

TEXT ANALYTICS

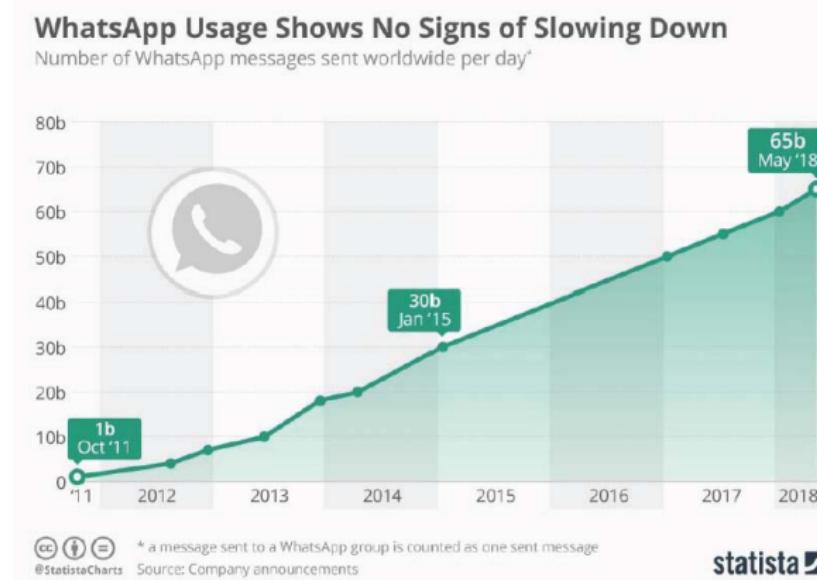
- Imdadullah Khan

Text Analytics

- Applying data analytics to deriving information from text
- Abundance of textual data is available in the form of:
 - Social media posts
 - Tweets
 - Question answer forums
 - Blogs
 - YouTube video comments
 - SMS
 - Product reviews
 - News articles

How much textual data is produced?

- 2.5 quintillion bytes of data created each day (Forbes)
- More than 65 billion messages sent on WhatsApp every day (Statista)
- 500 million tweets per day

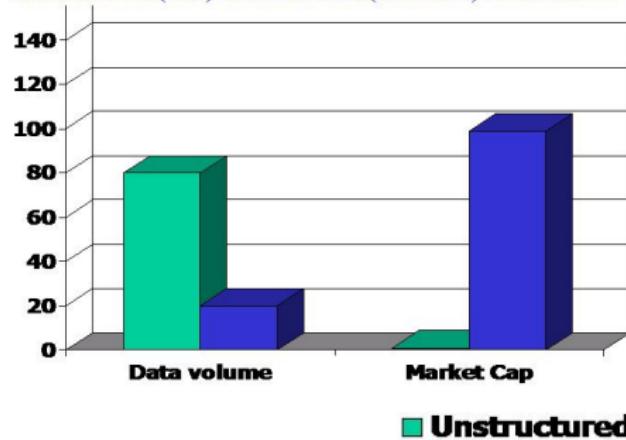


Stakeholders of text analytics

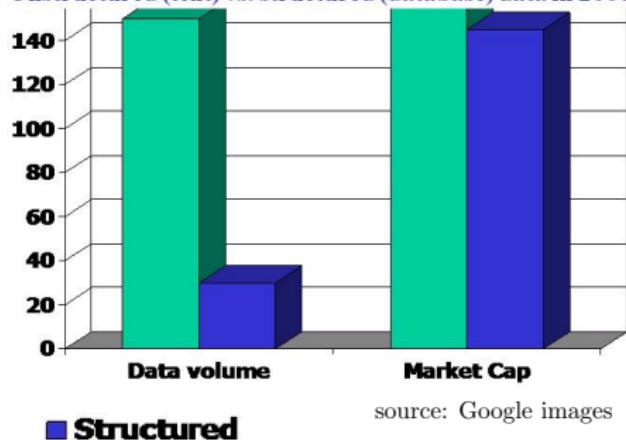
- **Government**
 - What is the response of people towards a particular policy?
- **Advertisers**
 - What is trending that could be used for advertisement?
 - Careem used **LUMSU** as promo code
- **Movie Makers**
 - What people disliked about a movie?
 - This information is used in future to deliver what people want
- **Brand Managers**
 - What value added services people want in a brand?
 - How people respond towards social responsibility campaigns that a brand launches?
- **Academia**
 - Is this document plagiarized?
 - Retrieve similar documents

Structured Vs Unstructured Data

Unstructured (text) vs. structured (database) data in 1996



Unstructured (text) vs. structured (database) data in 2006



source: Google images

- Unstructured (text) vs. structured (database) data in 1996 (left) and 2006 (right)
- Market cap of unstructured data has grown massively
- Need to come up with better techniques to handle the queries/search over both structured and unstructured data

Text Analytics: Tasks

Document Classification

Classify a collection of texts into a fixed set of categories

(Application of classification after text analytics)



Documents	Terms		C		
	T ₁	T ₂	T _N	
D ₁	12	0	6	Sports
D ₂	3	10	28	Travel
⋮	⋮	⋮	⋮	⋮	⋮
D _M	0	11	16	Jobs

Text Analytics: Tasks

Sentiment Analysis and Emotion Mining

Given a text, determine if the sentiment in the text is **positive** or **negative**
(Emotion Mining is fine-grained Sentiment Analysis)

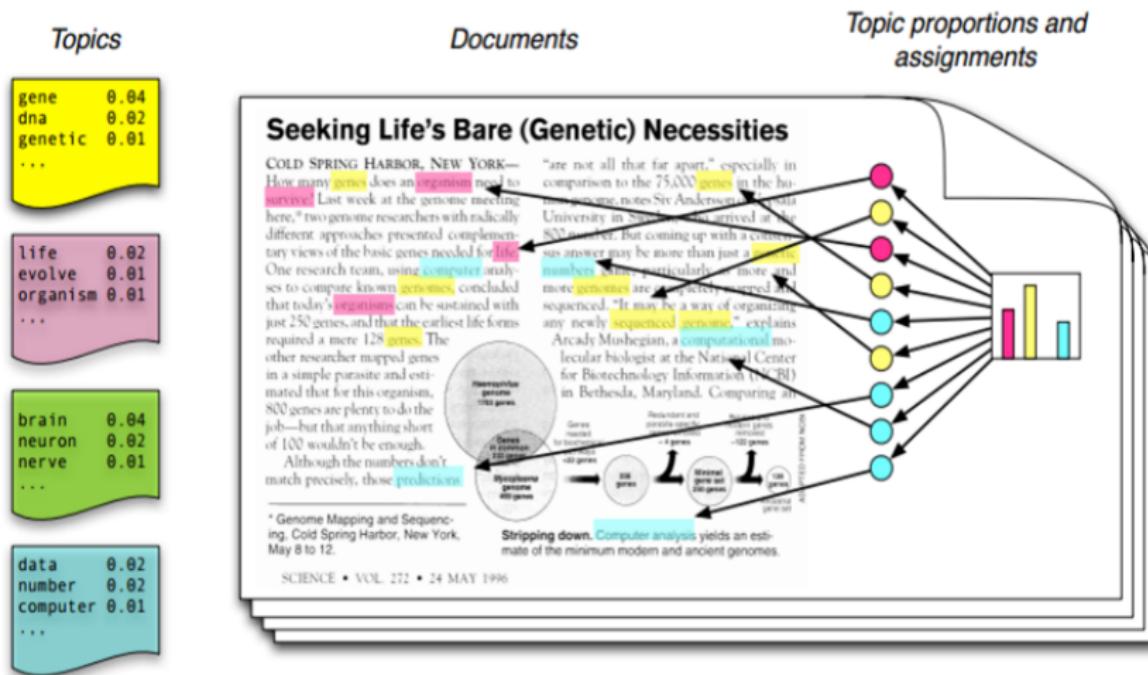
- Given reviews, determine how the product is perceived by public
- The Obama administration used it to gauge public opinion on policies and campaign messages ahead of 2012 election
- Customer services use it to improve products or services
- Given news headlines for last n days, would the stock market go up?



