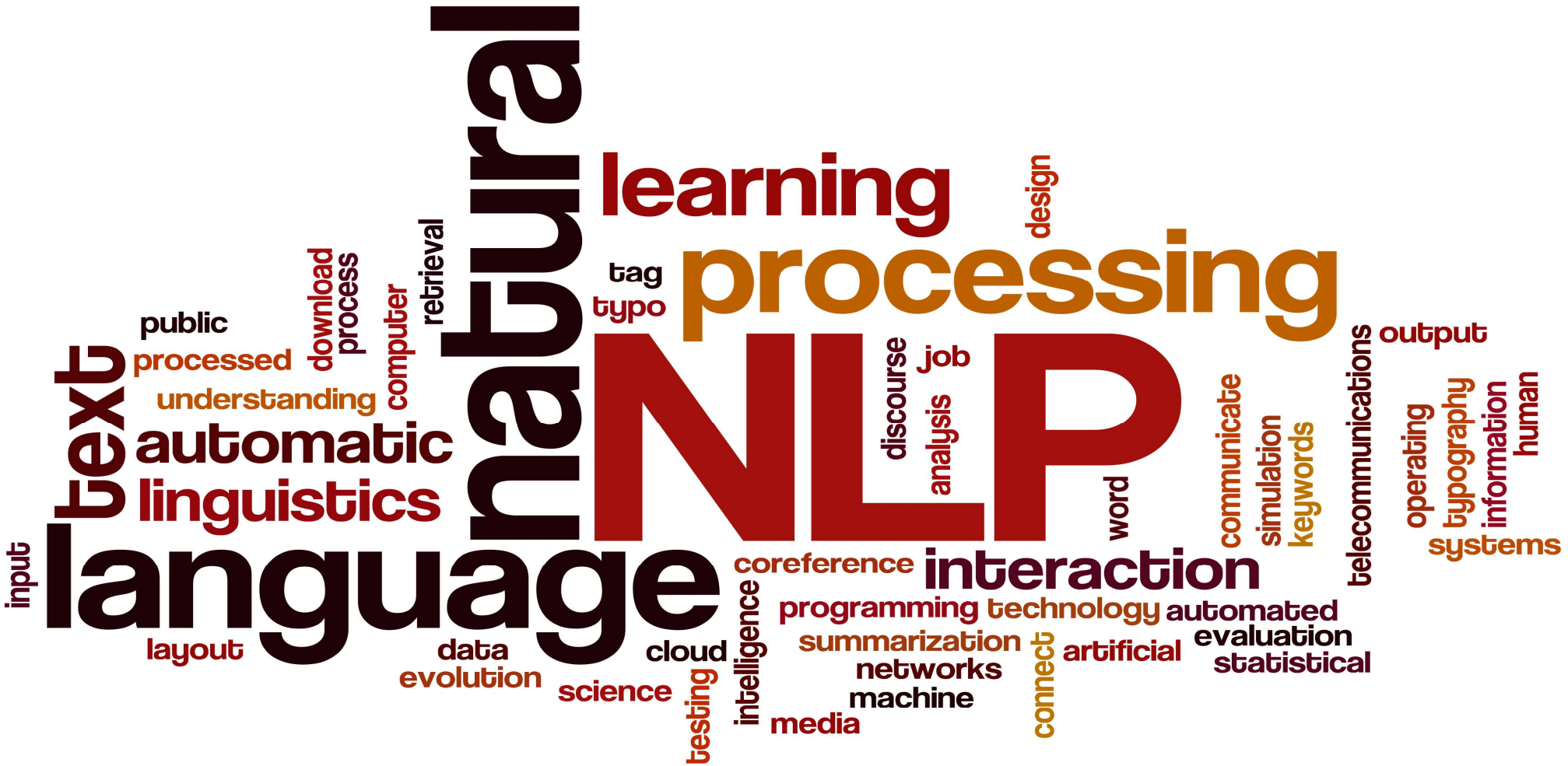
mindful-ai

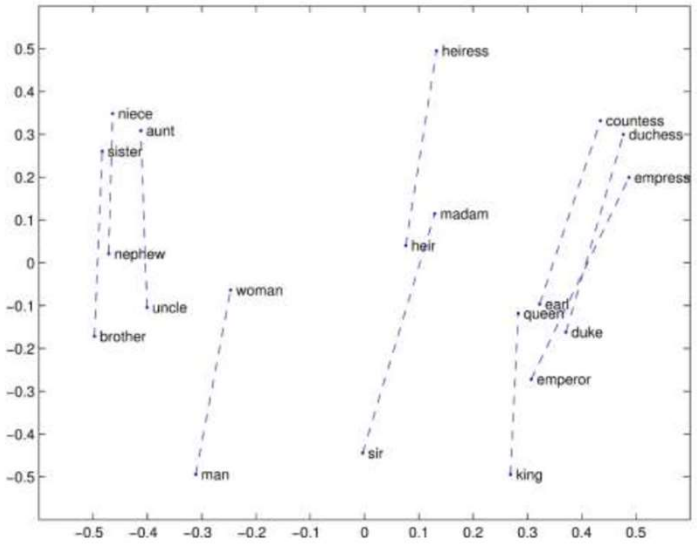

natural
learning
processing
NLP
language
text
automatic
linguistics
interaction
public
processed
understanding
download
process
computer
retrieval
tag
typo
design
job
discourse
analysis
word
communicate
simulation
keywords
telecommunications
output
operating
typography
information
human
systems
input
layout
data
evolution
science
testing
intelligence
cloud
coreference
programming
technology
automated
summarization
networks
machine
media
connect
artificial
evaluation
statistical

