Text Mining 1

Brock Tibert
btibert@bu.edu

Outline

- Homework and Exam Discussion
- Text Mining - Working with Text as a Data Source
- Hands-on in python
- In-class Group Data Challenge - Classify SMS messages!

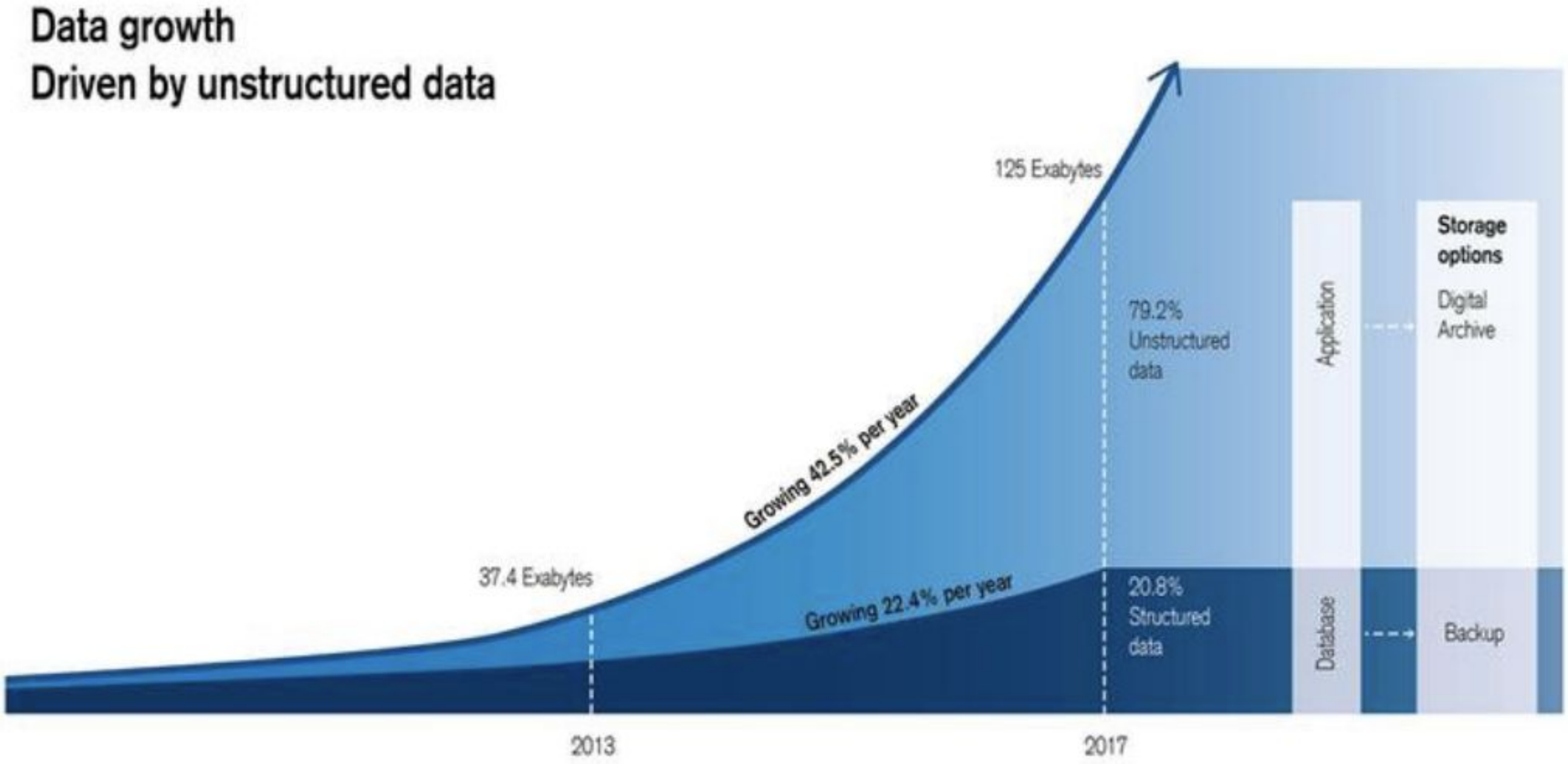
First Half Discussion

Text Analytics

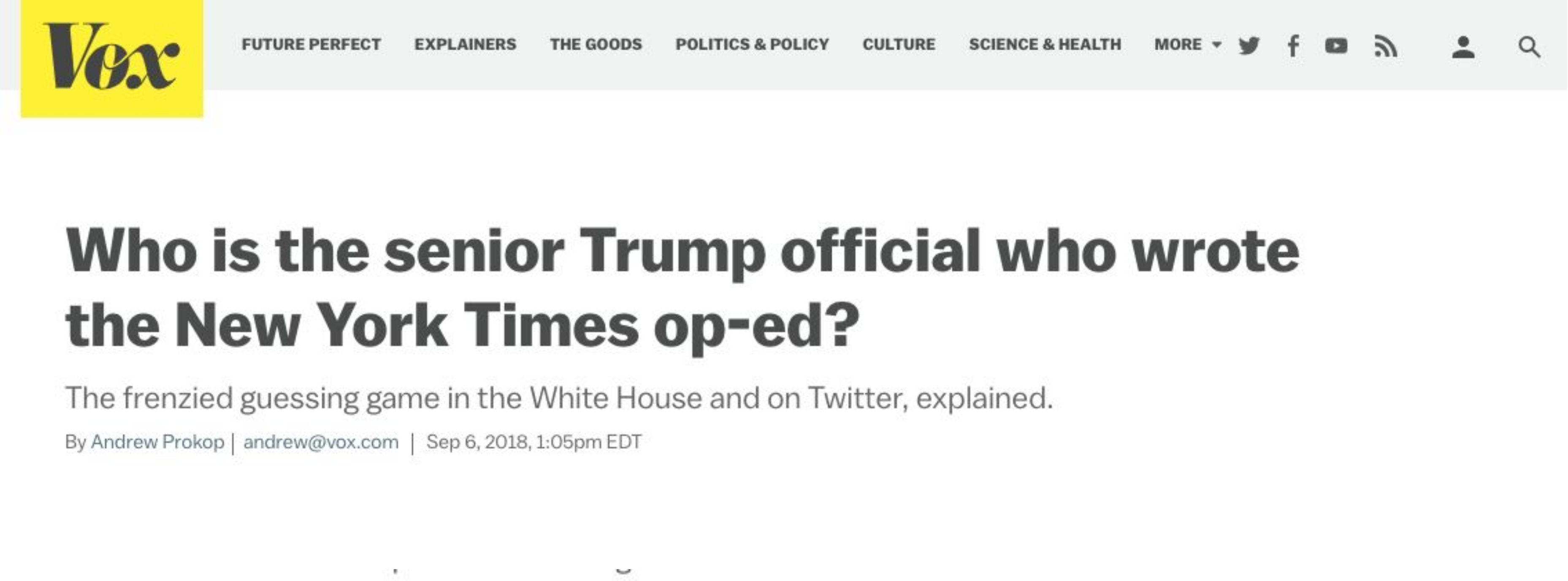
```
^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$
```

Why Text Mining?

- 80-90% of all corporate data is in some kind of unstructured form (e.g. text)
- Benefits of text mining are (or will be) obvious especially in text-rich data environments
    - Email and spam filtering
    - Law (Court-orders)
    - Financial Disclosures
    - Medicine (discharge summaries, doctor notes)
    - Marketing (Customer comments and reviews)
    - Customer Support (In-bound help, problems with ordering, FAQs, etc.)
    - Survey research and analysis of open-ended results
    - Content Management



