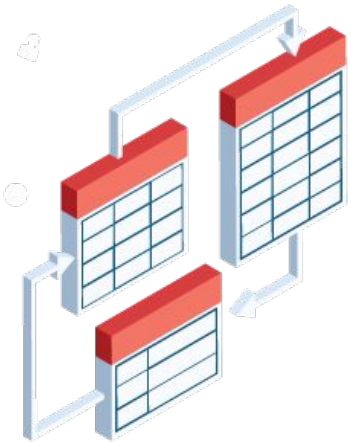
Text Mining 2

Brock Tibert
btibert@bu.edu

Boston University Questrom School of Business

Data Sources



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)

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

CountVectorizer, and because its a sparse output, .toarray()

Team Challenge 1

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

Use Text Analytics to SMS Spam

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

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

Tokenization

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
- Synonyms/normalization
- homonyms
- 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

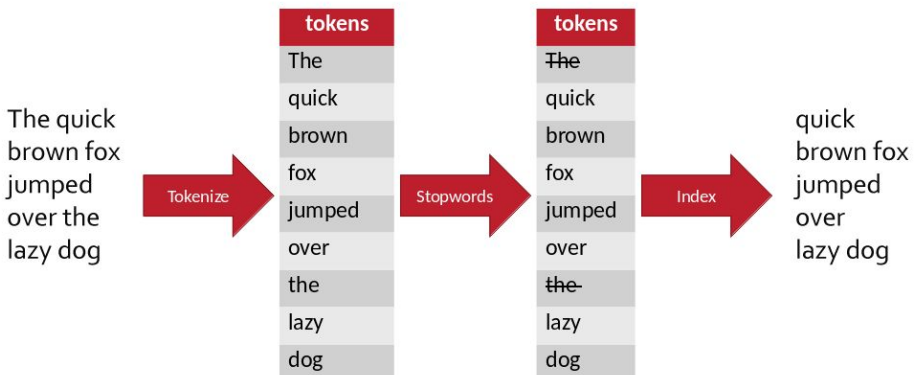
Compiling the Dictionary: Stop/Rare Words

Remove domain-specific Stopwords

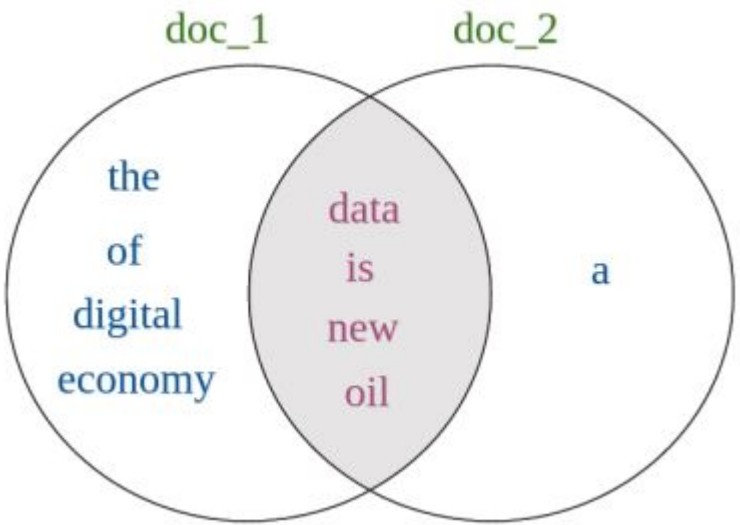
Common words are typically removed as well, but it's always good to review the stopwords for domain-specific projects

Also want to consider removing extremely rare and frequent words

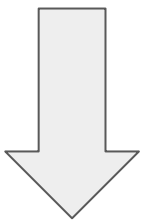
- Either too rare it won't add value
- Too common, it just adds noise

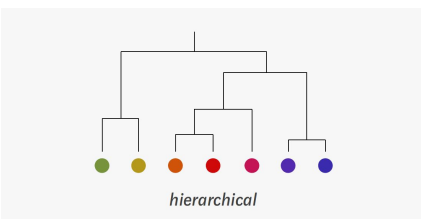


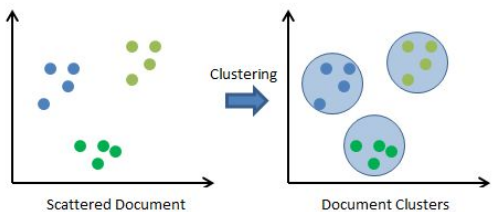
All the (S|U)ML tasks still apply!



	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

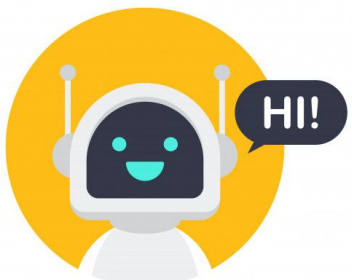






Let’s think about the possibilities!

- You **can** annotate a dataset for your specific problem
  - Customer support requests
  - HR requests (time off, support)
  - FAQ
  - Label Studio!
- Put your dataset into a dtm/embedding space
  - Remember, the words/tokens and their weights now represent that document in a feature space!
- When a new document comes in
  - Find “N” most similar documents
  - Suggest answers, label, or use it to predict a label (i.e. intent)
  - Duplicate detection
- You could easily serve this with an API via fastAPI!



Text Analytics Mechanics

Tokenization, Bag of Words and Clustering