Natural Language Processing

- By “natural language” we mean a language that is used for everyday communication by humans.
- NLP is an Intersection of several fields
  - Computer Science
  - Artificial Intelligence
  - Linguistics
- It is basically teaching computers to process human language
- Two main components:
  - Natural Language Understanding (NLU)
  - Natural Language Generation (NLG)
- NLP is AI Complete
  - Requires all types of knowledge humans possess

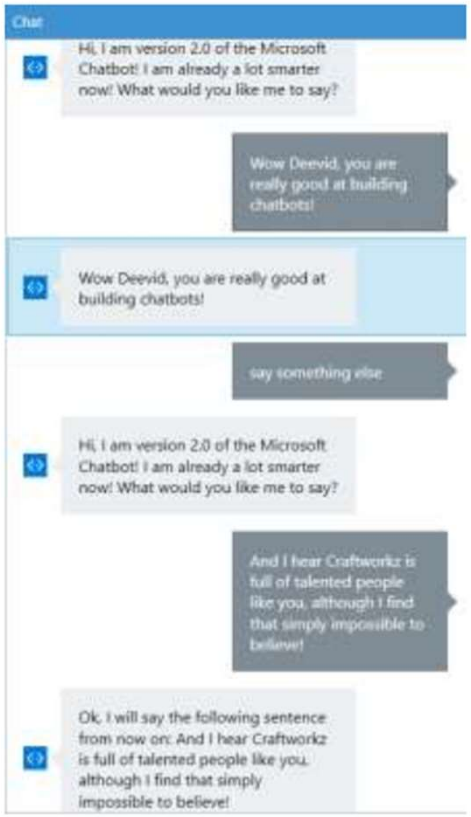
Natural Language Understanding

- Deriving meaning from natural language
- Imagine a Concept (aka Semantic or Representation) space
  - In it, any idea/word/concept has unique computer representation
  - Usually via a vector space
  - NLU > Mapping language into this space



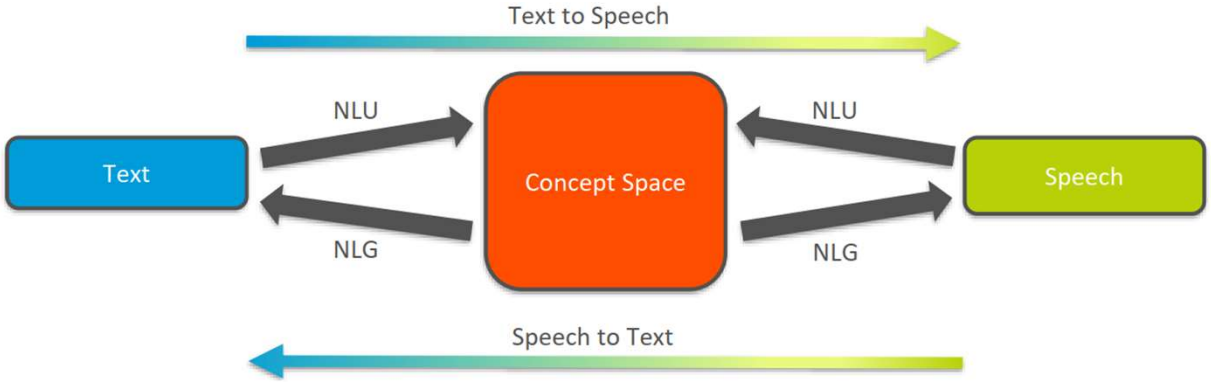
Natural Language Generation

- Mapping from computer representation space to language space
- Opposite direction of NLU
- Usually need NLU to perform NLG!



