

UNIVERSITY *of* WASHINGTON

Data Science UW

Methods for Data

Analysis

Intro to Natural Language Processing

Lecture 9

Nick McClure



SPAMMERS ARE BREAKING TRADITIONAL CAPTCHAS WITH AI, SO I'VE BUILT A NEW SYSTEM. IT ASKS USERS TO RATE A SLATE OF COMMENTS AS "CONSTRUCTIVE" OR "NOT CONSTRUCTIVE."



THEN IT HAS THEM REPLY WITH COMMENTS OF THEIR OWN, WHICH ARE LATER RATED BY OTHER USERS.



BUT WHAT WILL YOU DO WHEN SPAMMERS TRAIN THEIR BOTS TO MAKE AUTOMATED CONSTRUCTIVE AND HELPFUL COMMENTS?



MISSION.
[REDACTED]
ACCOMPLISHED.



W

Topics

> Natural Language Processing

- Text Normalization
- Text Distances
- Corpus/Dictionaries
- Naïve Bayes
- Word Frequencies
- Latent Dirichlet Allocation



Why Text?

> How much data?

- Twitter has more text data recorded than all that has been written in print in the history of mankind. (<http://www.internetlivestats.com/twitter-statistics/>)
- Most of the world's data is unstructured:
 - > 2009 HP Survey: 70%
 - > Gartner: 80%
 - > Teradata: 85%

> Why use text?

- Structured (numerical/categorical) data very often misses elements critical to modeling.
 - > Un-transcribed notes, comments, logs
 - > Surveys, medical charts



Measuring Text Distance

> Hamming Distance

- Line up strings, count number of positions that are the different.
- Assumes strings are of the same length.

$$\text{Hamming}(\text{beer}, \text{bear}) = 1 \qquad \text{Hamming}(101101, 100011) = 3$$

> Levenshtein distance

- Measures edit distance between two strings (insertion, deletion, substitution only)

$$\text{Lev}(\text{beer}, \text{bear}) = 1 \qquad \text{Lev}(\text{banana}, \text{ban}) = 3$$



Measuring Text Differences

> Jaccard index

- Size of intersection of characters divided by size of union of characters.

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

$$J(\text{beer}, \text{bear}) = 1 - \frac{3}{4} \quad J(\text{banana}, \text{ban}) = 1 - \frac{3}{3}$$

> Weighted Jaccard Index

- For each letter, calculate the minimum times it appears, m_i and the max, M_i

$$J'(A, B) = 1 - \frac{\sum m_i}{\sum M_i}$$

$$J(\text{beer}, \text{bear}) = 1 - \frac{m_a + m_b + m_e + m_r}{M_a + M_b + M_e + M_r}$$

$$J(\text{beer}, \text{bear}) = 1 - \frac{0 + 1 + 1 + 1}{1 + 1 + 2 + 1} = 1 - \frac{3}{5}$$

$$J(\text{banana}, \text{ban}) = 1 - \frac{m_a + m_b + m_n}{M_a + M_b + M_n}$$

$$J(\text{banana}, \text{ban}) = 1 - \frac{1 + 1 + 1}{3 + 1 + 2}$$

This may be an issue

> R demo

W

Text Normalization (Pre processing)

> Strip extra white space:

I <3 statistics, it's my \u1072 fAvoRitE!! 11!!! → I <3 statistics, it's my \u1072 fAvoRitE!! 11!!!

> Remove Unicode text

I <3 statistics, it's my \u1072 fAvoRitE!! 11!!! → I <3 statistics, it's my fAvoRitE!! 11!!!

> Lower case

I <3 statistics, it's my fAvoRitE!! 11!!! → i <3 statistics, it's my favorite!! 11!!!

