



# Intro to Neural Nets

RNNs for Text

# Today's Agenda

## Background on NLP

- Use Cases
- Quick review on bag of words approaches, etc.

## TextVectorization Layer

- This implements basic standardization and punctuation removal. It assumes 1-grams, then one-hot encodes.
- No stemming or stop word removal, by default.

## Sequence vs. Bag-of-Words

- Conceptually

## Architectures for Sequences

- Bidirectional LSTM



# Quick Review of NLP Concepts

## Pre-processing Text

- Lower-casing, stop word removal, stemming, removing punctuation, stripping rare tokens, etc.
- Tokenization (this may be chars, words, sentences, etc.
- Integer encoding / indexing the tokens.
- Finally, I may or may not leverage sequence information.
- *Q: what is a bag of words approach? What are n-grams?*

	Database	SQL	Index	Regression	Likelihood	linear
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	43	31	20	0	3	0
D6	2	0	0	18	7	6
D7	0	0	1	32	12	0
D8	3	0	0	22	4	4
D9	1	0	0	34	27	25

# Weighting Term-Documents: TF-IDF

**Not all phrases are of equal importance...**

- E.g., David less important than Beckham
- If a term occurs all the time, observing its presence is less informative

**Inverse-document frequency (IDF) helps address this.**

$$\text{IDF} = \log(N/n_j)$$

- Term 'weighting' is then calculated as Term Frequency (TF) x IDF
- $n_j$  = # of docs containing the term,  $N$  = total # of docs
- A term is deemed important if it has a high TF and/or a high IDF.
- As TF goes up, the word is more common generally. As IDF goes up, it means very few documents contain this term.

# TextVectorization Layer

## Pre-processing Text

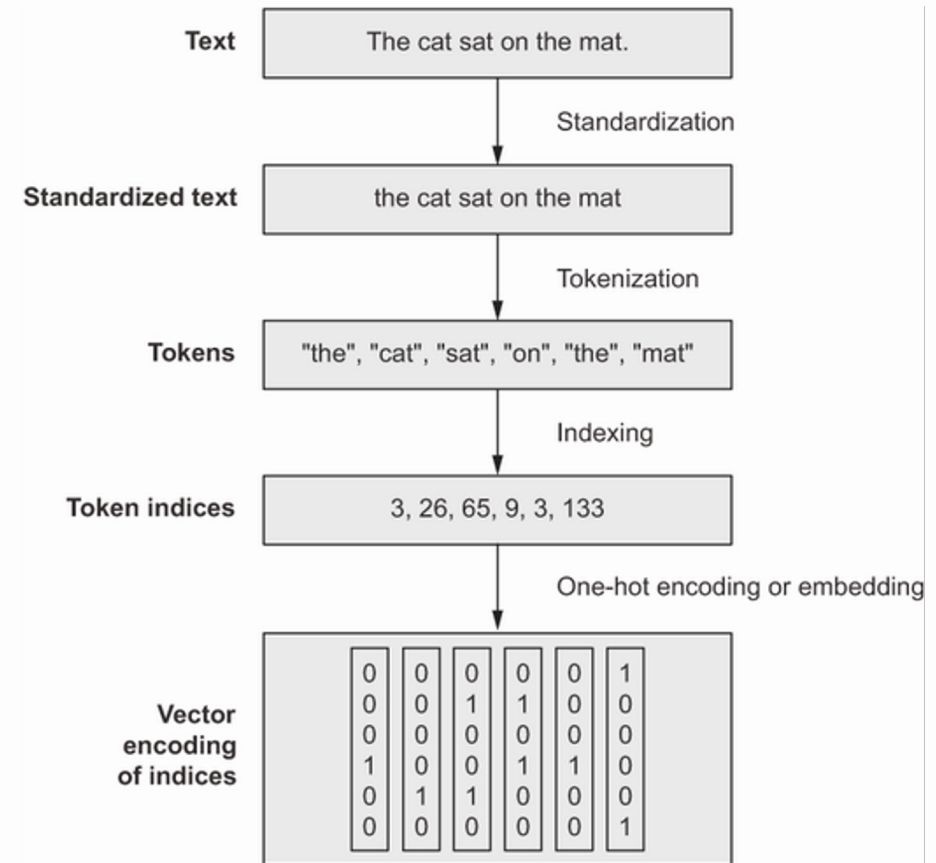
- Standardization, tokenization (words), one-hot-encoding / vectorization.
- The Keras TextVectorization() layer achieves these steps quickly.

## Customization

- You can work with n-grams, and do other sorts of pre-processing, using arguments.

## Options

- Include as part of TF Dataset pipeline (more efficient)
- Include as a layer in your Keras model.