Text Analytics: Tasks

Topic Modeling: Determine the topics and subject of documents

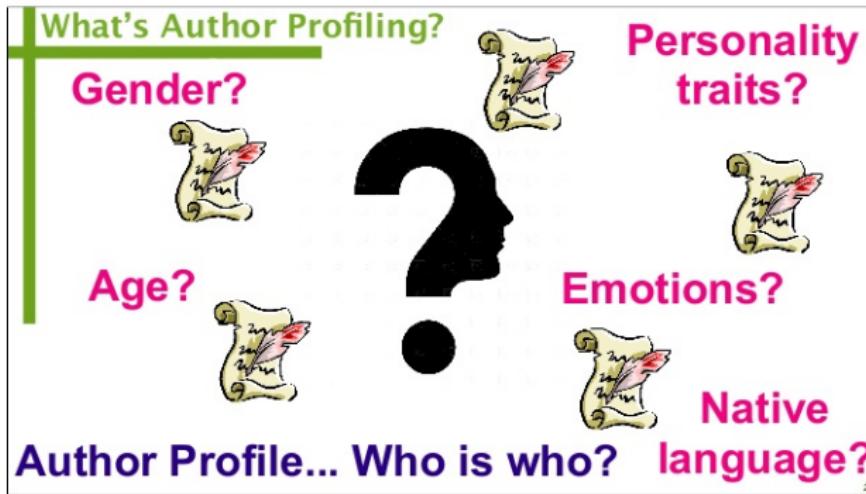
- Document clustering, information retrieval, reviewer assignment



Text Analytics: Tasks

Author profiling: Determine author attributes (age, gender, name etc.)

- **Security:** Who is behind anonymous threat message?
- **Sales and marketing:** Determine the demographic of the people behind online reviews who liked or disliked the products

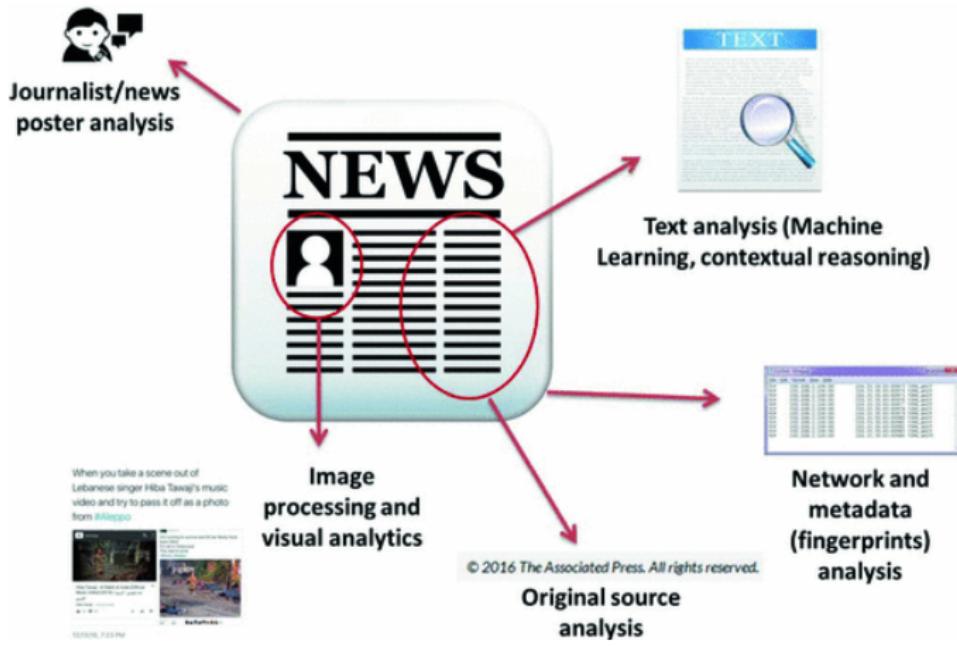


Text Analytics: Tasks

Fake News Identification:

Determine if a news item is fake or real

- Filtering and blocking of misleading information
- Identify trustworthy news sources

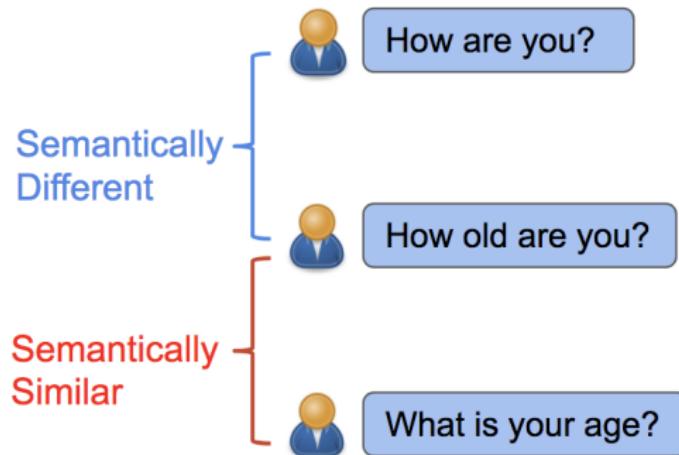


Text Analytics: Tasks

Paraphrase Identification: Find paraphrases or duplicates of documents

- Used for document clustering, information retrieval, plagiarism
- Useful for question-answer forums, where an answer could be retrieved if a question has already been asked and answered

source: Google AI blog



Text Analytics: Basic Concepts

Vocabulary (language lexicon): Unique words that may appear in texts

***n*-gram:** a (sub)sequence of *n* contiguous words in texts (aka **shingle**)

Texts considered as sequences of *n*-grams, large *n* captures more context

source: devopedia.org

This is Big Data AI Book

<i>Uni-Gram</i>	This	Is	Big	Data	AI	Book
<i>Bi-Gram</i>	This is	Is Big	Big Data	Data AI	AI Book	
<i>Tri-Gram</i>	This is Big	Is Big Data	Big Data AI	Data AI Book		