> Remove punctuation

i <3 statistics, it's my favorite!! 11!!! → i 3 statistics its my favorite 11

> Remove numbers

i 3 statistics its my favorite 11 → i statistics its my favorite

> Remove stop words

i statistics its my favorite → statistics favorite

> Stem words (optional)

statistics favorite → statisti favori



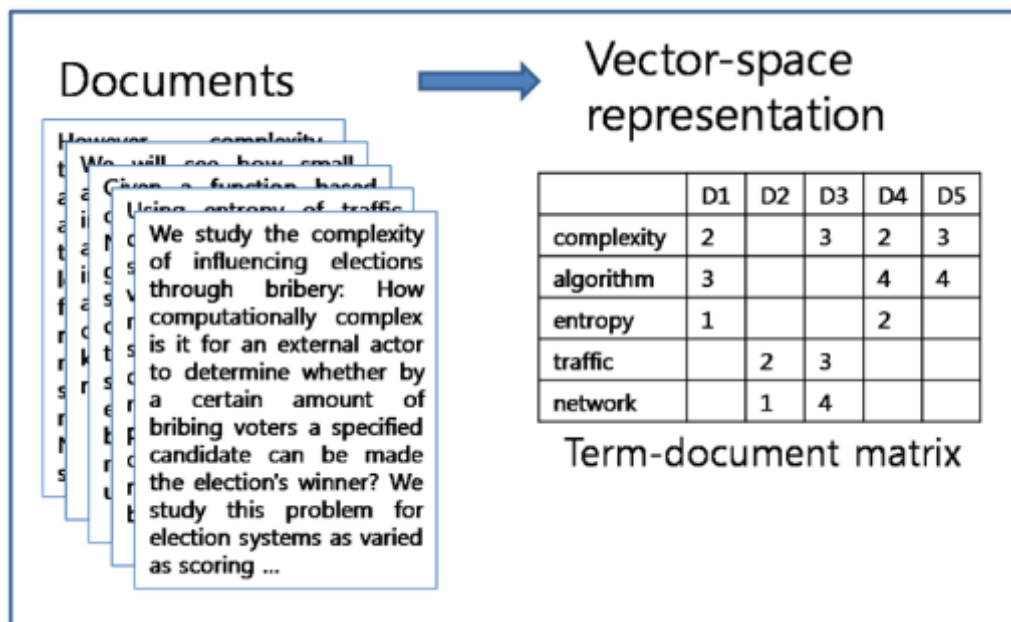
Text Normalization (Pre processing)

- > Remember that order is important when processing text!!
- > (Lowercase then stopwords removal) vs (stopword removal then lowercase). This can cause problems if stopwords are all lowercased.
- > Dealing with punctuation:
 - can't → can t or cant
 - we'll → well or we ll
- > R Demo



Dictionary Creation

- > Usually we have a whole bunch of entries, each entry is a sequence of variable length text.
- > A document is just one entry.
- > A collection of documents is called a corpus.
- > We can represent that corpus many ways.
- > Term document matrix (shown below) is just a count of how many times a word appears in a document.



W

Dictionary Creation

- > Term document matrix is the transpose of a document term matrix.
- > The larger the corpus, the more sparse the matrix of counts.
 - It is common practice to drop terms (rows in the term-document matrix) that have occurred less than X times.
- > We can also consider the column vector of a term-document matrix as a vector of numbers that represent the document.

Document similarity: Term-document matrix

- Two documents are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

W

Naïve Bayes!

- > Now we have a numeric representation of key words in our documents.
- > We can use called Naïve Bayes algorithm to classify documents.
- > Remember:

$$P(A|B) = P(B|A) \frac{P(A)}{P(B)}$$

Or

$$P(h_0|Data) = P(Data|h_0) \frac{P(h_0)}{P(Data)}$$



Another Application of Bayes Theorem (Naïve Bayes Classification)

- > Naïve Bayes Classification.
- > Let's say we want to test a hypothesis, h_0 :

$$P(h_0|Data) = P(Data|h_0) \frac{P(h_0)}{P(Data)}$$

- > We pick either the null or alternative, based on which has the highest probability given the data.