NLP : Speech Vs Text

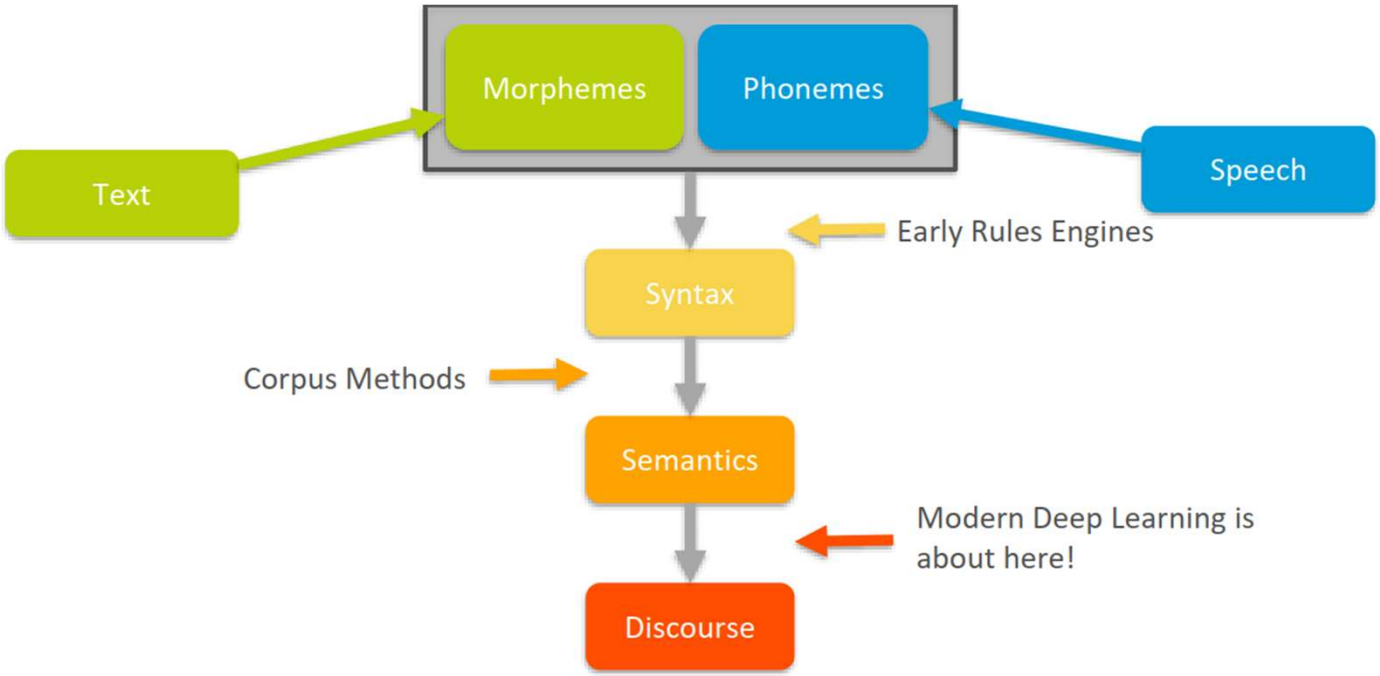
- Natural Language can refer to Text or Speech
- Goal of both is the same: translate raw data (text or speech) into underlying concepts (NLU) then possibly into the other form (NLG)



NLP Definitions

- Phonemes: the smallest sound units in a language
- Morphemes: the smallest units of meaning in a language
- Syntax: how words and sentences are constructed from these two building blocks
- Semantics: the meaning of those words and sentences
- Discourse: semantics in context. Conversation, persuasive writing, etc.

Level of NLP


Morphemes
Phonemes
Text
Speech
Early Rules Engines
Syntax
Corpus Methods
Semantics
Modern Deep Learning is about here!
Discourse

NLU Applications

- ML on Text (Classification, Regression, Clustering)
- Document Recommendation
- Language Identification
- Natural Language Search
- Sentiment Analysis
- Text Summarization
- Extracting Word/Document Meaning (vectors)
- Relationship Extraction
- Topic Modeling
- …and more!

NLU Application: Document Classification

- Classify “documents” - discrete collections of text - into categories
  - Example: classify emails as spam vs. not spam
  - Example: classify movie reviews as positive vs. negative
  - Example: classify legal documents as relevant vs. not relevant to a topic

NLU Application: Document Recommendation

- Choosing the most relevant document based on some information:
  - Example: show most relevant webpages based on query to search engine
  - Example: recommend news articles based on past articles liked
  - Example: recommend restaurants based on Yelp reviews

NLG Applications

- Image Captioning
- (Better) Text Summarization
- Machine Translation
- Question Answering/Chatbots
- …so much more
- Notice NLU is almost a prerequisite for NLG

NLG Application: Image Captioning

- Automatically generate captions from images
- **https://cs.stanford.edu/people/karpathy/cvpr2015.pdf**

NLG Application: Text Summarization

- Automatically generate text summaries of documents
- Example: Generate headlines of news articles
- **https://ai.googleblog.com/2016/08/text-summarization-withtensorflow.html**

Input: Article 1st sentence	Model-written headline
metro-goldwyn-mayer reported a third-quarter net loss of dlrs 16 million due mainly to the effect of accounting rules adopted this year	mgm reports 16 million net loss on higher revenue
starting from july 1, the island province of hainan in southern china will implement strict market access control on all incoming livestock and animal products to prevent the possible spread of epidemic diseases	hainan to curb spread of diseases
australian wine exports hit a record 52.1 million liters worth 260 million dollars (143 million us) in september, the government statistics office reported on monday	australian wine exports hit record high in september

Regex

- Regular expression describe a pattern to look for in a text
- Use cases:
  - Parsing documents with expected component structure
  - Validating User Input
  - NLP Preprocessing

NLP Toolkits

- NLTK (Natural Language Toolkit)
  - The most popular NLP library
- TextBlob
  - Wraps around NLTK and makes it easier to use
- spaCy
  - Built on Cython, so it’s fast and powerful
- gensim
  - Great for topic modeling and document similarity