- In computational biology, they are called ***k*-mers**

Tokenization: Break a character sequence into predefined units

- Can be character level or word level, *n*-gram tokens

Text Analytics: Text Normalization

Text Normalization

- Initial Pre-processing of text dataset
- The goal is to standardize sentence structure and vocabulary
- Helps reduce number of variables (dimensionality)

Exact set of preprocessing steps depends on application, they include

- Remove duplicate whitespaces, punctuations, accents, capital letters, special characters
- Substitute word numerals by numbers (thirty → 30), values by type (\$100 → currency/money), contractions by phrases (I've → I have)
- Standardize formats (e.g. dates), replace abbreviation (e.g. USA)
- Stopwords removal
- Stemming
- Lemmatization

Text Analytics: Basic Concepts

Stop words

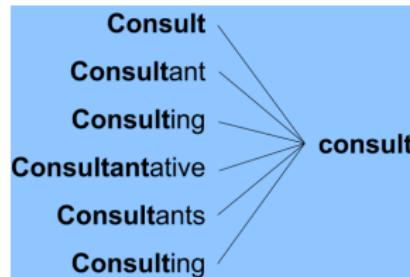
- The most common words that do not provide useful information. For instance: the, it, is, are, an, a etc.
- Often removed (filtered out) during pre-processing
- No universally good list of stop words
- Reduces time/space complexity, can improve analytics quality

Sr.No	Tokenized sentences	SWR sentences
1	I have been on this medicine for 2 years	Medicine 2 years.
2	It has no side effects except for gaining of weight.	Side effects except gaining weight.
3	It also helps me sleep at night.	Also helps sleep night.
4	I was extremely suicidal and depressed	Extremely suicidal depressed.
5	completely did the opposite effect of what it is meant for	Completely opposite effect meant.

Text Analytics: Basic Concepts

Stemming and Lemmatization

- In texts different grammatical variations of a words are used
- We convert different variations of a word to a common root form



- **Stemming:** crude heuristic way of chopping off ends of words
- **Lemmatization:** grammatically sound words replacing
 - am, are, is ⇒ be
 - car, cars, car's, cars' ⇒ car
 - “the boy's cars are different colors” ⇒ “ the boy car be differ color”

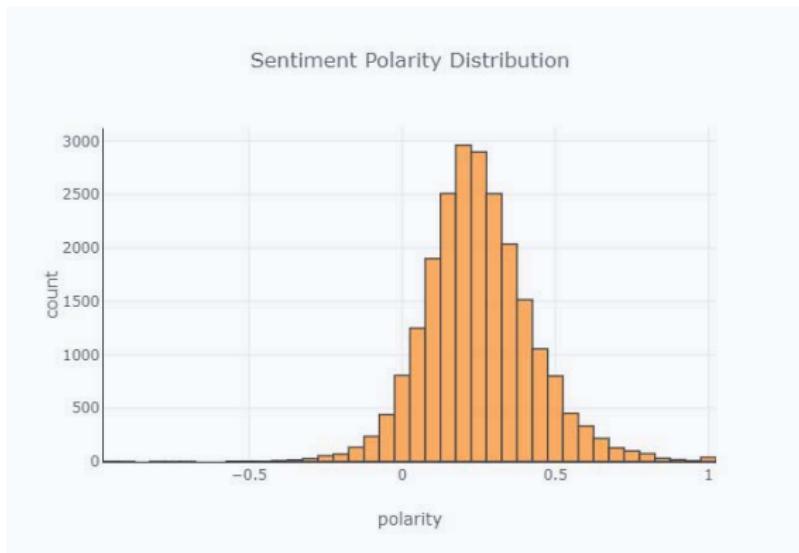
Text Analytics: Where to start?

- First step in text analytics is Exploratory Data Analysis (**EDA**)
- Gives insight about the data such as:
 - Class distribution
 - Top occurring words in the dataset
 - Distribution of words per document
- These insights help in formulating strategies to solve the task at hand
 - What preprocessing should be used?
 - What classifier should be used?

Text Exploratory Data Analysis

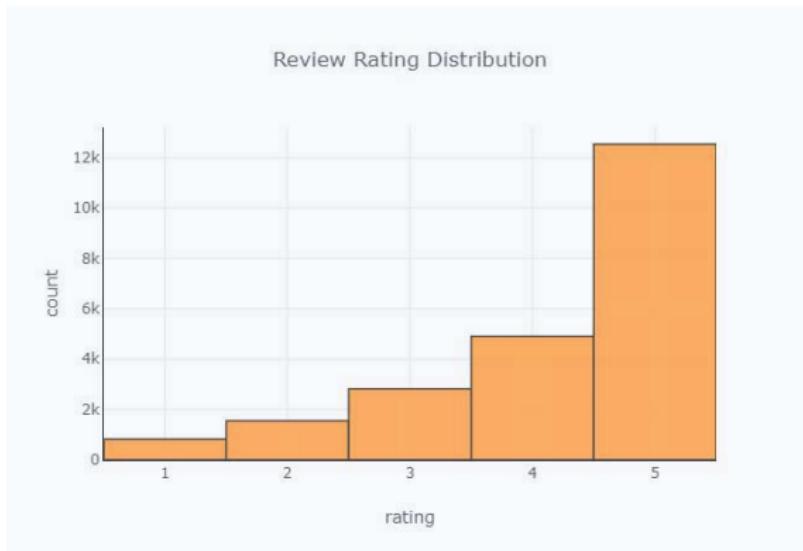
■ Sentiment Polarity Detection Dataset

- Clothing products review text, Reviewer info, rating and sentiment
- Sentiment labels $\in \{-1, 0, 1\} = \{\text{Negative}, \text{Neutral}, \text{Positive}\}$
- The problem is treated as Regression
- Polarity distribution plot