Another Application of Bayes Theorem

$$P(h_0|Data) = P(Data|h_0) \frac{P(h_0)}{P(Data)}$$

- > We want to predict if a consumer will buy an ad:
- > Null hypothesis is yes (or could be no)
- > We have observed 10 outcomes in the past:

Age	Income	Student	Credit	Buys Ad
35	Medium	Yes	Fair	Yes
30	High	No	Average	No
40	Low	Yes	Good	No
35	Medium	No	Fair	Yes
45	Low	No	Fair	Yes
35	High	No	Excellent	Yes
35	Medium	No	Good	No
25	Low	No	Good	No
28	High	No	Average	No
35	Medium	Yes	Average	Yes



Another Application of Bayes Theorem

Age	Income	Student	Credit	Buys Ad
35	Medium	Yes	Fair	Yes
30	High	No	Average	No
40	Low	Yes	Good	No
35	Medium	No	Fair	Yes
45	Low	No	Fair	Yes
35	High	No	Excellent	Yes
35	Medium	No	Good	No
25	Low	No	Good	No
28	High	No	Average	No
35	Medium	Yes	Average	Yes

$$P(h_0|Data) = P(Data|h_0) \frac{P(h_0)}{P(Data)}$$

New Data: Consumer is 35 years old
And has a medium income

- > $P(\text{buys Ad})=0.5$, $P(\text{not buy Ad}) = 0.5$
- > $P(35\text{yrs \& med income}) = 4/10 = 0.4$
- > $P(35\text{yrs \& med income} | \text{buys ad}) = 3/5 = 0.6$
- > $P(35\text{yrs \& med income} | \text{not buy ad}) = 1/5 = 0.2$

$$> P(\text{buy} | \text{demographics}) = P(\text{demographics}|\text{buy}) \frac{P(\text{buy})}{P(\text{demographics})}$$

