NAVAL
POSTGRADUATE
SCHOOL

# OA3802

## Computational Methods for Data Analytics

# Text Mining and Natural Language Processing

Prof. Sam Buttrey

Fall AY 2020

# Admin Notes

- This is week 9 of 11 (not 10)
  - Remaining topics:
  - Spatial statistics and mapping
  - Visualization
  - Anything else I can think of
- New AWS instance starting 3 p.m. today
  - All your files will be deleted!

# Thrusts in Text Mining

- Search and Retrieval
- Computational Linguistics
- Natural Language Processing
- …And of course these overlap

# Search and Retrieval

- **Search** tries to find documents that match a words (or sets of words)
- Difficult to handle differences in spelling, synonyms, etc. automatically
- Direction: XML/XHTML encodes “meta-data” about author, subject, date, etc.
- No new information generated
- No claim about understanding

# Search vs Text Mining

- Search find documents that match
- Text mining seeks to deduce **meaning** from text (examples: IED reports, EOD team reports, Marine PCR's)…
- …Or at least establishing similarities among documents

# Computational Linguistics

- Reads lots of text docs., tries to extract statistical-type data (frequencies, co-occurrences etc.)
- That data would then be used in algo-rithms to locate parts of speech, resolve ambiguities, translate and so on
- Operates in aggregate over large bodies of documents

# Natural Language Processing

- Turn sentences typed by humans into machine-readable “thoughts”
- “Safety is our organization’s first priority”
- “I feel the XO puts too much emphasis on speed and not enough on safety”
- Very difficult in any language; perhaps it will never be foolproof since even humans can’t do it perfectly
  - Words have multiple meanings, and so can sentences: “the man saw a boy with a telescope”

# Free-Form Text

- Text mining requires us to extract information from **free-form** text
- Examples of data:
  - Web pages, blogs
  - Tweets and other social media input
  - Open-ended survey responses
  - Incident reports, press releases
  - Documents like theses, patent apps, journal articles

# Free-Form Text

- Examples of output:
  – Categorization, classification
  – Social media trends, early warning
  – Sentiment analysis; positive and negative reviews
  – Identify topics in document, where longer documents might span topics
  – Label images using a combination of neural nets for the image and some sort of text analysis for the label

# Text Mining Tasks: Low Level

- “Tokenizing” (extracting words)
- Stemming/lemmatization, removal of stop words
- Part-of-speech tagging
  - Lots of English words have ambiguous senses (noun and verb, for example)
- Named Entity Extraction
  - Produce a list of the people, places, dates, organizations, amounts (etc.) in a document
- Resolution of Coreferences
- Constructing the **term-document matrix**

# Tokenization

- English words are separated by spaces, but:
  - Tokens should be “linguistically significant” and “methodologically useful”
  - Some tokens need two words (“kung fu”); some combinations are ambiguous (“Down Under”)
  - Punctuation and abbreviations can confuse
    - “where is meadows dr. who asked”
    - Some hyphens split end-of-line words; others don’t
    - “She’s” = “she has” or “she is”
  - Dates and times, phone #s, e-mail/street addresses, SSNs, book citations…

https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en

# Stemming

- **Stemming** is the replacement of inflected forms by their base form (turn “bringing” but maybe not “brought” into “bring”)
- Stemming is done one word at a time, without neighboring context; sometimes produces a root that is not a word
- **Lemmatization** uses the context of nearby words, maybe parts of speech indicators, to find the “lemma” – the root – that is always a word.
- Ex.: “dove” (n) $\Rightarrow$ dove; “dove” (v) $\Rightarrow$ dive”

# Stop Words

- **Stop words** are words that can be removed with little loss of meaning (“the,” “who” -- unless you’re looking for “The Who” or “The The”) but this might be context-dependent (“first” and “second”)
- Most of these language-specific tasks will need to be done quite differently in other languages
  – One text handling tool, Udpipe, handles ~50 languages

# Part-of-Speech Tagging

Chars A dog is chasing a boy on the playground

Tokens A dog is chasing a boy on the playground

Det Noun Aux Verb Det Noun Prep Det Noun

Part of Speech (POS) Tagging (97%)

Noun Phrase

Verb Phrase

Prep Phrase

Verb Phrase

Syntactic Structures Parsing (>90%)

