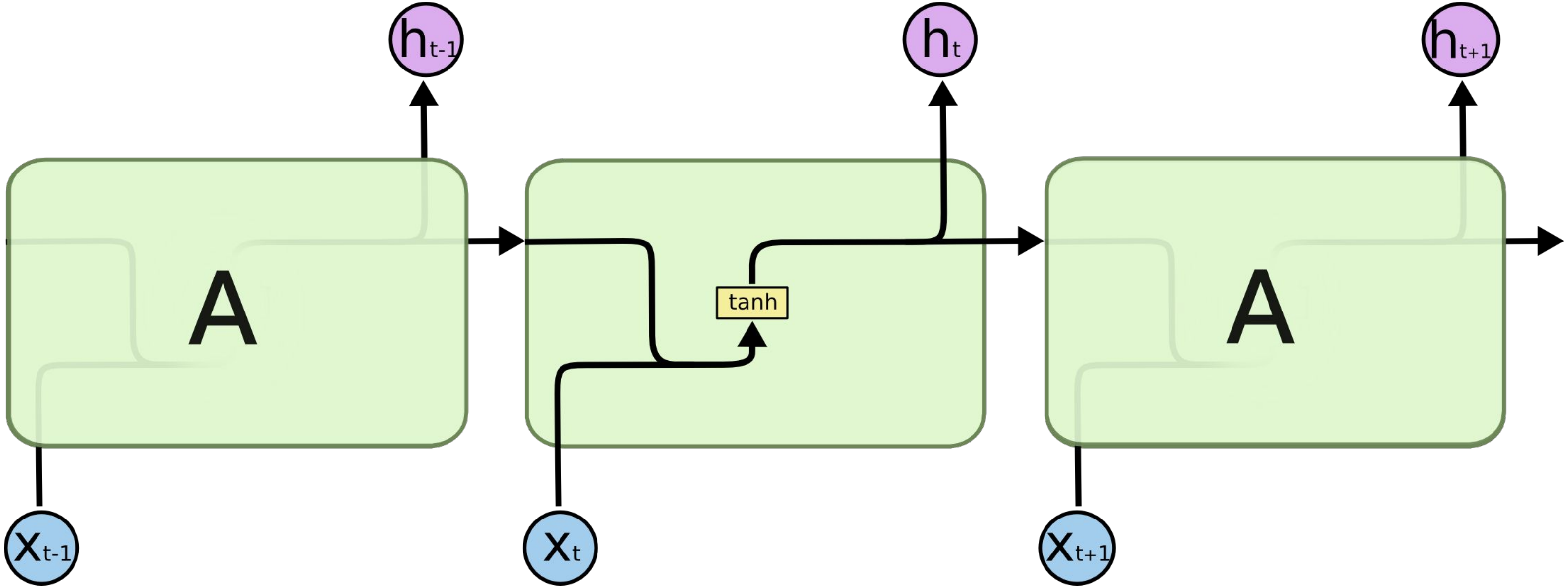
We are typically analyzing a sequence of words and our output can be a single value (e.g. sentiment classification) or another sequence (e.g. language translation, entity recognition)

Recurrent Neural Networks (RNNs)