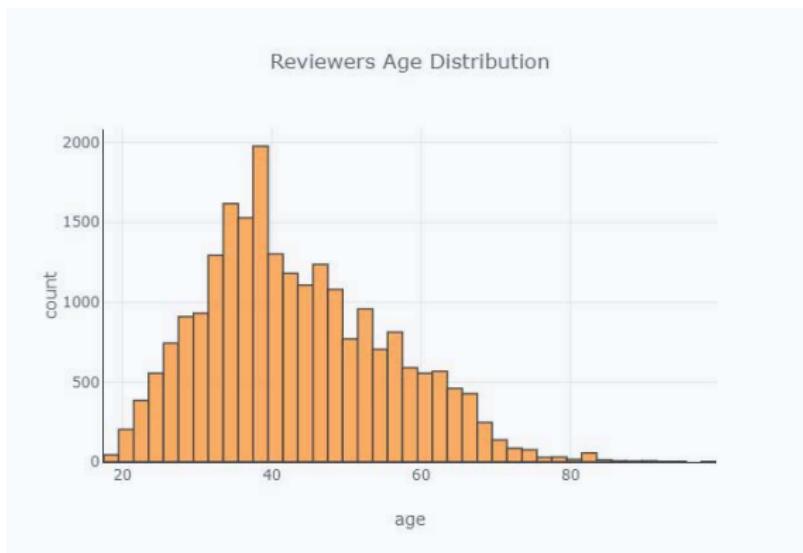
Text Exploratory Data Analysis

■ Rating distribution



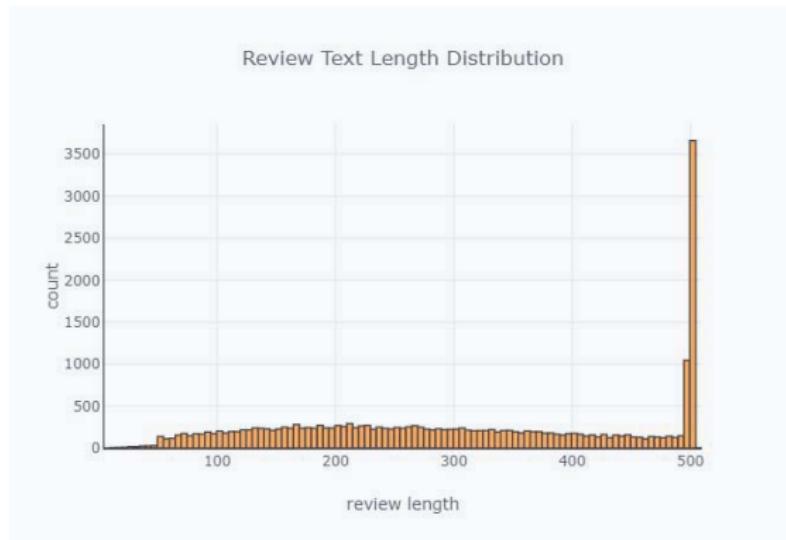
Text Exploratory Data Analysis

■ Distribution of age of the reviewers



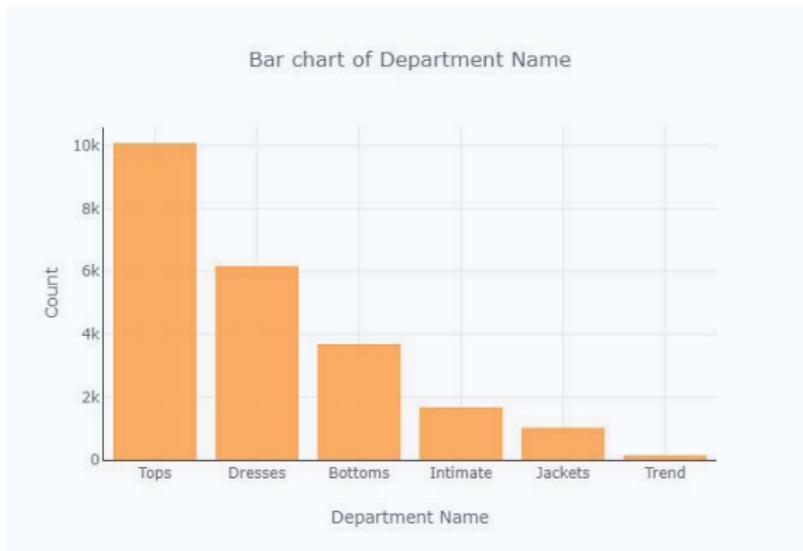
Text Exploratory Data Analysis

■ Distribution of the text length of the reviews



Text Exploratory Data Analysis

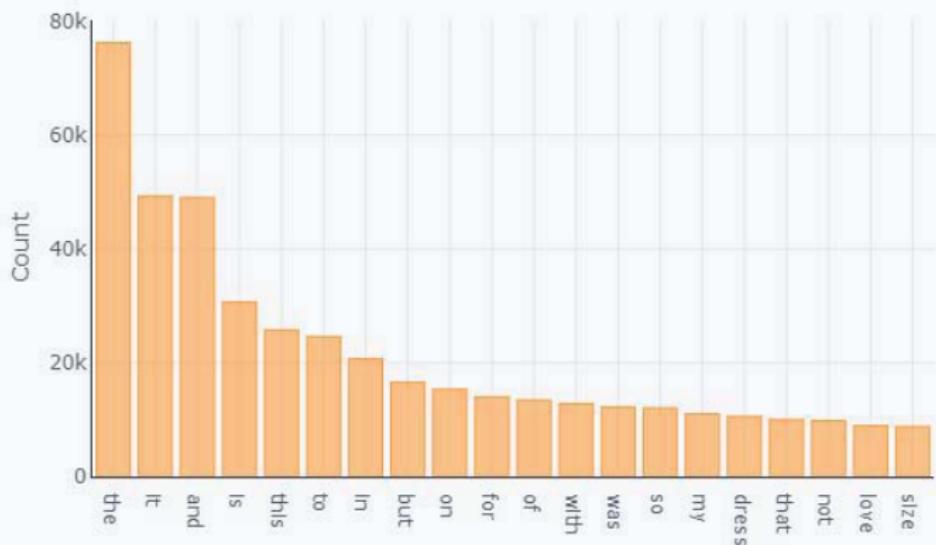
■ Reviews per department



Text Exploratory Data Analysis

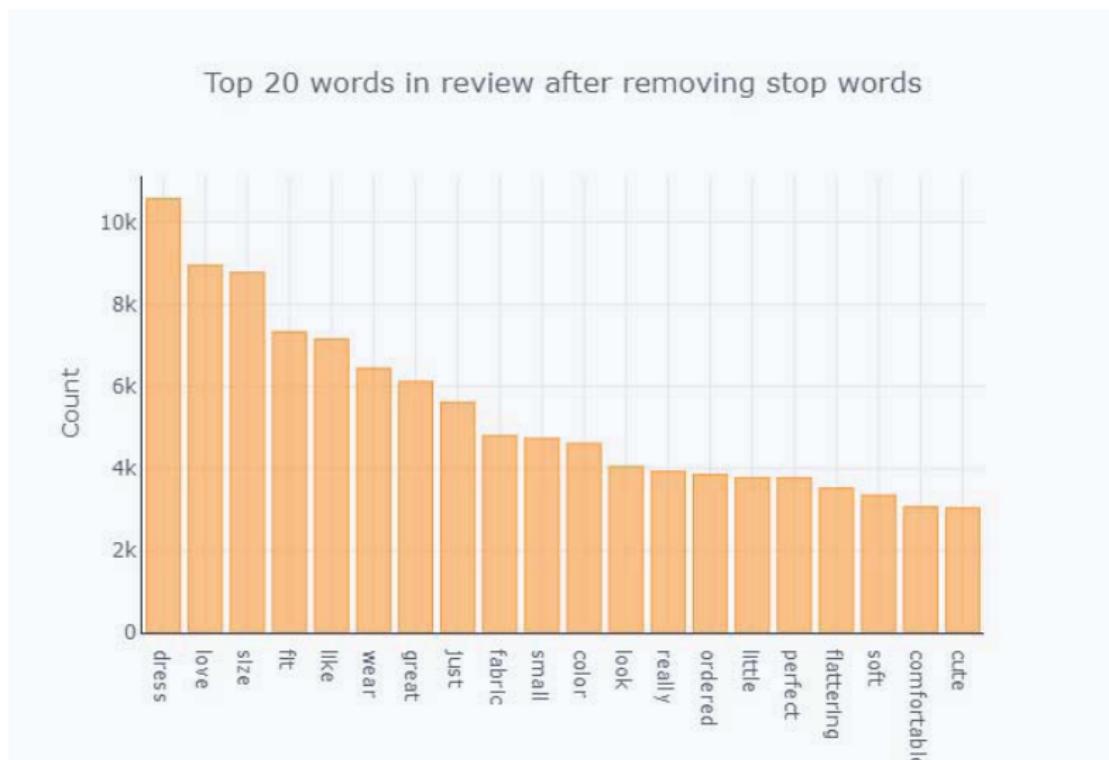
- Frequency of top unigrams **before** removing stopwords