Sentence

Semantics: some aspects
-**Entity**/relation extraction
-Word sense disambiguation
-**Sentiment** analysis

A dog → a boy → a playground

Animal CHASE **Person** ON **Place**

Taken from Coursera: Natural Language Content Analysis; ChengXiang Zhai (2015)

# Inter-entity distances (again)

- We may need to compute distances between character strings for web search to check spelling, to identify duplicates, or for record linkage (entity disambiguation)
  - We've met edit (Levenshtein) distance
  - Alternatives include Jaro-Winkler distance for comparing census entries (e.g. Geraldine Massey vs. Jeraldine Massie)
- Your text probably has typos!

# Coreferences

- One thing can have two names (e.g. Napoleon, Bonaparte)
- One thing can have multiple titles (king, emperor), nicknames, patronymics
- A thing can be referred to by personal or relative pronouns, “former”/“latter,” etc.
- E.g. Ruth B. Ginsberg: “the Justice,” “the New York City native,” “she,” …
- We would like to be clear about what thing each noun (etc.) in the text refers to

# Relationships between Words

- Words exhibit synonymy, polysemy (same word, multiple meanings), plus hyper- and hyponymy
  - “Bear” means “carry” but also
  - “Bear” is a hyponym of “mammal” and a hypernym of “grizzly”
- Relationships between words can be **paradigmatic** (substitutable, like “May” and “April”) or **syntagmatic** (frequent co-occurrence, like “car”/“drive,” “cat”/“YouTube”)

# Relationships between Words

- Understanding these relationships can help with POS tagging, entity recognition, acronym expansion, learning of grammar
- Useful in summarizing: “in negative iPhone reviews, which words are most strongly related to ‘battery’?”
- Direct application: broaden search queries (if I search for “Napoleon,” add in Bonaparte)

# Text Mining Tasks: High Level

- Document classification
  - What sort of document is this?
  - Summarization
  - Sentiment Analysis: product reviews, blog "chatter"
  - E.g. LDA (Bag of Words) Model
- Document clustering
  - Which documents go together in groups?
  - E.g. Vector space model
- Many of these tasks use the **term-document matrix**

# Constructing the TDM

- Term-document matrix (or TDM) or its transpose, document-term matrix (DTM)
  - “Term” = “word” or “unique token”
- Start with a corpus (pl. “corpora”) of documents that serves as the training set
- Different corpora for different applications (e.g. biomedical vs. military)
- Suppose you tabulate all the words in the corpus
- Each word has a frequency in the corpus

# Constructing the TDM (cont'd)

- Represent document *d* by a vector of counts across your vocabulary
- The $w^{\text{th}}$ count shows the number of times word *w* appears in the document
  - This is one column of the TDM
- Now construct a column like that for each of *D* documents
- We end up with an $W \times D$ matrix whose $(w, d)^{\text{th}}$ element is the number of times word *w* appears in document *d*

# Adjusting TDM weights

- But some words are common in the corpus; if they're frequent in the document, that's not as interesting as when rare words are frequent in the document
- Moreover, some documents are just longer – they have more words
- So it's reasonable to weight the entries in the TDM to account for these two factors

# One scheme: Tf-idf Weights

- **Term frequency tf$_{ij}$** is the count for each term *i* in doc *j*, normalized by the total number of all terms in doc *j*
- It's a vector whose elements add up to 1 across terms *i* for each document *j*
- It's common to use **tf-idf**: term frequency, inverse document frequency
- Weigh tf$_{ij}$ by, e.g., log(1 + #docs/# with *i*)

# Vector Space Model

- Consider each document (and each query) to be a vector in *W*-space where *W* is the number of terms in the corpus
- Each document is represented by a column in the TDM
- Measure document similarity by the **cosine** of the angle between them
- For $\{a_i\}$, $\{b_i\}$: $\Sigma_{\mathrm{i}}\,(a_i\,b_i) \,/\, \sqrt{[\Sigma_{\mathrm{i}}\,(a_i^2)\,\Sigma_{\mathrm{i}}\,(b_i^2)]}$
- 0 = orthogonal = no overlap; 1 = match