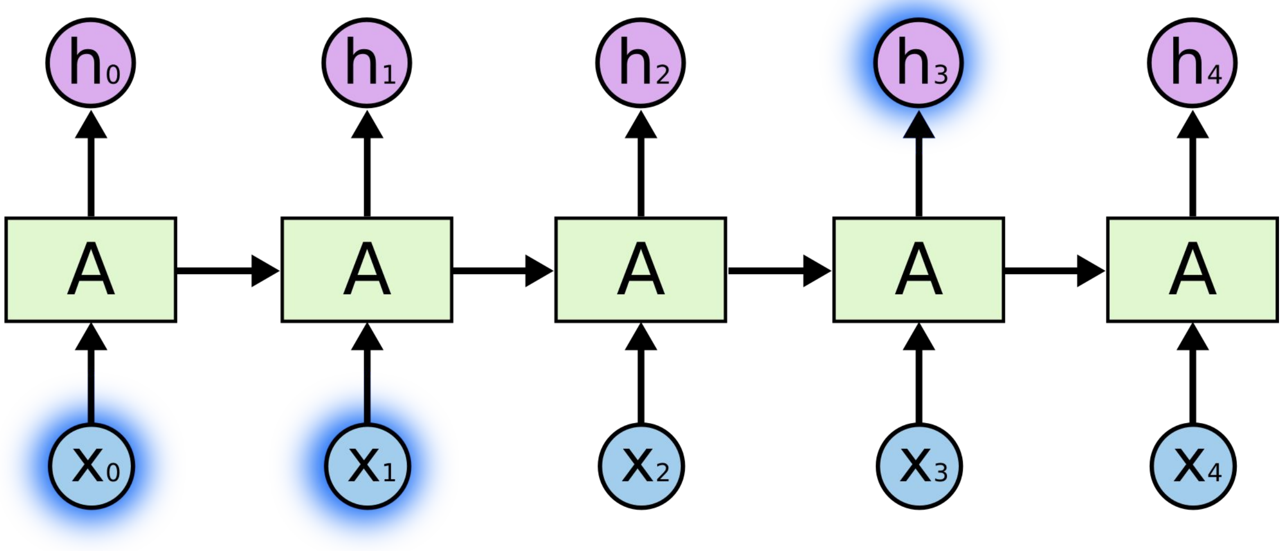
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks (RNNs)



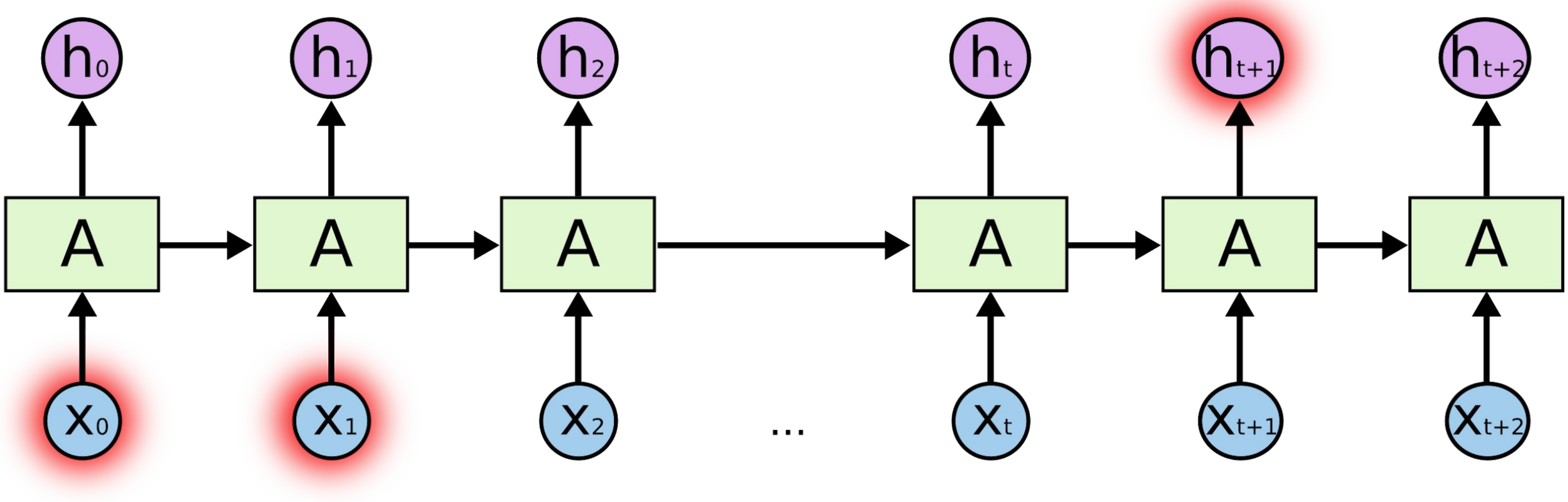
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Networks (RNNs)