NLTK Installation

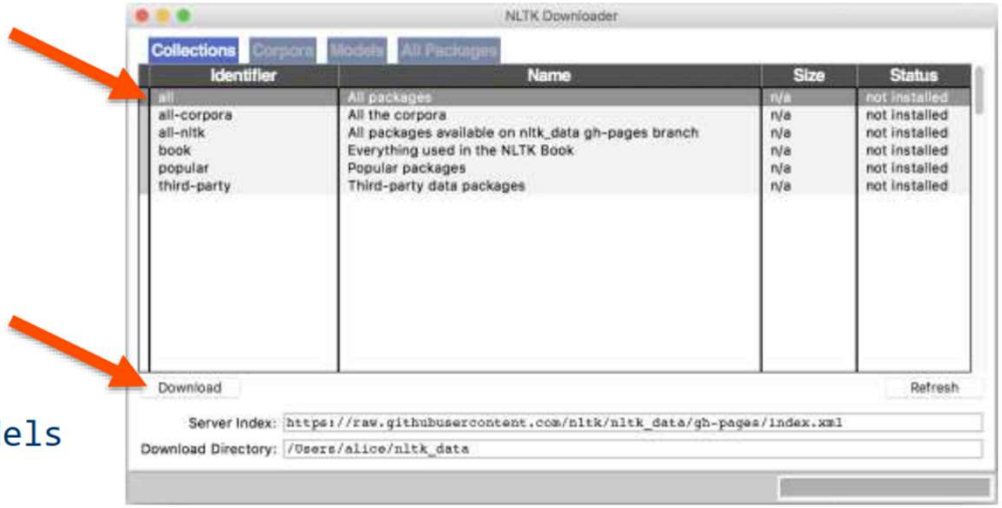
Command Line

```
pip install nltk
```

Jupyter Notebook

```
import nltk
nltk.download()

# downloads all data & models
# this will take a while
```



Tokenization

- Tokenization = splitting raw text into small, indivisible units for processing
- These units can be:
  - Words
  - Sentences
  - N-grams
  - Other characters defined by regular expressions

Removing Characters

- How can we normalize this text?
  - Remove punctuation
  - Remove capital letters and make all letters lowercase
  - Remove numbers

Stop Words

- Stop words are words that have very little semantic value.
- There are language and context-specific stop word lists online that you can use.

Stemming

- Stemming & Lemmatization = Cut word down to base form
  - Stemming: Uses rough heuristics to reduce words to base
  - Lemmatization: Uses vocabulary and morphological analysis
  - Makes the meaning of run, runs, running, ran all the same
  - Cuts down on complexity by reducing the number of unique words
- Multiple stemmers available in NLTK
  - PorterStemmer, LancasterStemmer, SnowballStemmer
  - WordNetLemmatizer

Parts of Speech Tagging

- Parts of Speech
  - Nouns, verbs, adjectives, etc.
  - Parts of speech tagging labels each word as a part of speech

Named Entity Recognition

- Named Entity Recognition (NER) aka Entity Extraction
  - Identifies and tags named entities in text (people, places, organizations, phone numbers, emails, etc.)
  - Can be tremendously valuable for further NLP tasks
  - For example: “United States” --> “United_States”

Compound Term Extraction

- Extracting and tagging compound words or phrases in text
  - This can be very valuable for special cases
  - For example: “black eyed peas“ --> “black_eyed_peas”
  - This totally changes the conceptual meaning!
  - Named entity recognition groups together words and identifies entities, but doesn’t capture them all, so you can identify your own compound words

Text Similarity Measures

- What are Text Similarity Measures?
  - Text Similarity Measures are metrics that measure the similarity or distance between two text strings.
  - They can be done on surface closeness (lexical similarity) of the text strings or meaning closeness (semantic similarity)
- We will be discussing lexical word similarities and lexical documents similarities.
- Measuring similarity between documents is fundamental to most forms of document analysis. Some of the applications that use document similarity measures include; information retrieval, text classification, document clustering, topic modeling, topic tracking, matrix decomposition

Text Similarity Measures

- Word Similarity
  - Levenshtein distance
- Document Similarity
  - Count vectorizer and the document-term matrix
  - Bag of words
  - Cosine similarity
  - Term frequency-inverse document frequency (TF-IDF)

Word Similarity

- Why is word similarity important? It can be used for the following:
  - Spell check
  - Speech recognition
  - Plagiarism detection
- What is a common way of quantifying word similarity?
  - Levenshtein distance
  - Also known as edit distance in computer science

Word Similarity

- How similar are these words?

MATH	MATH
MATH	BATH
MATH	BAT
MATH	SMASH

Levenshtein Distance

- Levenshtein distance: Minimum number of operations to get from one word to another. Levenshtein operations are:
  - Deletions: Delete a character
  - Insertions: Insert a character
  - Mutations: Change a character
- Example: kitten —> sitting
  - kitten —> sitten (1 letter change)
  - sitten —> sittin (1 letter change)
  - sittin —> sitting (1 letter insertion)

Levenshtein distance = 3

Word Similarity

- For the list of words we had earlier:

MATH	MATH	Levenshtein distance = 0
MATH	BATH	Levenshtein distance = 1
MATH	BAT	Levenshtein distance = 2
MATH	SMASH	Levenshtein distance = 2

Document Similarity

- When is document similarity used?
  - When sifting through a large number of documents and trying to find similar ones
  - When trying to group, or cluster, together similar documents
- To compare documents, the first step is to put them in a similar format so they can be compared
  - Tokenization
  - Count vectorizer and the document-term matrix

Document Similarity

There are a few ways that text data can be put into a standard format for analysis

"This is an example"

Split Text Into Words

['This','is','an','example']

Tokenization

Numerically Encode Words

This	[1,0,0,0]
is	[0,1,0,0]
an	[0,0,1,0]
example	[0,0,0,1]

One-Hot Encoding

Text Format for Analysis: Count Vectorizer

	and	document	first	fun	is	one	second	the	third	this
0	0	1	1	0	1	0	0	1	0	1
1	0	1	0	0	1	0	1	1	0	1
2	1	0	0	1	1	2	0	1	1	0

This is called a **Document-Term Matrix**

- Rows = documents
- Columns = terms

Text Format for Analysis : Bag of Words

Bag of Words Model

- Simplified representation of text, where each document is recognized as a bag of its words
- Grammar and word order are disregarded, but multiplicity is kept

	and	document	first	fun	is	one	second	the	third	this
0	0	1	1	0	1	0	0	1	0	1
1	0	1	0	0	1	0	1	1	0	1
2	1	0	0	1	1	2	0	1	1	0

Document Similarity: Cosine Similarity

Cosine Similarity is a way to quantify the similarity between documents

- Step 1: Put each document in vector format
- Step 2: Find the cosine of the angle between the documents

Cosine similarity measures the similarity between two non-zero vectors with the cosine of the angle between them.