125 Exabytes of enterprise data was stored in 2017; 80% was unstructured data. (Source: Credit Suisse)

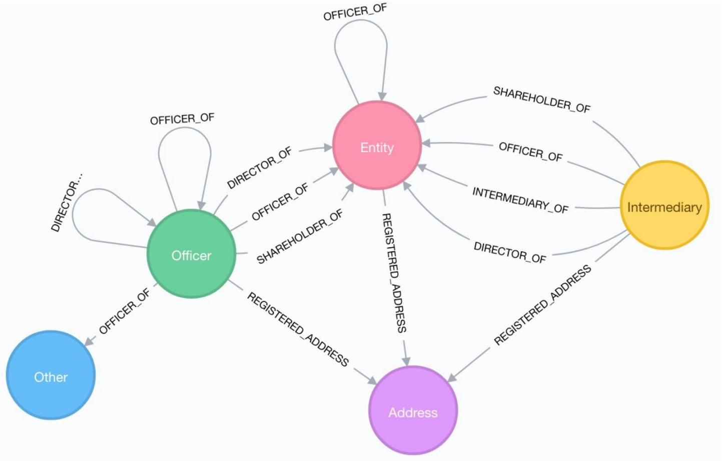


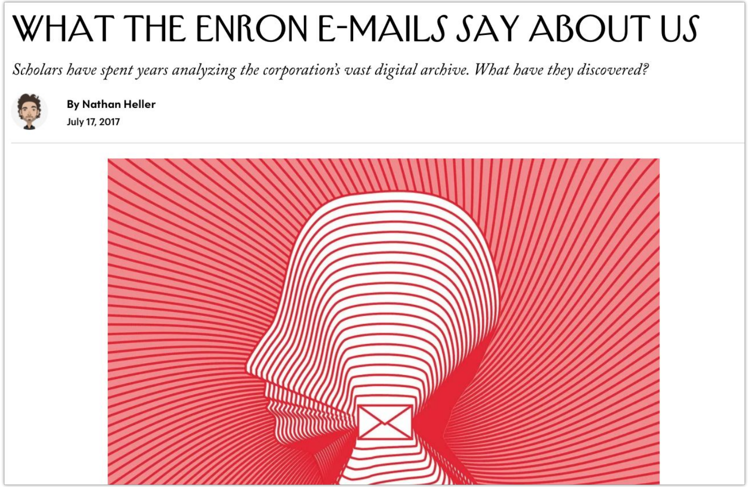
Who is the senior Trump official who wrote the New York Times op-ed?

The frenzied guessing game in the White House and on Twitter, explained.

By Andrew Prokop | andrew@vox.com | Sep 6, 2018, 1:05pm EDT

Are there linguistic clues to the author’s identity?

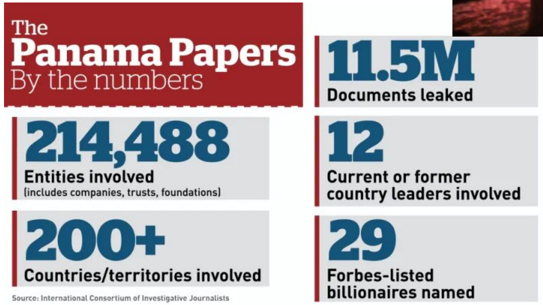




WHAT THE ENRON E-MAILS SAY ABOUT US

Scholars have spent years analyzing the corporation's vast digital archive. What have they discovered?

By Nathan Heller
July 17, 2017



The
Panama Papers
By the numbers

11.5M
Documents leaked

214,488
Entities involved
(includes companies, trusts, foundations)

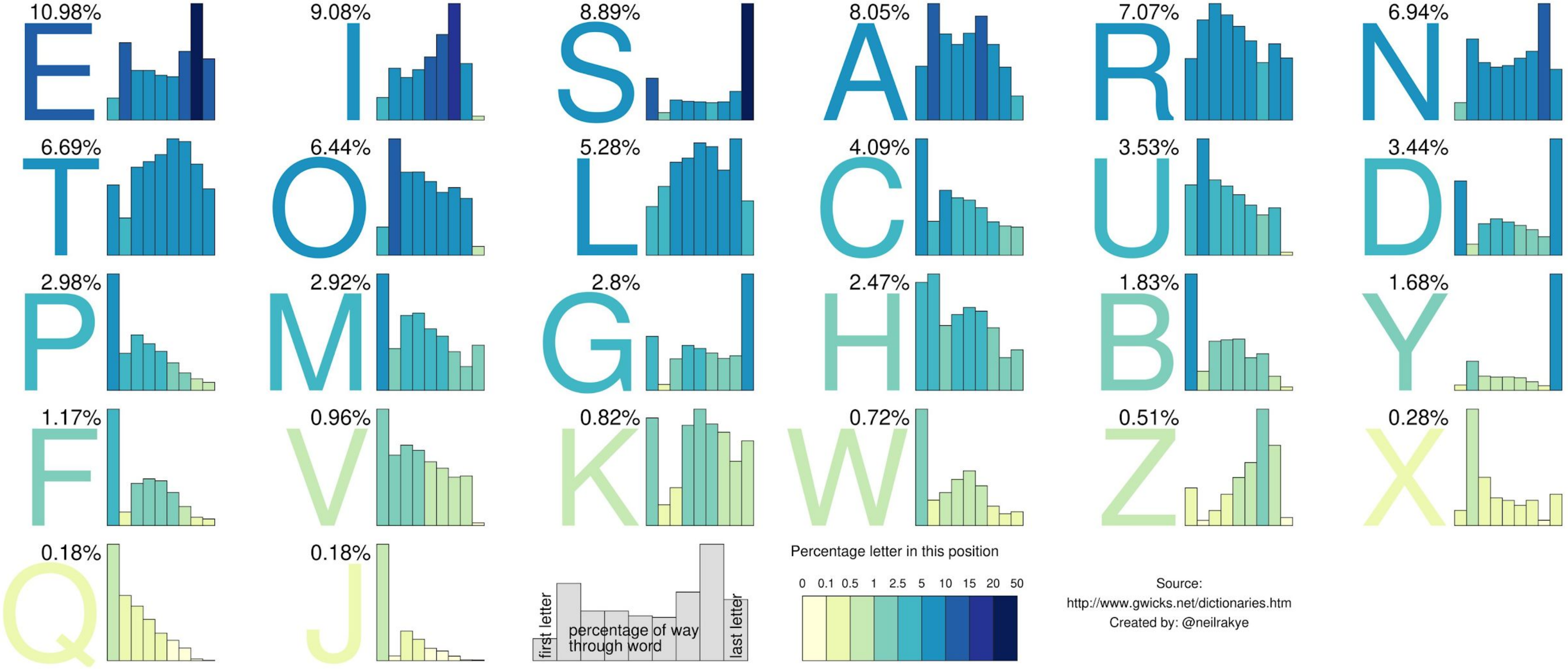
12
Current or former country leaders involved