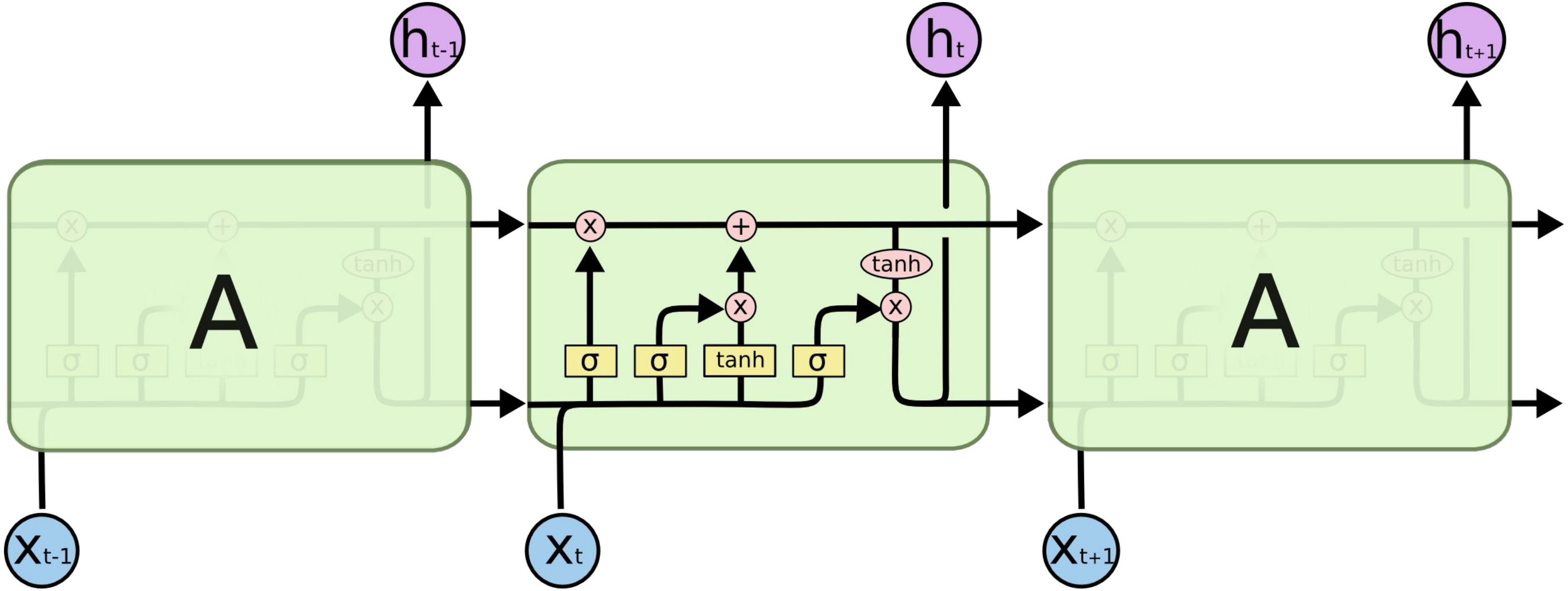
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Term Dependency Problem