"I love you"

"I love NLP"

	i	love	you	nlp
Doc 1	1	1	1	0
Doc 2	1	1	0	1

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2}$$

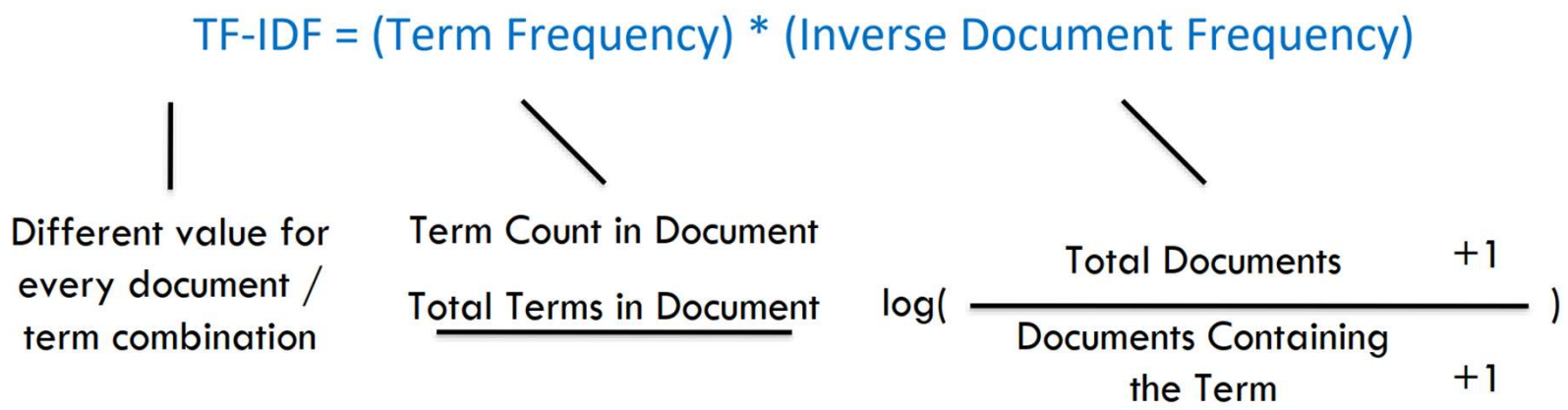
a = [1, 1, 1, 0]
b = [1, 1, 0, 1]

= 0.667

Beyond Count Vectorizer

- Downsides of Count Vectorizer
  - Counts can be too simplistic
  - High counts can dominate, especially for high frequency words or long documents
  - Each word is treated equally, when some terms might be more important than others
- We want a metric that accounts for these issues
  - Introducing Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF


TF-IDF = (Term Frequency) * (Inverse Document Frequency)
Different value for every document / term combination
Term Count in Document
Total Terms in Document
log(
Total Documents
+1
Documents Containing the Term
+1
)

TF-IDF

Term Frequency

- So far, we've been recording the term (word) count

"This is an example"

This	is	an	example
1	1	1	1

- However, if there were two documents, one very long and one very short, it wouldn't be fair to compare them by word count alone
- A better way to compare them is by a normalized term frequency, which is (term count) / (total terms).
- There are many ways to do this. Another example is log(count +1)

This	is	an	example
0.25	0.25	0.25	0.25

TF-IDF

Inverse Document Frequency

- Besides term frequency, another thing to consider is how common a word is among all the documents
- Rare words should get additional weight

The log dampens the effect of IDF

$$\log\left(\frac{\text{Total Documents} + 1}{\text{Documents Containing the Term} + 1}\right)$$

Want to make sure that the denominator is never 0

TF-IDF

TF-IDF Intuition:

- TF-IDF assigns more weight to rare words and less weight to commonly occurring words.
- Tells us how frequent a word is in a document relative to its frequency in the entire corpus.
- Tells us that two documents are similar when they have more rare words in common.

Building a ML Model for Text

Steps for classification with NLP

1. *Prepare the data*: Read in labelled data and preprocess the data
2. *Split the data*: Separate inputs and outputs into a training set and a test set, repsectively
3. *Numerically encode inputs*: Using Count Vectorizer or TF-IDF Vectorizer
4. *Fit a model*: Fit a model on the training data and apply the fitted model to the test set
5. *Evaluate the model*: Decide how good the model is by calculating various error metrics

Example

A classic use of text analytics is to flag messages as spam

Below is data from the SMS Spam Collection Data, which is a set of over 5K English text messages that have been labeled as spam or ham (legitimate)

Text Message	Label
Nah I don't think he goes to usf, he lives around here though	ham
Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's	spam
WINNER!! As a valued network customer you have been selected to receivea £900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.	spam
I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.	ham
I HAVE A DATE ON SUNDAY WITH WILL!!	ham
…	…