200+
Countries/territories involved

29
Forbes-listed billionaires named

Source: International Consortium of Investigative Journalists

Frequency of letters in English words and where they occur in the word



Source:
http://www.gwicks.net/dictionaries.htm
Created by: @neilrakye

Objectives

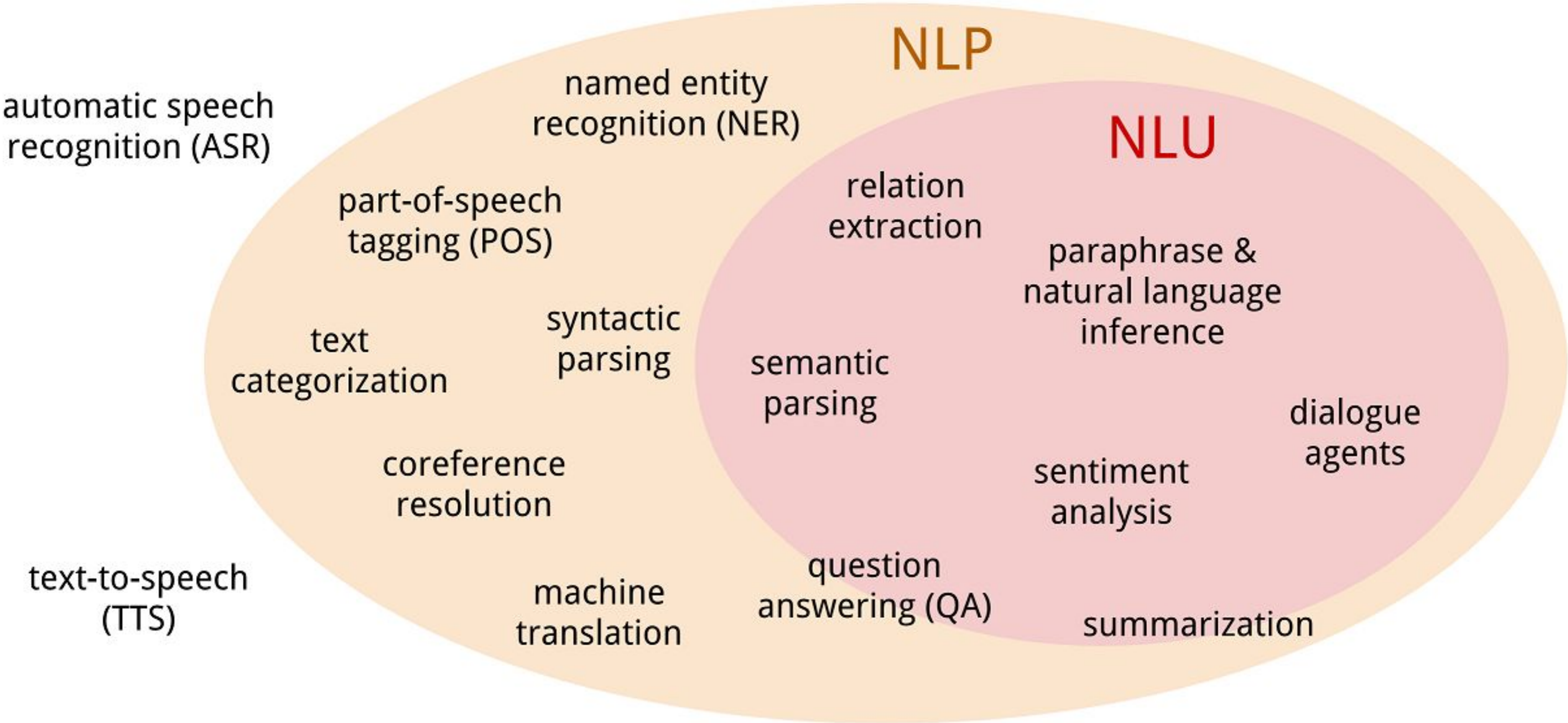
- Pattern Discovery (Unsupervised Learning)
- Prediction (Supervised Learning)

These are the **same goals** for **all of the tasks** we have covered in in the program. It’s just about putting the data together **properly**, for our **need**.