Top 20 words in review before removing stop words



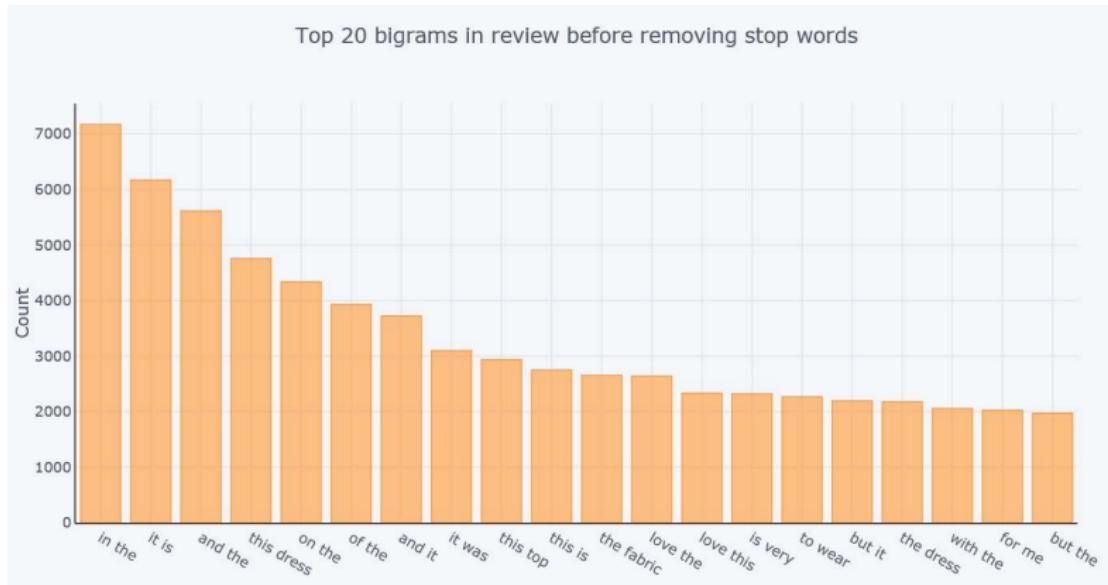
Text Exploratory Data Analysis

- Frequency of top unigrams **after** removing stopwords



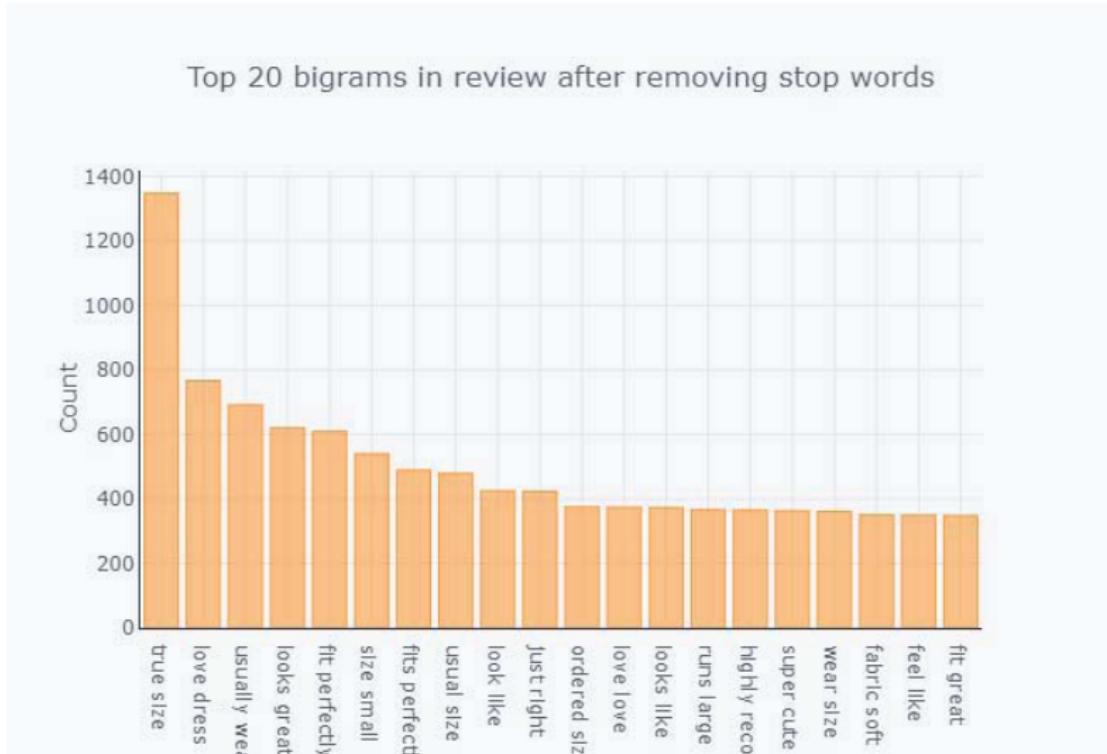
Text Exploratory Data Analysis

■ Frequency of top bigrams **before** removing stopwords



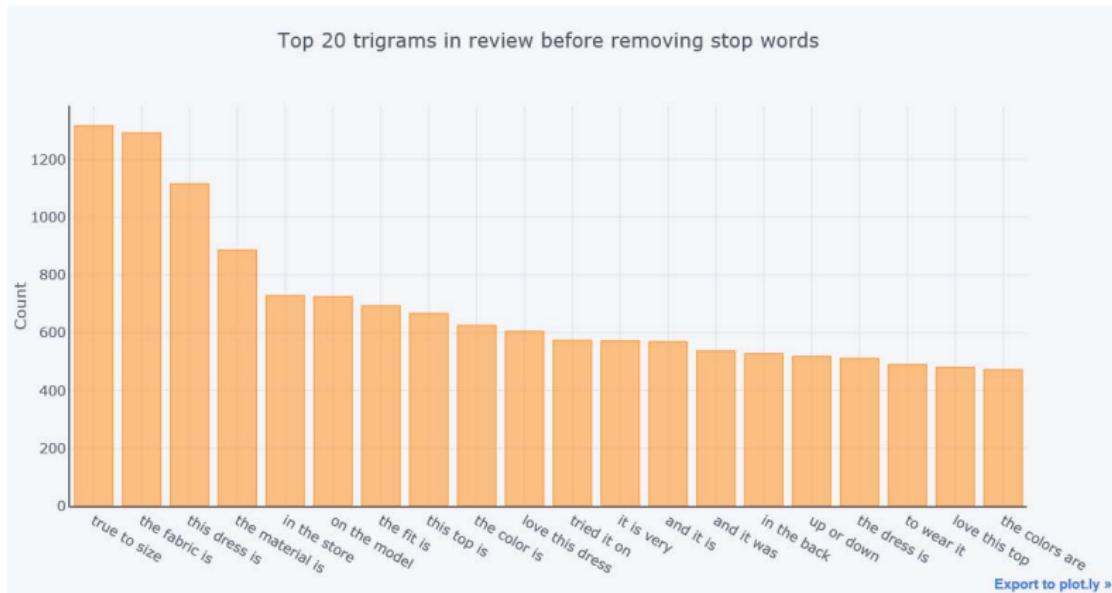
Text Exploratory Data Analysis

- Frequency of top bigrams **after** removing stopwords



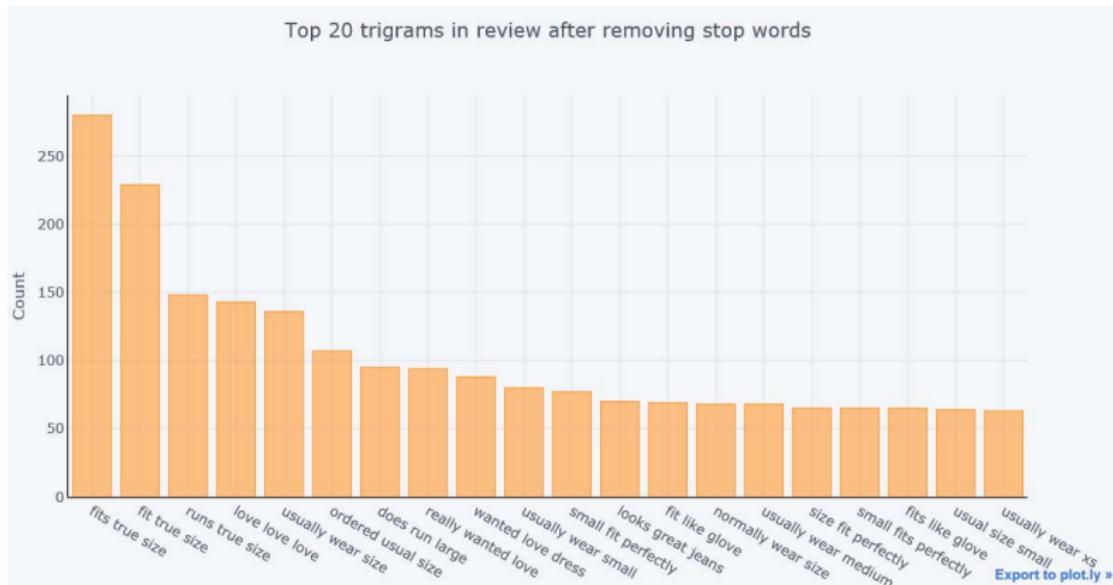