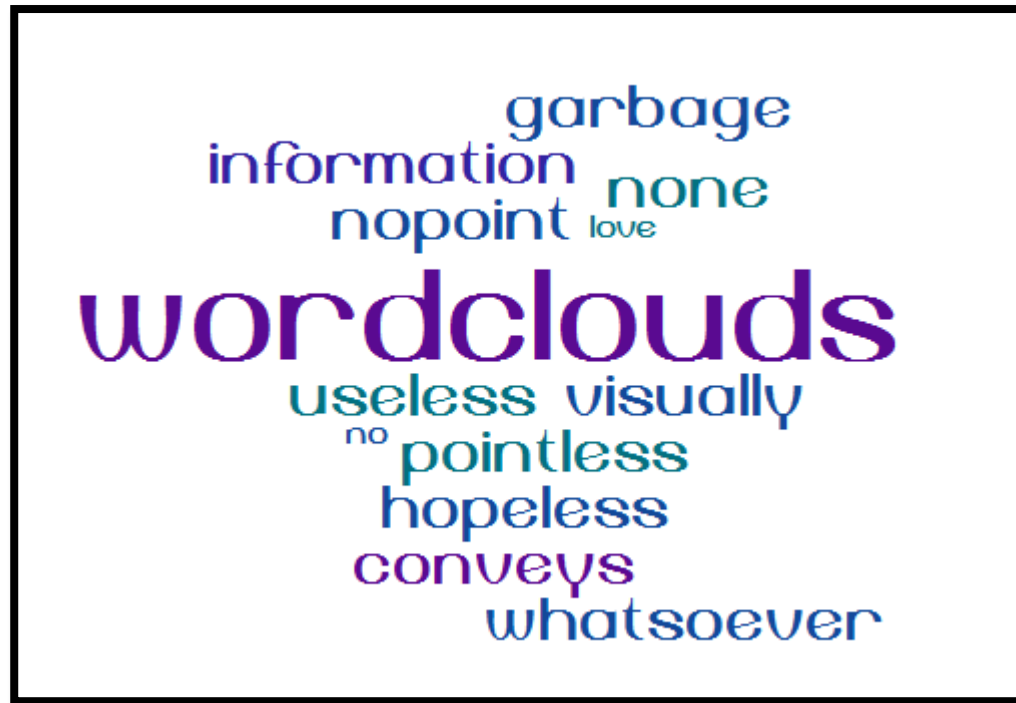
$$> =0.6 * (0.5 / 0.4) = 0.75$$

$$> \text{Similarly, } P(\text{not buy} | \text{demographics}) = 0.2 * (0.5 / 0.4)=0.25$$

W

Wordclouds

- > Completely useless display of information that people love to see.
- > R demo



W

Creating Features

- > Text Frequency: Word frequency or TF
 - Count how many times a word appears in a single document.
 - Terms that occur in fewer documents are more descriptive and may contain more information (Rarity matters).
- > Inverse Document Frequency (IDF)
 - Inverse of the proportion of documents containing term in the whole collection.
 - $\#documents / \#documents \text{ with word}$ might be too severe.
 - > A word appearing twice instead of once shouldn't have twice the impact

$$IDF = \log \left(\frac{\#Documents}{\#Documents \text{ with Word}} \right)$$

- > Maybe we should tie these together:
- > TF-IDF: Rare terms in whole collection that appear frequently in some documents maybe very important!
 - Multiply these two



How to Use TF-IDF

$$TF - IDF = \log \left(\frac{\#Documents}{\#Documents\ with\ Word} \right) \times f(Word)$$

- > Finding important words to describe the document collections or subgroups of collections.
- > Using the count of important words as a feature in a model.
- > Using the distribution of a document's TF-IDF values.
 - Characterize writing styles
 - Comparing authors
 - Determining original authors
 - Finding plagiarism
- > R-demo



Sentiment Analysis

- > There are some words that instinctively bring up positive and negative thoughts.
 - Positive: nice, good, happy, These give a +1 count.
 - Negative: two-faced, abhor, hate, These give a -1 count.
- > We could do a count or average over a document to come up with a average sentiment for the text.



Topic Modeling

- > We sometimes would like to divide our corpus of documents into naturally occurring groups.
 - Help Forums into product groups
 - Wikipedia articles into subjects
 - Texts/messages into +/- sentiment
- > LDA (Latent Dirichlet Allocation) is a method for fitting a topic model.
 - Each document is a mixture of topics.
 - Each topic is a mixture of words.
- > Every document can have multiple topics!



Topic Modeling

- > Algorithm:
- > For each document:
 - Go through and randomly assign each word in the document to one of the K topics.
 - Note: This gives us a representation of all document and word distributions. (not very good ones).
- > For each document (d):
 - For each word in document (w):
 - > For each topic (t):
 - Compute $P(t|d)$ = proportion of words in document for topic.
 - Compute $P(w|t)$ = over all documents, proportion of this word assigned to topic t .
 - > Then reassign topic according to highest probability of topic generating word w : $P(t|d) \cdot P(w|t)$
- > Repeat many times until distributions stop changing.



Topic Modeling

- > Now, given any document, we can predict the distribution of topic probabilities!
- > In other words, we can translate a document of text into a vector of categorical probabilities.
- > We can use such a numerical vector to help in prediction of other things.
- > R Demo



Part of Speech Tagging

- > Tag each word in a sentence with what kind of word it is.
- > There are 36 canonical types:

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	<i>and, but, or</i>	SYM	symbol	<i>+, %, &</i>
CD	cardinal number	<i>one, two</i>	TO	“to”	<i>to</i>
DT	determiner	<i>a, the</i>	UH	interjection	<i>ah, oops</i>
EX	existential ‘there’	<i>there</i>	VB	verb base form	<i>eat</i>
FW	foreign word	<i>mea culpa</i>	VBD	verb past tense	<i>ate</i>
IN	preposition/sub-conj	<i>of, in, by</i>	VBG	verb gerund	<i>eating</i>
JJ	adjective	<i>yellow</i>	VBN	verb past participle	<i>eaten</i>
JJR	adj., comparative	<i>bigger</i>	VBP	verb non-3sg pres	<i>eat</i>
JJS	adj., superlative	<i>wildest</i>	VBZ	verb 3sg pres	<i>eats</i>
LS	list item marker	<i>1, 2, One</i>	WDT	wh-determiner	<i>which, that</i>
MD	modal	<i>can, should</i>	WP	wh-pronoun	<i>what, who</i>
NN	noun, sing. or mass	<i>llama</i>	WP\$	possessive wh-	<i>whose</i>
NNS	noun, plural	<i>llamas</i>	WRB	wh-adverb	<i>how, where</i>
NNP	proper noun, sing.	<i>IBM</i>	\$	dollar sign	<i>\$</i>
NNPS	proper noun, plural	<i>Carolinas</i>	#	pound sign	<i>#</i>
PDT	predeterminer	<i>all, both</i>	“	left quote	<i>‘ or “</i>
POS	possessive ending	<i>’s</i>	”	right quote	<i>’ or ”</i>
PRP	personal pronoun	<i>I, you, he</i>	(left parenthesis	<i>[, (, {, <</i>
PRP\$	possessive pronoun	<i>your, one’s</i>)	right parenthesis	<i>],), }, ></i>
RB	adverb	<i>quickly, never</i>	,	comma	<i>,</i>
RBR	adverb, comparative	<i>faster</i>	.	sentence-final punc	<i>. ! ?</i>
RBS	adverb, superlative	<i>fastest</i>	:	mid-sentence punc	<i>: ; ... --</i>
RP	particle	<i>up, off</i>			

Figure 10.1 Penn Treebank part-of-speech tags (including punctuation).



Part of Speech Tagging


- > Single Word Lookup tables:
 - Look up word with part of speech.
- > The major problem is that words in the English language can take many parts of speech.
 - ‘I will book a flight’, vs. ‘That is a book’
 - ‘**Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo**’
 - > ‘Buffalo from Buffalo that are intimidated by buffalo from Buffalo intimidate buffalo from Buffalo.’
- > While ambiguous words account for ~15% of the English dictionary, they make up ~60% of common word usage.
- > If we just use the most frequent tag for ambiguous words:
 - Obtain about 90% accuracy overall on English.



Part of Speech Tagging

- > We can improve on this tagging by taking into account the types of words nearby the target word.
 - We maximize a sequence of POS tags on a sequence of words.
- > Idea is to train a classifier that takes into account the nearby words to predict current word.

$$t_n = \operatorname{argmax} \prod P(w_i | t_i) P(t_i | t_{i-1})$$


Emission of Tag Transition of Tag

- There are large sources of already pre-tagged texts out there for us to base these probabilities on.
- This method is called a “Hidden Markov Model”.
- For example, if we observe the weather everyday, we want to devise a day-to-day model that predicts the next day based on the prior day. We want to choose probabilities of transitions between weather states that maximizes the $P(\text{data}|\text{parameters})$:

W

Further interesting links

- > Text Mining with R. 2017. <http://tidytextmining.com/>
- > Project detecting sarcasm of tweets:
<http://www.thesarcasmdetector.com/about/>
- > Detecting Sarcasm in Text: An Obvious Solution to a Trivial Problem: http://cs229.stanford.edu/proj2015/044_report.pdf
- > Why Sentiment analysis is so hard:
- > <http://idibon.com/why-is-sentiment-analysis-hard/>
- > Brief overview of the area of text mining:
http://www.ntu.edu.sg/home/asahtan/papers/tm_pakdd99.pdf