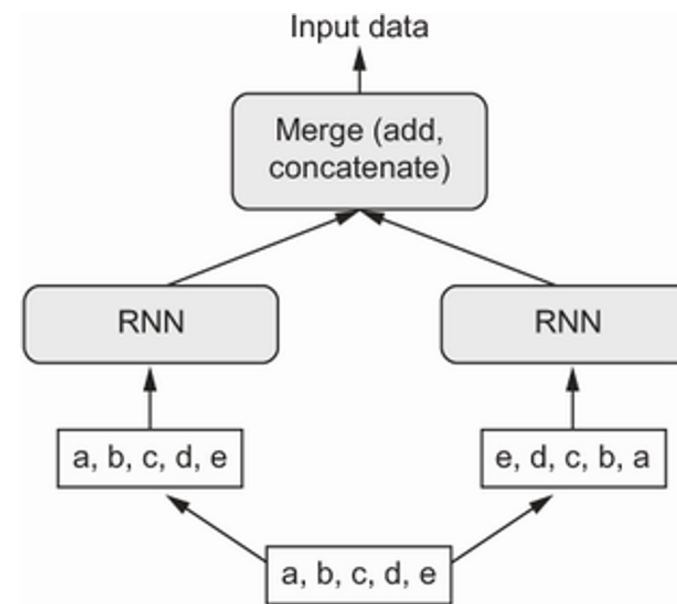
# Bidirectional LSTM

## We Saw This Last Time

- Take each sequence as input data, as well as a flipped/reversed copy.
- Was state of the art for text processing until relatively recently (transformers now dominate).

## Instead of Time Series We Pass...

- Sequences of one-hot-encodings of terms.
- Sequences of pre-trained vector embeddings of terms.



# Embedding Layer



## With Hot Encodings, Model Will Still Struggle to Figure Out Semantics

- Despite having sequence, the model is “told” that the tokens are orthogonal / independent of one another in their meanings. But that’s not true!

## Textual Embedding Layer First Provides Dimensionality Reduction

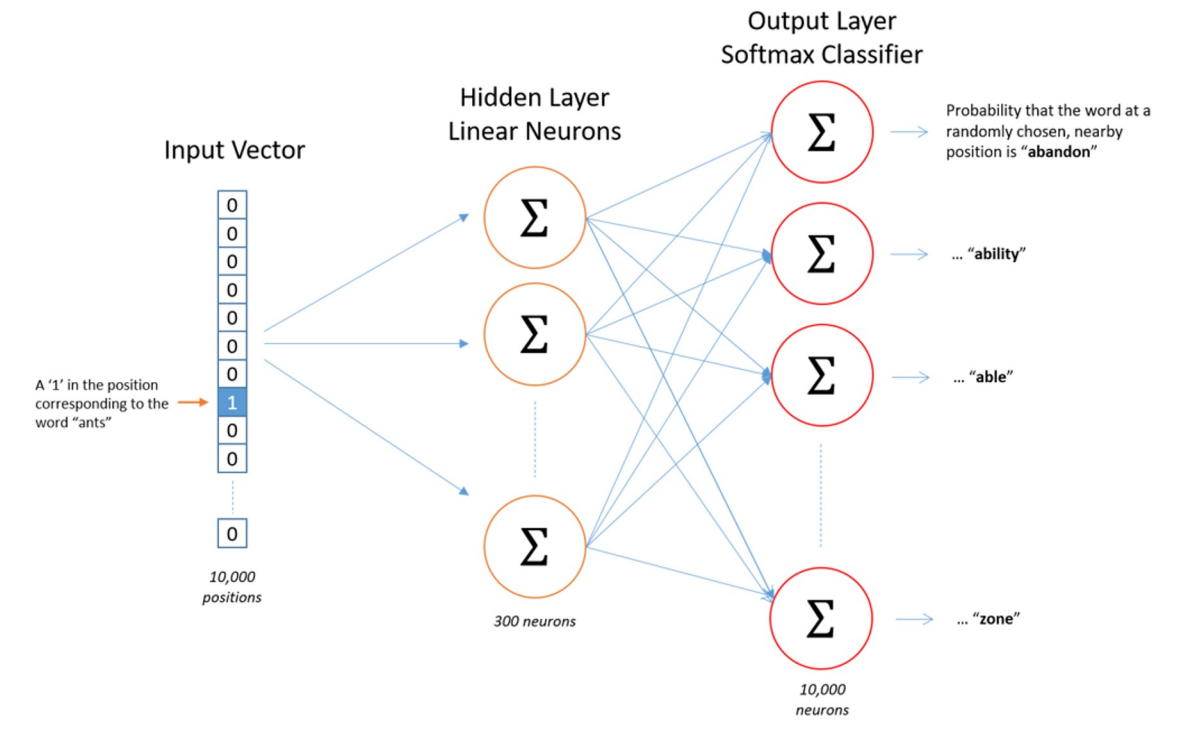
- Represent words into a lower dimensional space – similar vector = similar meaning.
- The Embedding layer is a lookup table that maps tokens to vectors. For each token in the vocabulary, the network learns a vector representation. The vectors are initially random, and the network updates them in training to learn representations that help in prediction (just like with convolution filters!).
- In practice, it is learning semantic relationships...
- This is much better for an RNN than a hot encoding, because 120 values (for example) is  $\ll$  20,000!