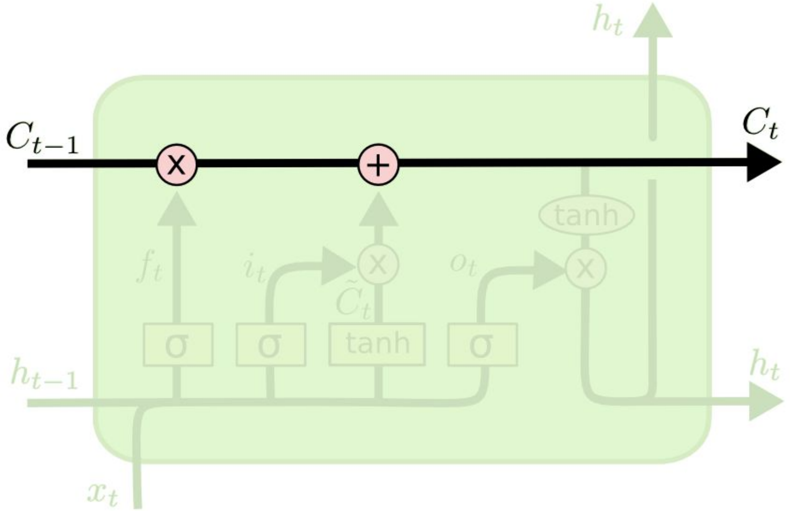
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTMs)



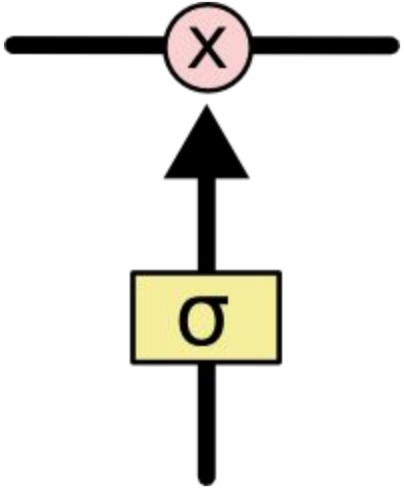
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTMs)



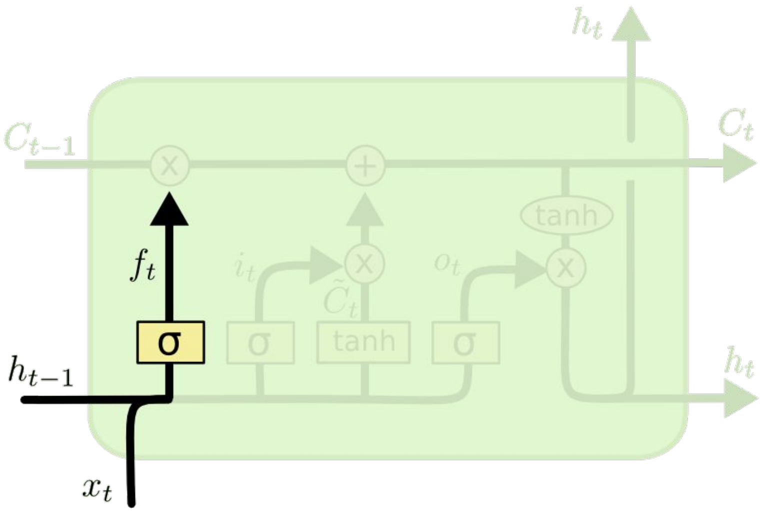
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTMs)