Text Exploratory Data Analysis

- Frequency of top trigrams **before** removing stopwords



Text Exploratory Data Analysis

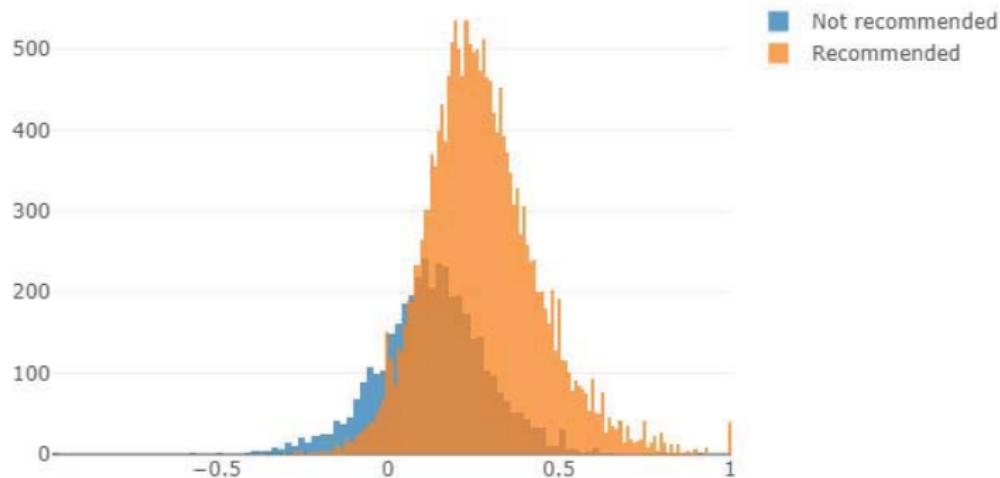
■ Frequency of top trigrams **after** removing stopwords



Text Exploratory Data Analysis

- Visualizing class-wise polarity distribution
- Shows the threshold of sentiment score after which people tend to recommend clothing

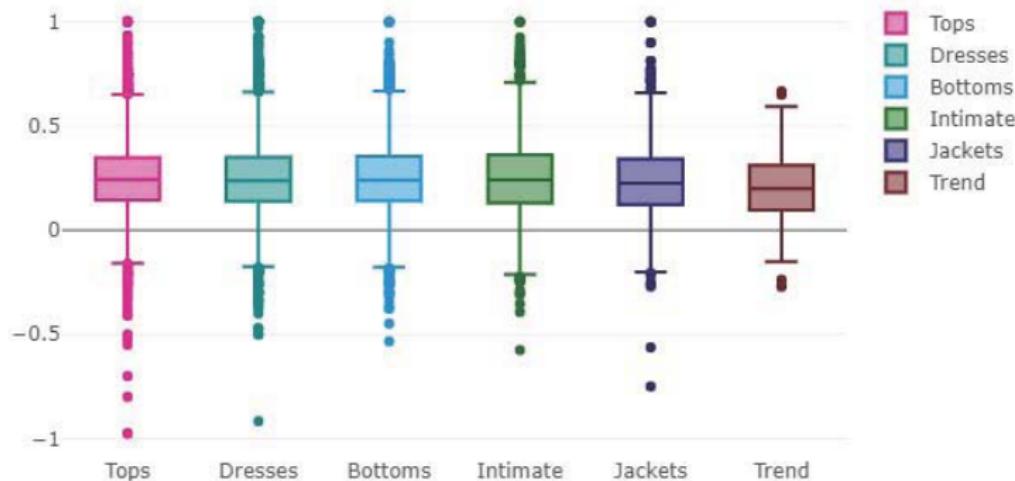
Distribution of Sentiment polarity of reviews based on Recommendation