# Principal Components

- The TDM represents documents in a *W*-dimensional space, but…
- …the TDM is very sparse -- mostly zeros
- One idea: use Principle Components to reduce the dimensionality of the space
- Then measure cosine similarity between these new representations
- This is **Latent Semantic Analysis**

NAVAL POSTGRADUATE SCHOOL

# Latent Dirichlet Allocation Model

- Understanding whole sentences is hard
- **LDA** uses the "bag of words" idea to describe a simple model for document generation
- Suppose that we have *n* topics, *N* words
- Each document is represented by a vector $(a_1, a_2, \ldots, a_n)$ with $\Sigma a_i = 1$:
- A distribution of **documents across topics**

# Bag of Words

- Topic $t$ is described by a distribution **across words** $(b_{t1}, b_{t2}, \ldots, b_{tN})$
- Here's how you construct a bag of words representing a document:

1. Pick a topic from your distribution of topics; then
2. Pick a word from the distribution of words from that topic, and
3. Add that word to the bag

# Goal

- Goal: identify the parameters (including the number of topics)
  - There are lots, but we have lots of data
  - Algorithms (like the “EM”) exist
- Assign documents to topics: classification
- “Topic” is different from “content”
- No notion of sentiment here
- The assumption of interchangeable words seems to be not too costly

# General Text Mining Problem

- (1) Represent this free-form text as something we can analyze (a vector in a space)
- (2) Determine what patterns the set of somethings show, among text items
- Outputs might be distances (for MDS), clusters, in the form of association rules ("If <safety concern> then <unhappiness>: covers 20%, accuracy 70%") or something else

# One Author's View

| | | Finding Patterns | Finding Nuggets: New | Finding Nuggets: Not New |
|---|---|---|---|---|
| Non-text | | DM | ? | Database queries |
| Text | | Comp Ling | TM | Retrieval |

(Hearst, Proc. ACL '99)

# Medical Diagnostic Example

- Experts can’t read everything, especially outside their own fields
- Swanson (1988) used a text-mining-like approach (not all automated) to generate a new hypothesis about disease which, he says, was later found to be supported by evidence

# Swanson

- “Extracted evidence from titles of articles in the biomedical literature.”
- Apparently only titles used – this wouldn’t work in statistics!
- “Extraction” does not seem to have been automatic – but this was nearly 20 years ago

# Swanson Found That…

- Stress is associated with migraines
- Stress can lead to loss of magnesium
- Calcium channel blockers prevent some migraines
- Magnesium is a natural calcium channel blocker
- Spreading cortical depression (SCD) is implicated in some migraines
- High levels of magnesium inhibit SCD
- Migraine patients have high platelet aggregability
- Magnesium can suppress platelet aggregability

# …and concluded that…

- …magnesium deficiency might play a role in some migraines
- There does appear to be a relationship, although I'm not an expert here
- While not entirely automatic, the key point here is the (allegedly) **new** hypothesis, generated from text

# Sentiment Analysis

- Move beyond “bag of words” to try to extract positive or negative sentiment in text
  - Claim: humans only agree on this 80% of the time; if true, this is a hard problem!
- One way: use humans to characterize sentiment, then fit supervised learning models
  - Possibly by reading every element in the training set
  - Possibly by assigning sentiment to words or phrases or emoji

# Language Models

- Language Models have applications in:
  - Speech, handwriting, optical character recognition
  - Predictive text (think cellphone text app)
  - POS tagging and parsing
- These might be supervised/unsupervised
- Example: Form bags from spam and from non-spam messages. Now assume a new e-mail is like a random sample from one of the two bags. Which bag is it more likely to have come from?
  - Easy extension to multiple classes

NAVAL POSTGRADUATE SCHOOL

# Bayesian Spam Detection

- E.g. Half the e-mail I get is spam
  - Pr (S) = 0.5; Pr (~S) = 0.5;
- Imagine a set of messages pre-identified as spam or not
- Pr (“Rolex” | S) = 0.4; Pr (“Rolex” | ~S) = 0.01
- Pr (S | “Rolex”) =

$$\frac{\Pr(\text{“Rolex”} \mid S)\Pr(S)}{\Pr(\text{“Rolex”} \mid S)\Pr(S)+\Pr(\text{“Rolex”} \mid {\sim}S)\Pr({\sim}S)}$$

$$= 0.98.$$

- How do we combine multiple words?