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

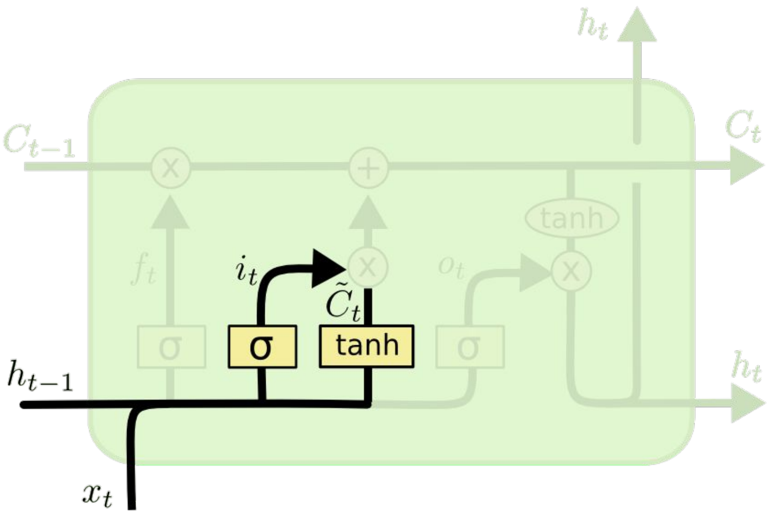
LSTM - Forget Gate



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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM - Learn Gate

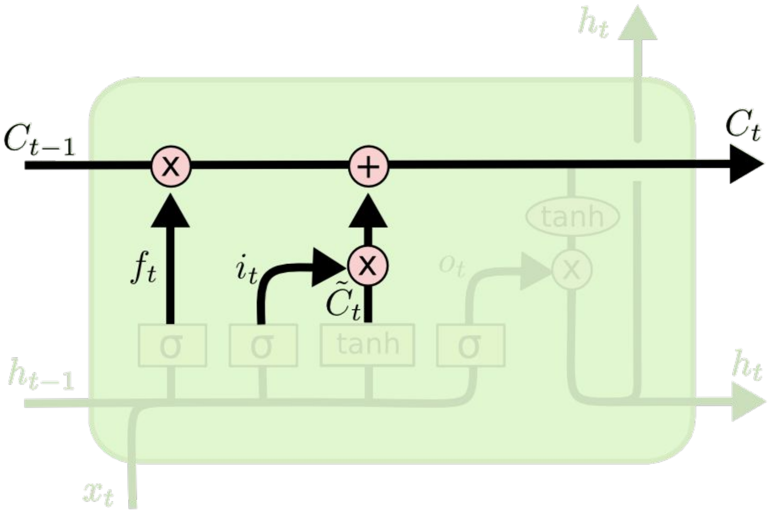


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

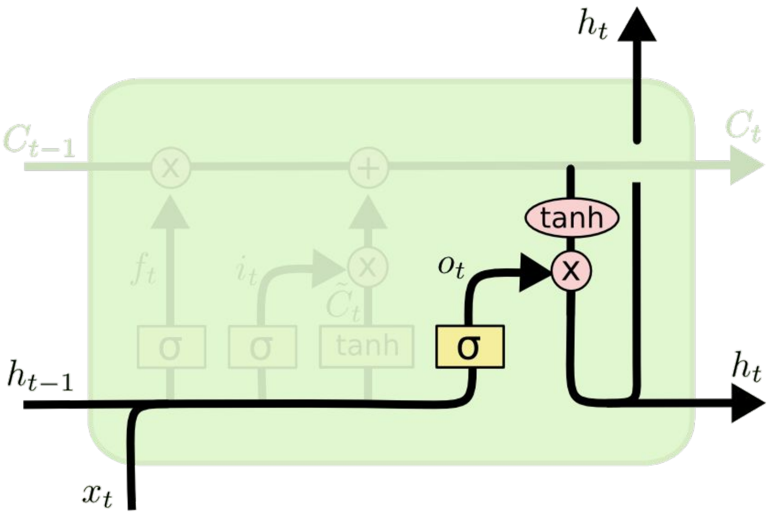
LSTM - Update Gate



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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM - Output Gate