# Pre-Trained Embeddings: Word2Vec

## Word2Vec

- Two types: CBoW and Skipgram
- Construct training examples and labels.

Source Text	Training Samples generated from source text
I will have orange juice and eggs for breakfast	(will, I) (will, have) (will, orange)
I will have orange juice and eggs for breakfast	( have, I) (have, will) (have, orange) (have, juice)
I will have orange juice and eggs for breakfast	(orange, will) (orange, have) (orange, juice) (orange, and)
I will have orange juice and eggs for breakfast	(juice, have) (juice, orange) (juice, and) (juice, eggs)
I will have orange juice and eggs for breakfast	(and, orange) (and, juice) (and, eggs) (and, for)
I will have orange juice and eggs for breakfast	(eggs, juice) (eggs, and) (eggs, for) (eggs, breakfast)
I will have orange juice and eggs for breakfast	( for, and) ( for, eggs) ( for, breakfast)





# Pre-Trained Embeddings: Limitation

## Out of Sample Words

- Both GloVe and Word2Vec are limited to words you've seen before in training. They cannot handle new words. Those words thus get omitted / dropped, or you need to do something different.

## FastText

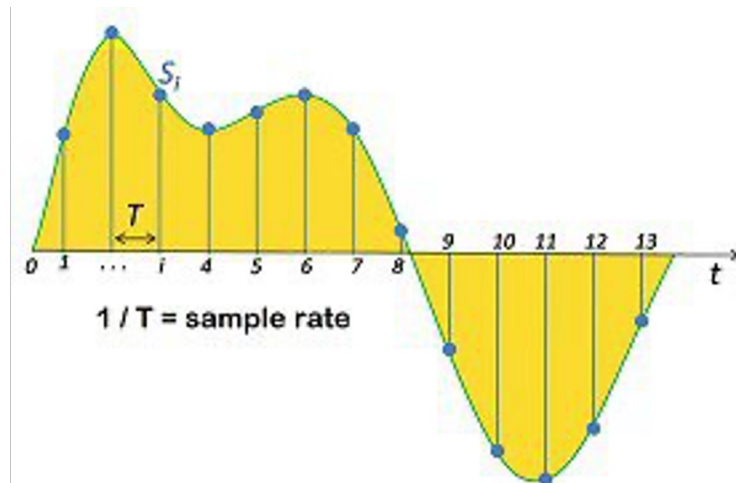
- An extension to Word2Vec which learns character n-grams of words. So, instead of embedding words, we embed portions of words (e.g., a 3-gram character representation would break up the word 'coffee' into 'cof', 'off', 'ffe', ... and then learn vector embeddings of each).



# RNN for Audio

## Same Sequence Concepts Work for Audio Data

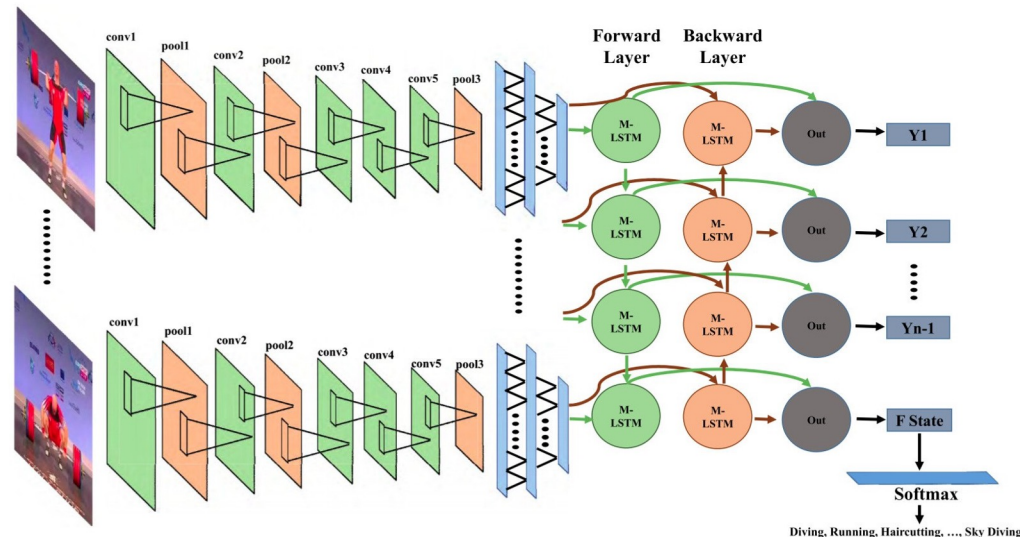
- Audio files are just sequences of numeric values (amplitude), possibly two if it was recorded in stereo.
- Once we recognize this, we realize we can predict things about audio sequences too!



# CNN-RNN for Video

## Hybrid Topology for Image Sequences

- We Use CNN's to detect features at a given input.
- We feed those feature maps into an RNN architecture, like LSTM.
- We can use this topology to predict things about videos.
- You might pre-process frames using a pre-trained CNN and pass feature maps as sequences to an RNN.



# Questions?