- Data acquisition, cleaning, reshaping, compression

In addition, text mining can be applied to look “inside” a document to:

- Identify targets of positive/negative sentiments
- Predict the intent
- Extract entities and relationships
- Identify the level of complexity in a document
  - E.g. Find the proper educational material for children



NLP

NLU

automatic speech recognition (ASR)

named entity recognition (NER)

part-of-speech tagging (POS)

text categorization

syntactic parsing

coreference resolution

machine translation

relation extraction

paraphrase & natural language inference

semantic parsing

dialogue agents

sentiment analysis

question answering (QA)

summarization

text-to-speech (TTS)

Text Mining Setup

- Data Mining techniques allow us to combine **natural language processing** with **ML methods** in order to identify patterns and make predictions

- All the classification and clustering techniques we have discussed are applicable, as soon as we transfer and parse the data into a familiar row-column format.
  - This is the format we will mostly focus on for our text work

There is a highly active community in python when it comes to building tools to work with text as a datasource. Unfortunately it's not as well as concentrated as there are "competing" projects that put their spin on core tasks and work.

variations of the same theme

Toolkits in python (an abbreviated list!)

nltk	sckit-learn	gensim	spacy	newspaper3k	TextBlob
coreNLP	pattern	PyNLPI	Monkeylearn	rasa	vaderSentiment
emoji	wordcloud	Vocabulary	Quepy	flair	AllenNLP
Transformers	fasttext	Tensorflow Text	pyLDAviz	OpenNLP	fastAI
graphBrain	polyglot	pyTextRank	stop-words	tokenwiser	textacy

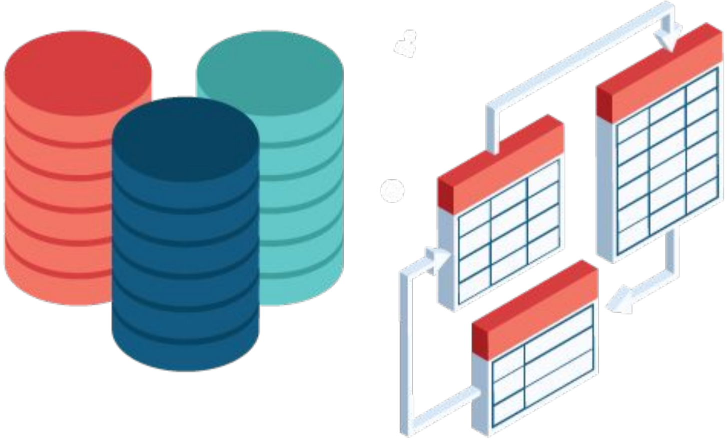
What makes this more *interesting* is that the packages depend on each other. Simply, some packages use other packages in the background.

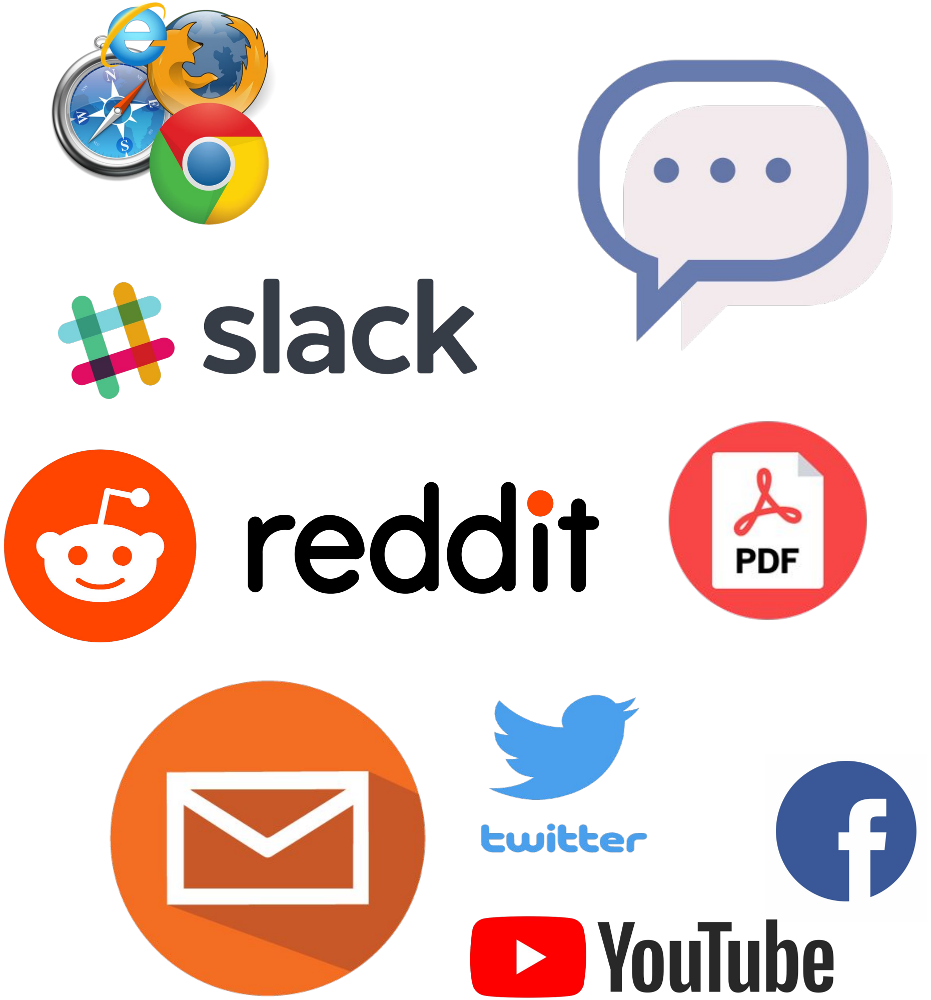
Toolkits in R (an abbreviated list!)