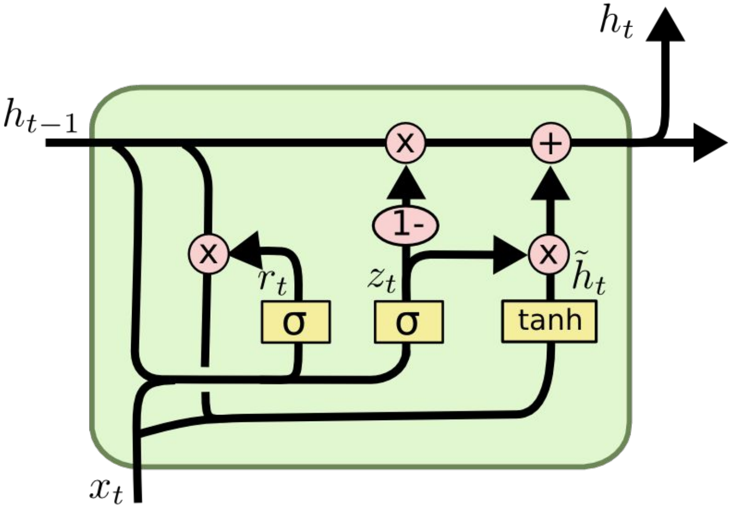


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

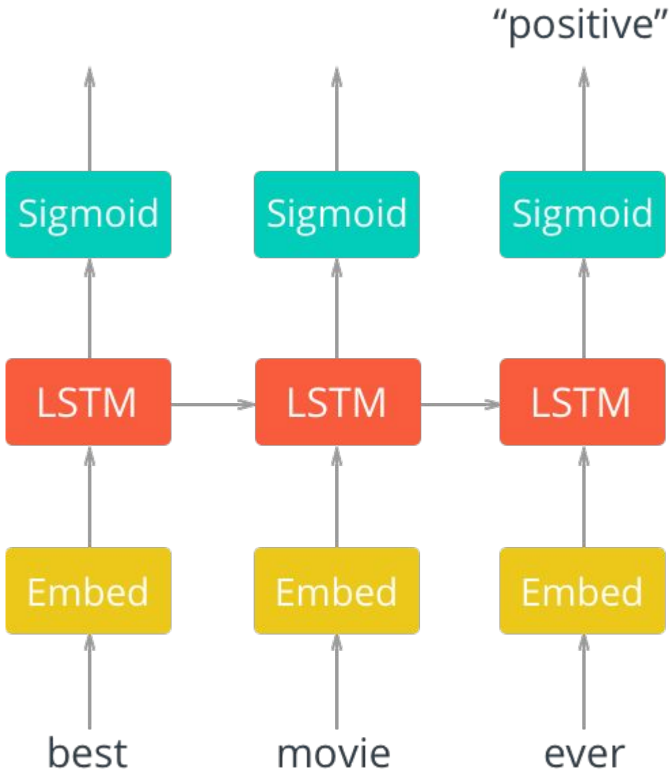
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Network Architecture