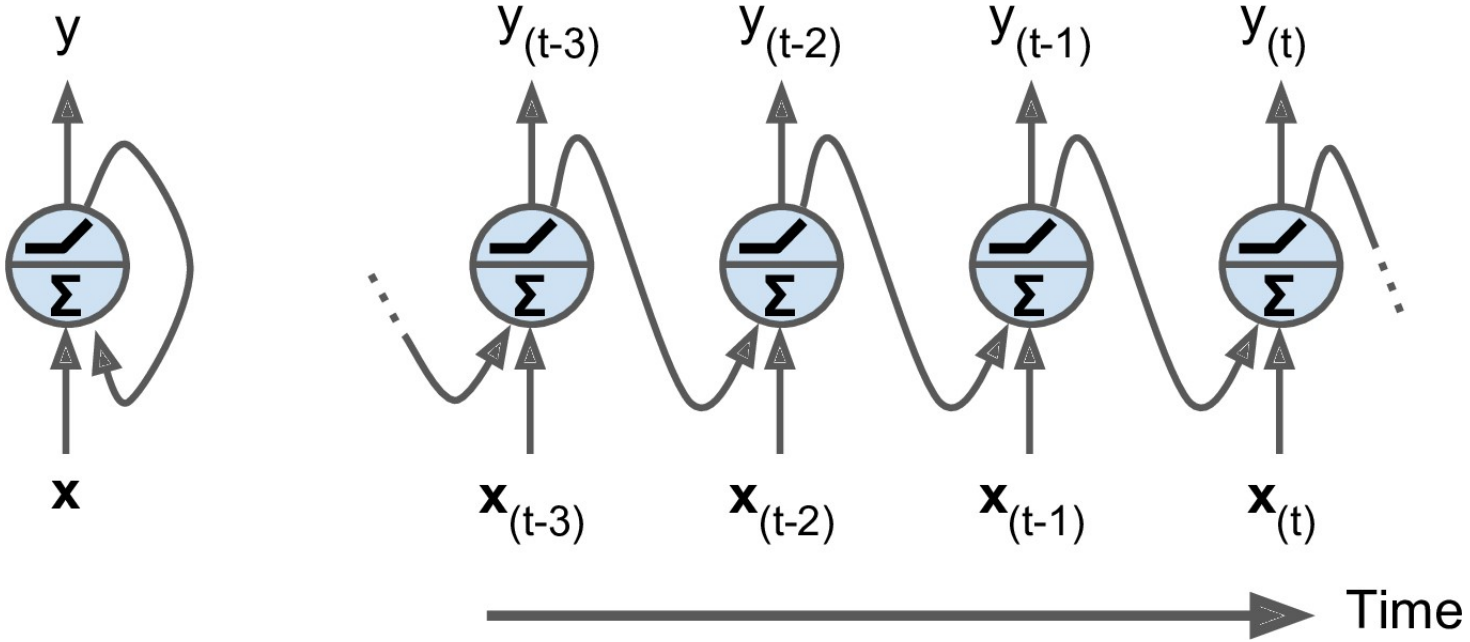
Recurrent Neural Networks

- Use RNN on Sequences of Data
  - Time Series Data
  - Sentences
  - Audio
  - Car Trajectories
  - Music
- Let's imagine a sequence [1,2,3,4,5,6]. Would you be able to predict a sequence shifted one time step into the future?
  - [2,3,4,5,6,7]

RNN

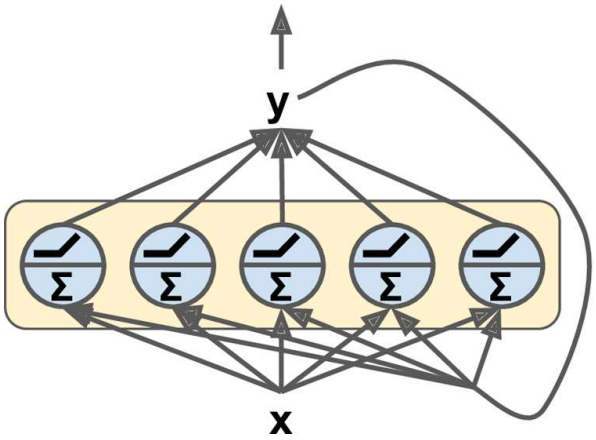
- To do this properly, we need to somehow let the neuron “know” about its previous history of outputs
- One east way to do this is to simply feed its output back into itself as an input

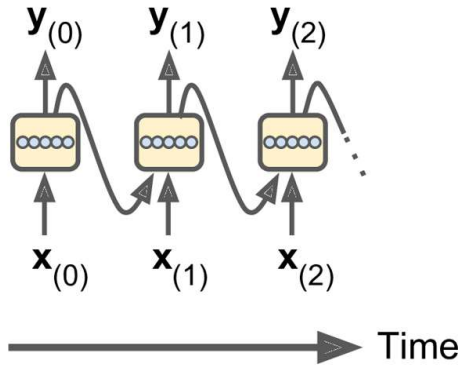
Recurrent Neuron



Memory Cells

- Cells that are a function of inputs from the previous time steps are also known as memory cells
- RNN are also flexible in their input and outputs for both sequences and single vector values
- We can create layers of recurrent neurons