tm	tidytext	**quanteda**	udpipe	stm	topicmodels
textfeatures	Crfsuite	sentimentr	stopwords	tokenizers	cleanNLP
openNLP	textclean	textstem	textrecipes	wactor	text2vec
textmineR	ruimtehol	btm	qdap	rake	lexicon
readtext	spelling	hunspell	meanr	wordword	wordVectors

In case you are inteRested in R

Data Sources





(Higher Level) Definitions

Corpus: A collection of **documents** is a **corpus**

Document: An individual text composed of **tokens**. A document could be a tweet, a book, a news article, a blog post, a song’s lyrics, customer support request, a financial disclosure ...

Token: A token is a contiguous set of characters that does not contain a **separator**

- In other contexts, can be N-grams, or a sequence of tokens (golf club)

Separator: How do we define breaks between tokens? Whitespace? Punctuation? Every character?

Example - Document Term Matrix Construction

- The collection of sentences is the corpus
- Each sentence is the document
- Each word boundary is the token
- Each value is the simple term count, or occurrence
- Various python packages construct these, with slight differences

Sentence	hockey	ftw	i	like	golf
I like golf!			1	1	1
I like hockey.	1		1	1	
Hockey and golf ftw	1	1			1

Text Mining Process - Word inclusion and weighting

Create the Document-Term Matrix (also seen noted as: DTM, TDM, DFM)

Should all terms (N-grams) be included?

- Stopwords
- Stemming/Lemmatization

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

What is the best representation of values in the cells?

- Raw counts/frequencies? Binary values? Log of the counts?
- Inverse document frequency

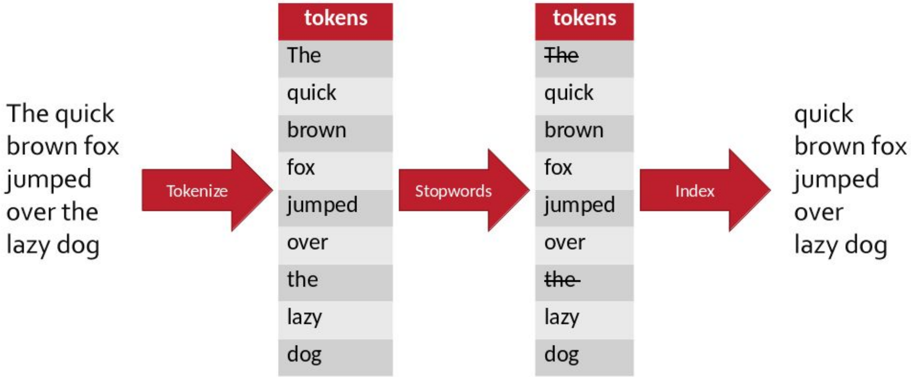
Compiling the Dictionary: Stop/Rare Words

Remove domain-specific Stopwords

Common words are typically removed as well, but it’s always good to review the stopwords for domain-specific projects