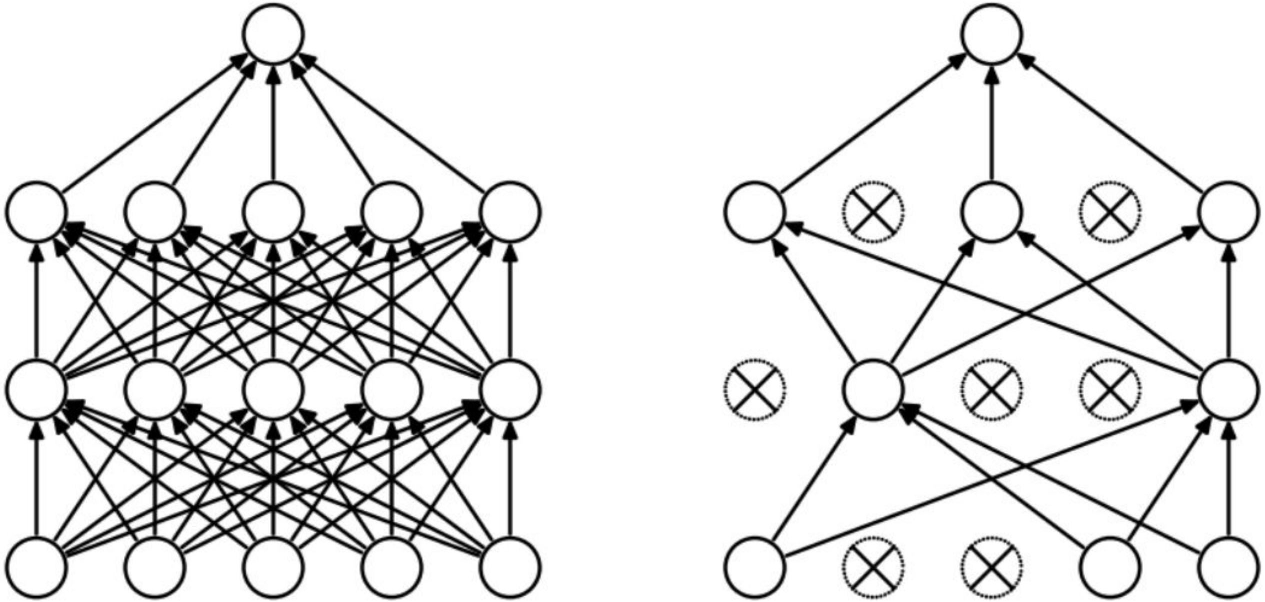


Learning Embeddings End-to-End

Distributed representations can also be learned in an end-to-end fashion as part of the model training process for an arbitrary task.

Trained under this paradigm, distributed representations will specifically learn to represent items as they relate to the learning task.

Dropout



(a) Standard Neural Net

(b) After applying dropout.

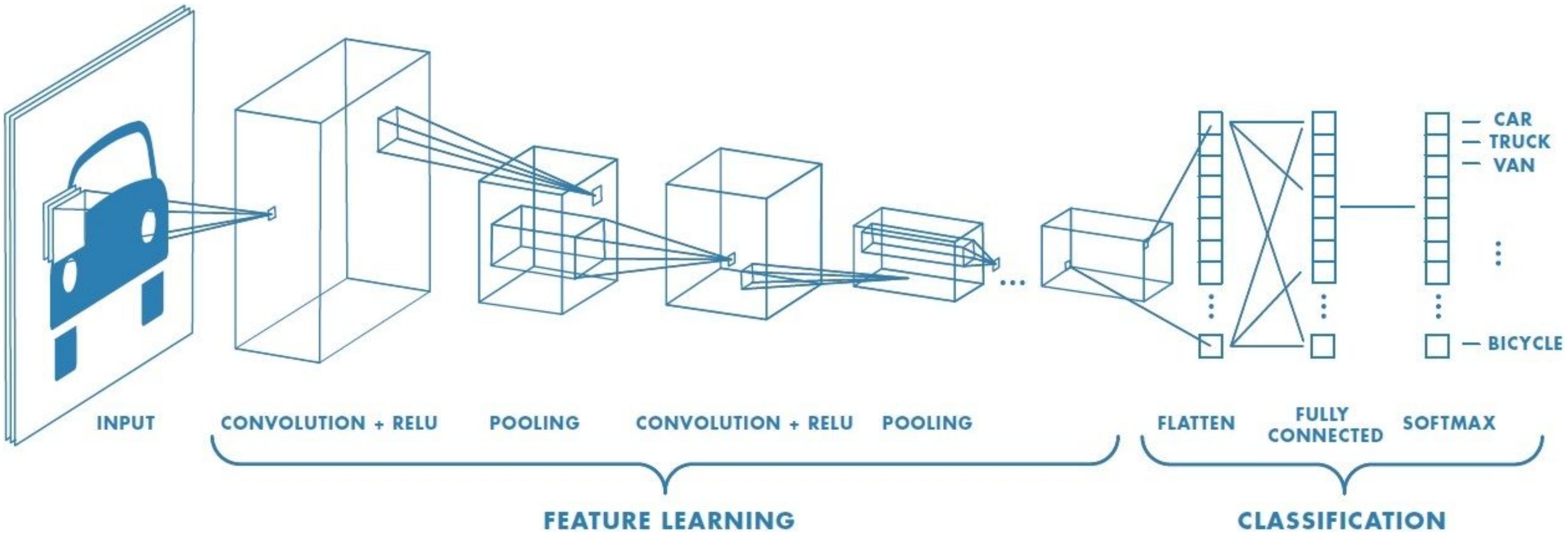
Convolutional Neural Networks for Language Tasks