Memory Cells

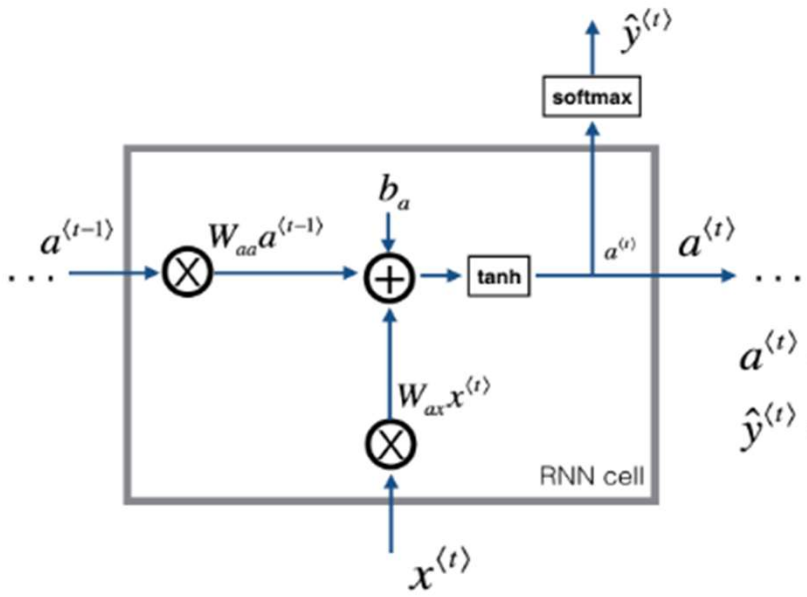
- A basic RNN has a major disadvantage. We only really “remember” the previous output.
- It would be great if we could keep track of longer history, not just short term history

LSTM

- An issue with RNN is that after a while the network will begin to “forget” the first inputs, as information is lost at each step going through the RNN
- We need some sort of “long-term-memory” for our networks
- The LSTM (Long Short Term Memory) cell was created to address this issue