Also want to consider removing extremely rare and frequent words

- Either too rare it won’t add value
- Too common, it just adds noise



The quick brown fox jumped over the lazy dog

Tokenize

tokens
The
quick
brown
fox
jumped
over
the
lazy
dog

Stopwords

tokens
~~The~~
quick
brown
fox
jumped
over
~~the~~
lazy
dog

Index

quick brown fox jumped over lazy dog

The Vocabulary: Stemming & Lemmatization

In order to reduce the term space dimensionality, we want to root of the word

Stemming

Big, bigger, biggest = **big**

reach, Reached, Reaching = **reach**

work, works, worked, Working = **work**

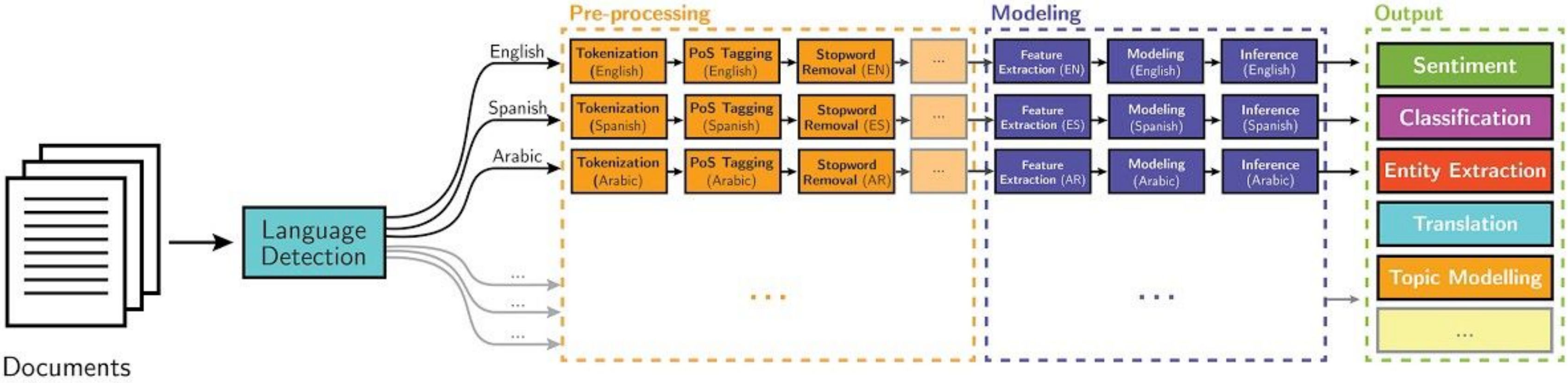
Lemmatization

drive, drove, driven = **drive**

Are, am, being = **be**

Of course there are different approaches to each
We usually perform one of these tasks, but try everything!
Decisions loosely depend on how compressed you want the feature space to be for the task ahead.

Once our data is processed, everything we have covered still applies



Text Analytics Mechanics

-*by hand to build intuition*-

python™

Team Challenge

Use Text Analytics to SMS Spam

Your analytics firm was hired to monitor spam messages that are now increasingly being sent as unsolicited SMS messages.

The datasets can be found on Big Query (questrom.SMSspam). The tables are train, and test. There is also an example submission file (that you will submit as a csv file).

You should consider combining the various techniques we have covered to date (data cleaning, clustering, dimensionality reduction,etc.) and use that work in concert with whatever classification method you feel is most appropriate.

Your submission to the leaderboard will be based on the accuracy.

HINT: Start simple and work towards complexity

Notes

- label is the label, and should be modeled as a classification problem
- Handle the data any way that you see fit
- Use test table as the data to apply your model and score the dataset with a label of ham/spam
- See the next slide for the format of your submission, which must be a csv
- Use any method you want to fit the classification model

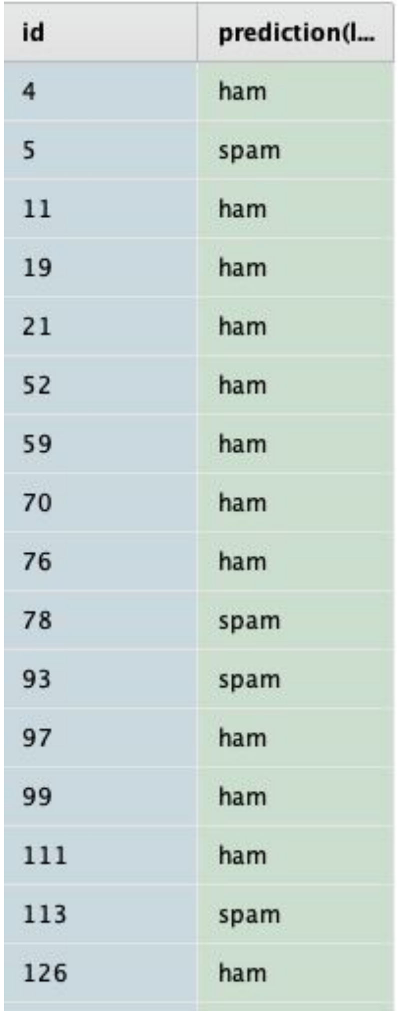
Tips and Tricks - What is our best, **naive** guess?