Text Exploratory Data Analysis

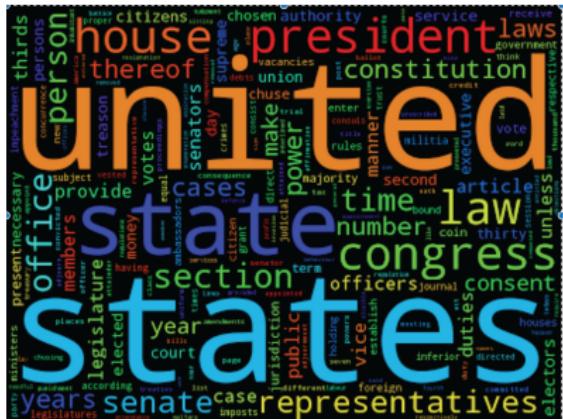
- Visualizing department wise sentiment polarity via boxplot
- Shows the statistical summary of the values

Sentiment Polarity Boxplot of Department Name



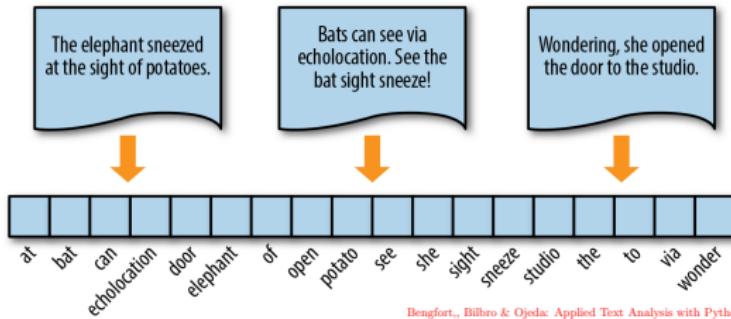
Text Exploratory Data Analysis

- An integral tool for text EDA is **Word Cloud**
 - What could be said about the texts by looking at below examples?



Vector Space Models

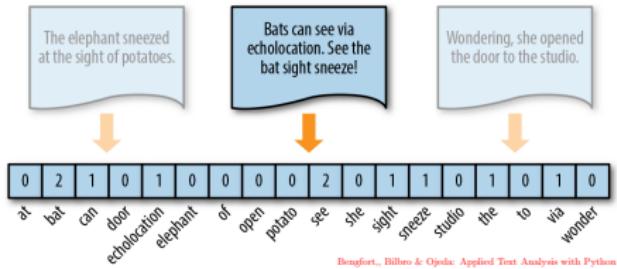
- Algorithms cannot work with raw texts directly
- Calculate similarity/difference between two documents?
- Convert texts to vectors. Vector Space Modeling



- Extract features from texts to reflect linguistic properties of the text
- Popular feature extraction methods (VSM variations) are
 - Set-of-Words: Binary word occurrences
 - Bag-of-Words: Word occurrences
 - TF-IDF
 - Word embedding

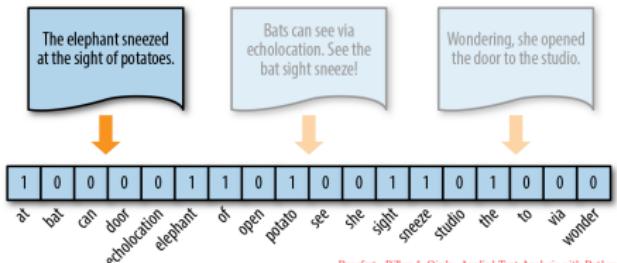
Set and Bag of Words Models

- Text represented as a set or a bag (multiset) of words it contains
- Disregard grammar and word order
- Binary Word Occurrences (Set of Words)



Bengfort., Billiou & Ojeda: Applied Text Analysis with Python

- Word Occurrences (aka Term Frequency) (Bag of Words)
- Bag-of-Words model is Set-of-Words but it accounts for frequencies



Bengfort., Billiou & Ojeda: Applied Text Analysis with Python

The Set-of-Words Model

Each document is represented by a vector $\in \{0, 1\}^{|\Sigma|}$

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	1	1	0	0	0	1	
BRUTUS	1	1	0	1	0	0	
CAESAR	1	1	0	1	1	1	
CALPURNIA	0	1	0	0	0	0	
CLEOPATRA	1	0	0	0	0	0	
MERCY	1	0	1	1	1	1	
WORSER	1	0	1	1	1	0	
...							

The Bag of words Model

Each document is represented by a (term) frequency vector $\in \mathbb{N}^{|\Sigma|}$

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	157	73	0	0	0	1	
BRUTUS	4	157	0	2	0	0	
CAESAR	232	227	0	2	1	0	
CALPURNIA	0	10	0	0	0	0	
CLEOPATRA	57	0	0	0	0	0	
MERCY	2	0	3	8	5	8	
WORSER	2	0	1	1	1	5	
...							

Bag of Words

Issues with Sets and Bag of Words

- Set representation has associated high computational complexity
- dimensionality blow up, $|\Sigma|$ could be very large
- (SoW) treats mere appearance of words as feature of document
(Word appearing 1000 times versus one appearing once only)

TF-IDF - Motivation

TF-IDF is a more refined model to select features to represent texts

- Key idea is to find special words characterizing the document
- Reflect how significant a word is to a “document” in a “collection”
- **Frequency:** Most frequent words implies most significant in doc
- Actually exactly the opposite is true
- Most frequent words (“the”, “are”, “and”) help English structure and build ideas but not significant in characterizing documents
- **Rarity:** Indicator of topics are rare words
- rare words overall but concentrated in a few docs “batsman”, “prime-minister”
- ball, bat, pitch, catch, run \implies cricket related doc
- An indicator word is likely to be repeated if it appears once

TF-IDF

- TF-IDF value increases proportionally to the number of times a word appears in a document
- Offset by the number of documents in corpus containing that word
- Best known weighting scheme in IR. Value for a term increases with
 - Number of occurrences within a document
 - Rarity of the term in collection
- Helps to adjust for the fact that some words appear more frequently in general (frequent words are less meaningful than the rare ones)
- Involve two characteristics of words (terms: bigram, trigram)
 - Term frequency
 - Inverse document frequency

TF-IDF: Term Frequency

Documents: D_1, \dots, D_N . Terms (Σ): t_1, \dots, t_m

- **Frequency, f_{ij} :** frequency of term t_i in document D_j
- Find a parameter to measure importance of t_i to D_j
- f_{ij} is not good, (very high for stop words in all documents)
- It is also possible that large docs D_j (books) have larger f_{ij} , than $f_{ij'}$ of short document $D_{j'}$, even if t_i is more important for $D_{j'}$ than D_j
- Recall normalization and scaling
- **Term Frequency:** $\text{TF}_{ij} := \frac{f_{ij}}{\max_i f_{ij}}$
- Most frequent term t_i in D_j gets $\text{TF}_{ij} = 1$ others are < 1