LSTM

- RRN Cell for Single Time Step

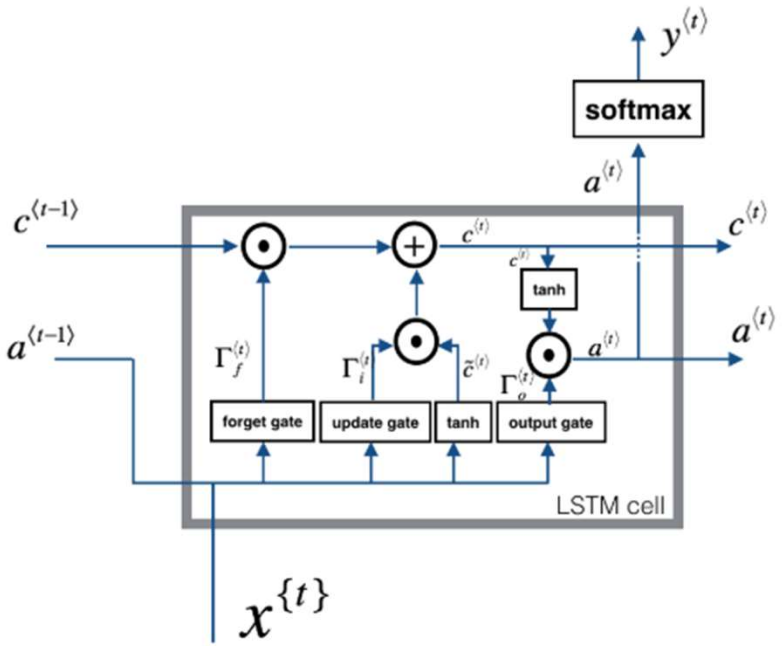


$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$$

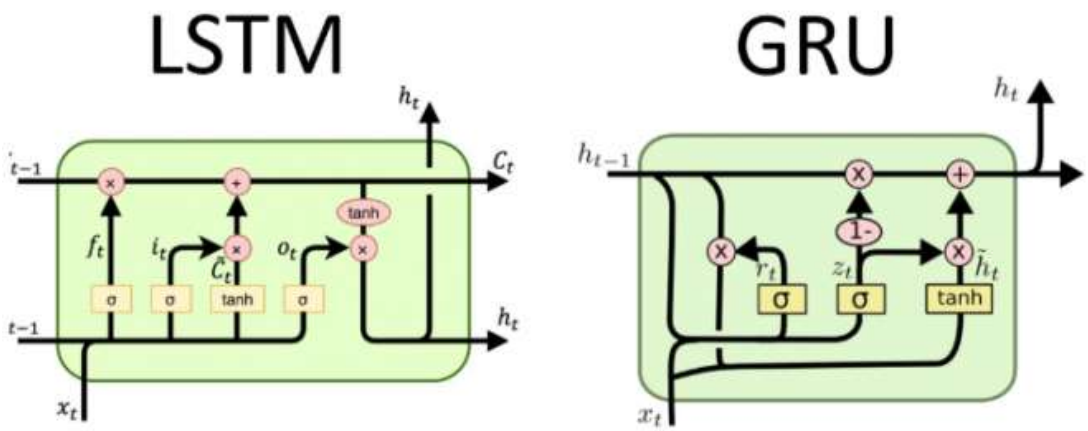
LSTM

- Long Short Term Memory Cell Architecture

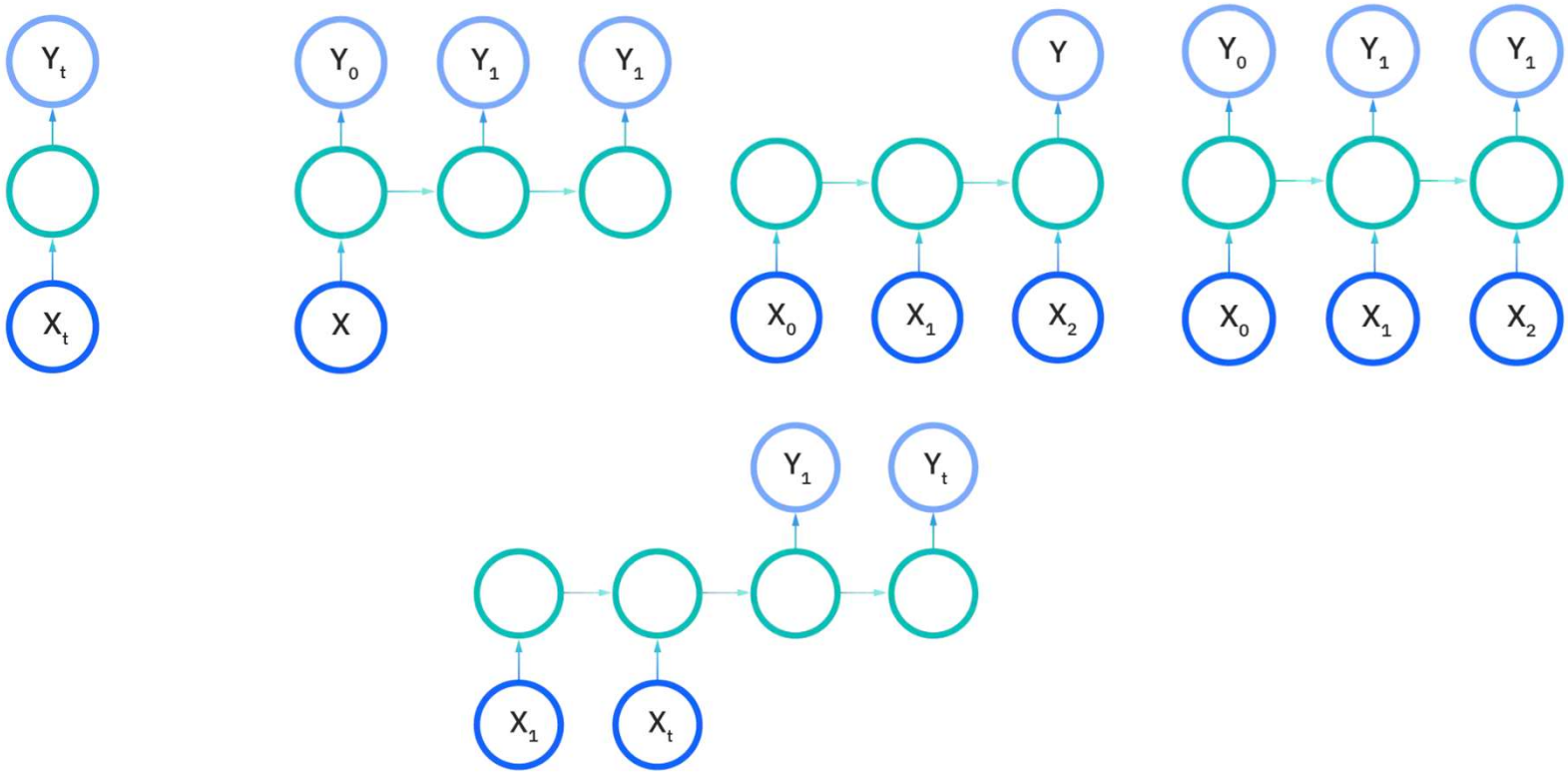


$$\Gamma_f^{\langle t \rangle} = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$$
$$\Gamma_u^{\langle t \rangle} = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$$
$$\tilde{c}^{\{t\}} = \tanh(W_C[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_C)$$
$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ c^{\langle t-1 \rangle} + \Gamma_u^{\langle t \rangle} \circ \tilde{c}^{\langle t \rangle}$$
$$\Gamma_o^{\langle t \rangle} = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o)$$
$$a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} \circ \tanh(c^{\langle t \rangle})$$

Gated Recurrent Unit



Types of RNN Connections



RNN

- Fortunately with out deep learning python library, we simple need to call the input for RNN or LSTM instead of needing to code all of this ourselves!!

RNN Batches

- Let's imagine a simple time series:
  - [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
- We separate this into 2 parts
  - [0, 1, 2, 3, 4, 5, 6, 7, 8]
  - [9]
- Given the training sequence, predict the next sequence value
- We can usually decide how long the training sequence and predicted label should be
  - [0, 1, 2, 3, 4, 5] [6, 7, 8, 9]

RNN Batches

- We can also edit the size of the training point, as well as how many sequences to feed per batch:
  - [1, 2, 3, 4] [5]
  - [2, 3, 4, 5] [6]
  - [3, 4, 5, 6] [7]
- Example: If we are training on seasonal data, include atleast 12 months in the training sequence
  - This often takes domain knowledge and experience as well as simply experimenting and using RMSE to measure error of forecasted predictions

Forecasting

- We can forecast predictions from the training data:
  - [6, 7, 8, 9] [10]
  - [7, 8, 9, 10] [11.2]
  - [8, 9, 10, 11.2] [12.4]
  - [9, 10, 11.2, 12.4] [14]
  - [10, 11.2, 12.4, 14] = Completed forecast