- Don’t be afraid to start simple and try different variations.
- Think about how to create columns/features from the dataset given the string of text
- Don’t try to build complex workflows right away, keep it simple for faster iteration and to see if you can improve your correlation score along the way
- What is our baseline assumption (naive guess)?

- Each team member can try a different approach to see who is getting better accuracy score

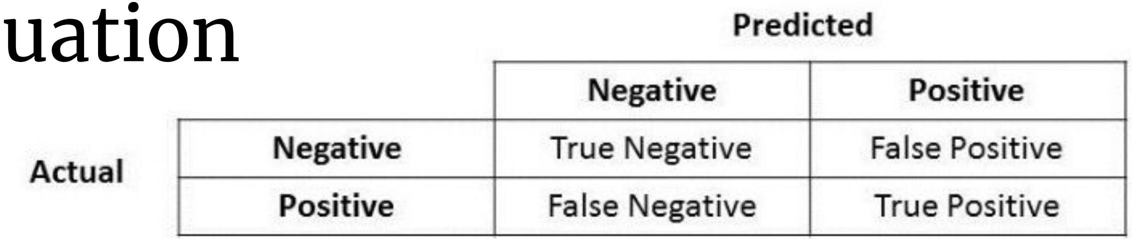
Submission CSV

- Filename does not matter
- Two column csv file with the id column and the predicted value as text
  - id
  - label
- You can submit as many times as you like
- Notice the prediction is ham/spam (string)
- NOTE: when writing your csv from pandas, remember, index=False
  - We only want those two columns



id	prediction(l...
4	ham
5	spam
11	ham
19	ham
21	ham
52	ham
59	ham
70	ham
76	ham
78	spam
93	spam
97	ham
99	ham
111	ham
113	spam
126	ham

Classification Evaluation

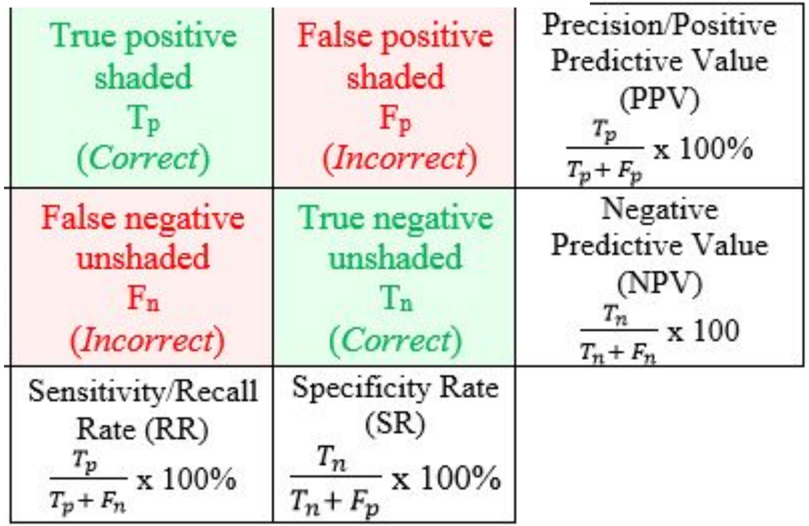


		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

- Accuracy
  - What percentage of the predictions were correct?
- Precision
  - How often were the model’s predictions accurate? TP / TP + FP
- Recall
  - What percentage of known positive cases were correctly identified? TP / TP + FN
- F1
  - Balance of Precision and Recall
  - Helpful when there is class imbalance

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Confusion Matrix and the Core Calculations



	Actual True/Yes	Actual False/No	
Predicted True/Yes	True positive shaded T_p *(Correct)*	False positive shaded F_p *(Incorrect)*	Precision/Positive Predictive Value (PPV) $\frac{T_p}{T_p + F_p}$ x 100%
Predicted False/No	False negative unshaded F_n *(Incorrect)*	True negative unshaded T_n *(Correct)*	Negative Predictive Value (NPV) $\frac{T_n}{T_n + F_n}$ x 100
	Sensitivity/Recall Rate (RR) $\frac{T_p}{T_p + F_n}$ x 100%	Specificity Rate (SR) $\frac{T_n}{T_n + F_p}$ x 100%	

- Depending on the source, actual or predicted could be rows or columns **so be careful**
- Green diagonal is the total correct cases. Accuracy rate is the green diagonal divided by total number of cases
- Red Diagonal is the total incorrect. Misclassification rate is the red diagonal divided by the total number of cases