TF-IDF: Inverse Document Frequency

Documents: D_1, \dots, D_N . Terms (Σ): t_1, \dots, t_m

- Term frequency considers all t_i equally important
- Stop words appear frequently but have little importance
- Need to weigh down the frequent terms while scale up the rare ones
- Some terms are rare but appear in many documents a few times
- Weigh TF_{ij} (inversely) by the term's overall popularity in collection
- Suppose the term t_i appears in n_i out of N documents. Then
- **Inverse Document Frequency:** $IDF_i := \log \left(\frac{N}{n_i + 1} \right)$
- +1 in denom. avoids dividing by 0 if t_i doesn't appear in any doc

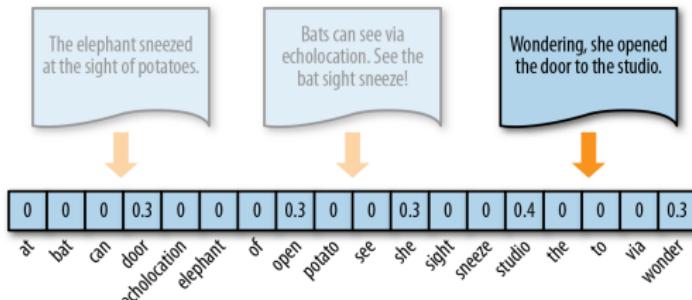
TF-IDF: Term frequency-inverse document frequency

Documents: D_1, \dots, D_N . Terms (Σ): t_1, \dots, t_m

- Finally, weight or importance of a term t_i in document D_j is given as

$$\text{TF-IDF}(i,j) = \text{TF}_{ij} \times \text{IDF}_i$$

- Check the extreme cases
- If t_i appears in all the documents, then $\text{TF-IDF}(i,j) = 0$ in all D_j
- Many stop words would get score close to 0
- If a term appears a few times in a few documents it gets higher score there but less in other



TF-IDF: Example

- D_1 : “The car is driven on the road”
- D_2 : “The truck is driven on the highway”

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	$\log(2/2) = 0$	0	0
Car	1/7	0	$\log(2/1) = 0.3$	0.043	0
Truck	0	1/7	$\log(2/1) = 0.3$	0	0.043
Is	1/7	1/7	$\log(2/2) = 0$	0	0
Driven	1/7	1/7	$\log(2/2) = 0$	0	0
On	1/7	1/7	$\log(2/2) = 0$	0	0
The	1/7	1/7	$\log(2/2) = 0$	0	0
Road	1/7	0	$\log(2/1) = 0.3$	0.043	0
Highway	0	1/7	$\log(2/1) = 0.3$	0	0.043

- Common words score is zero (not significant)
- Score of “car”, “truck”, “road”, and “highway” are non-zero (significant words)

THE tf-idf Model

Each document is represented by a real vector of TF-IDF weights $\in \mathbb{R}^{|\Sigma|}$

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35	
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0	
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0	
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0	
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0	
MERCY	1.51	0.0	1.90	0.12	5.25	0.88	
WORSER	1.37	0.0	0.11	4.15	0.25	1.95	
...							

Vector Space Models

- Consider two documents:
 - “worst acting, worst plot, worst movie ever”
 - “best acting, best movie ever”

- Binary Word Occurrences (Set of Words)

Row No.	acting	best	ever	movie	plot	worst
1	1	0	1	1	1	1
2	1	1	1	1	0	0

- Word Occurrences (aka Term Frequency) (Bag of Words)

Row No.	acting	best	ever	movie	plot	worst
1	1	0	1	1	1	3
2	1	2	1	1	0	0

- TF-IDF

Row No.	acting	best	ever	movie	plot	worst
1	0	0	0	0	0.316	0.949
2	0	1	0	0	0	0

Vector Space Models

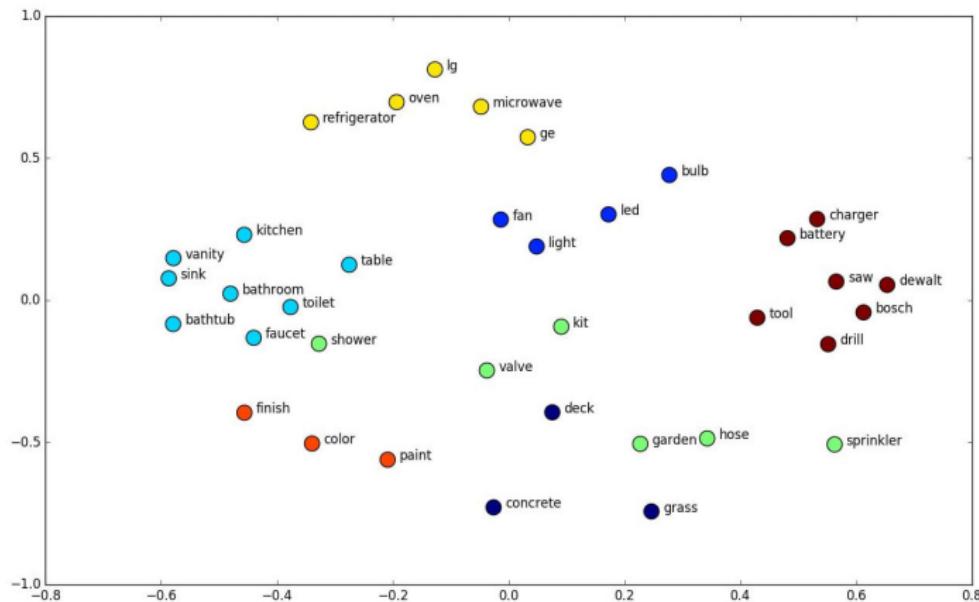
Problems with previous 3 VSM models

- What if we have very large vocabulary? Notice the problem of **sparsity**. This leads to curse of dimensionality
- All three models do not preserve order of the words. In linguistics, order carries very important contextual information
- Consider two documents:
 - *mary is faster than john*
 - *john is faster than mary*
- Both documents will produce identical vectors (in all 3 models), although the context and meaning is very much different
- They ignore synonyms (“old bike” vs “used bike”) and homonyms
- *n*-gram model of vocabulary takes care of context to some extent

Solution: Word embedding

Vector Space Models: Word embedding

- Represent each word with n dimensional **dense** vector
- Words that appear in similar context should be mapped to close-by points in Euclidean space
- Neural network is used to learn these mappings



Vector Space Models: Word embedding

- Can be extended to learn document level embeddings
- Consider following 2-D representation of n-dimensional document level embeddings. (You can convert any n-dimensional vector to 2-D vector by TSNE or PCA)