Computer Vision Models

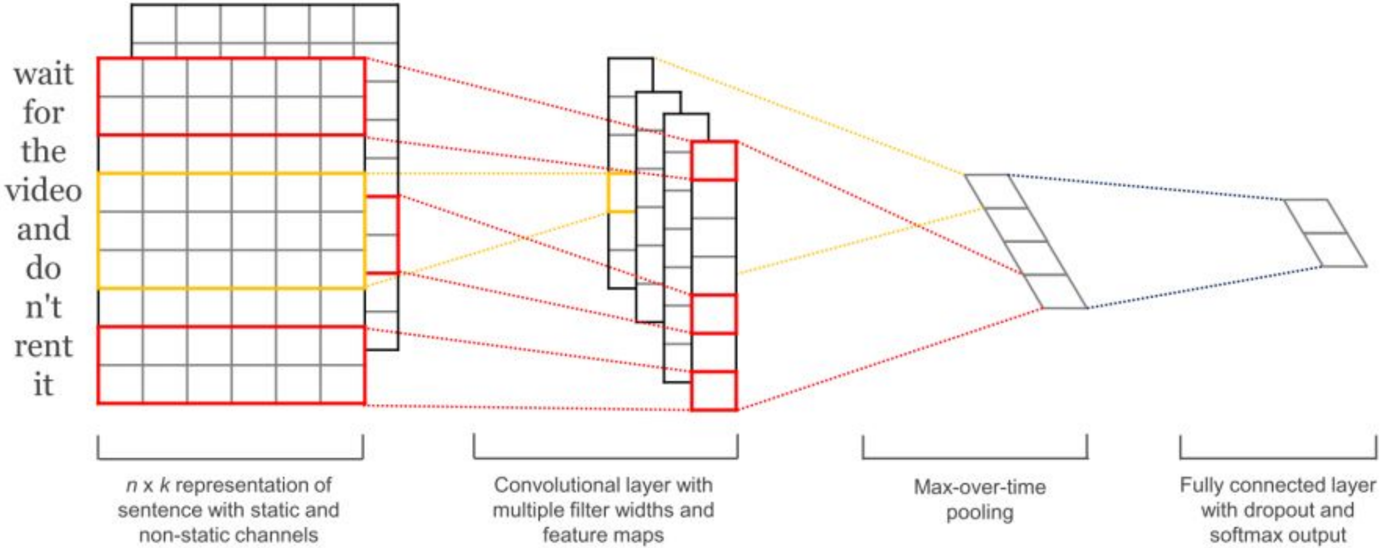
Computer Vision (CV) models are used for problems that involve working with image or video data - this typically involves image classification or object detection.

The CV research community has seen a lot of progress and creativity over the last few year - ultimately inspiring the application of CV models to other domains.

Convolutional Neural Networks (CNNs)



CNN Architecture for Text



Practical Considerations for Modeling with Your Data

Practical Considerations

- Data, data, data

Practical Considerations

- Data, data, data
- Subject Matter and Domain Specific Lexicon

Practical Considerations

- Data, data, data
- Subject Matter and Domain Specific Lexicon
- Changing Lexicon over Time

Thanks!

Any questions?

You can find me at

- @garrettleeh
- garrett@stocktwits.com

and related resources at

- https://github.com/GarrettHoffman/SAS2018_DL_4_NLP
- www.oreilly.com/people/d3807-garrett-hoffman