# Bayesian Spam Detection

- E.g. Half the e-mail I get is spam
  - $\boxed{Pr(S) = 0.5;\ Pr(\sim S) = 0.5;}$
- Imagine a set of messages pre
  spam or not
- Pr (“Rolex” | S) = 0.4; Pr (“Role
- Pr (S | “Rolex”) =

Prior

$$\frac{Pr(\text{“Rolex”} \mid S)\, Pr(S)}{Pr(\text{“Rolex”} \mid S)\, Pr(S) + Pr(\text{“Rolex”} \mid \sim S)\, Pr(\sim S)}$$

$$= 0.98.$$

- How do we combine multiple words?

# Bayesian Spam Detection

- E.g. Half the e-mail I get is spam
  - Pr (S) = 0.5; Pr (~S) = 0.5;
- Imagine a set of messages pre-identified as spam or not
- Pr ("Rolex" | S) = 0.4; Pr ("Rolex" | ~S) = 0.01
- Pr (S | "Rolex") =

Pr ("Rolex" | S) P

Pr ("Rolex" | S) Pr (S)+Pr ("R S)

= 0.98.

- How do we combine multiple

Easy to estimate

# Bayesian Spam Detection

- E.g. Half the e-mail I get is spam
  - Pr (S) = 0.5; Pr (~S) = 0.5;
- Imagine a set of messages pr
  spam or not
- Pr (“Rolex” | S) = 0.4; Pr (“Rolex
- Pr (S | “Rolex”) =

Posterior

$$\frac{\text{Pr (“Rolex” | S) Pr (S)}}{\text{Pr (“Rolex” | S) Pr (S)+Pr (“Rolex” | ~S) Pr (~S)}}$$

= 0.98.

- How do we combine multiple words?

# Naïve Bayes Classifier

- This is an example of the **Naïve Bayes** Classifier
- Widely used beyond language analysis
- Suppose in some general classification problem we have $k$ classes $C_1, \ldots, C_k$
- Given a vector of (usually categoricals w/small numbers of levels) $\mathbf{x}$, we seek

$$\Pr\,(y_i = C_j \mid x_{i1}, x_{i2}, \ldots, x_{ip}) \text{ for each class } j$$

$$= \Pr\,(C_j) \Pr\,(\mathbf{x} \mid C_j) / \Pr\,(\mathbf{x}) \quad \text{[“the posterior”]}$$

$$\propto \Pr\,(\mathbf{x} \mid C_j) \Pr\,(C_j) = \text{the joint } \Pr\,(\mathbf{x}, C_j)$$

# Naïve Bayes Classifier (cont'd)

- Now $\Pr(\mathbf{x}, Cj) = \Pr(x_1 \mid x_2, \ldots, x_p, C_j)$

  $\times \Pr(x_2, x_3, \ldots, x_p, C_j)$

$= \Pr(x_1 \mid x_2, \ldots, x_p, C_j) \times \Pr(x_2 \mid x_3, \ldots, x_p, C_j)$

$\times \Pr(x_3, x_4, \ldots, x_p, C_j)$

$= \Pr(x_1 \mid x_2, \ldots, x_p, C_j) \times \Pr(x_2 \mid x_3, \ldots, x_p, C_j)$

$\times \ldots \times \Pr(x_{p-1} \mid x_p, C_j) \times \Pr(x_p \mid C_j) \Pr(Cj)$

$= \prod \Pr(x_i \mid x_{i+1}, \ldots, x_p, C_j)$ – call this $J$ for joint

- And **now** comes the naïve part…

# Naïve Bayes Classifier

- Let's assume that each $x_i$ is **conditionally independent** of the others, given $C$
  - That's "naïve" because…why should they be?
- Then J = Pr $(C_j) \times \prod$ Pr $(x_i \mid C_j)$, and the posterior is proportional to J
- It's easy to estimate Pr $(C_j)$ **["the prior"]** from data
- The other part is $p$ separate one-dimensional density estimations; if the x's are categorical, these are just frequencies in the table of $x_i \mid C_j$

# Naïve Bayes Considerations

- Other schemes exist for continuous x, but binning into discrete values is common
- Frequencies are often "smoothed" to accommodate rare combinations of $C$ and $x_i$ which would otherwise be given prob. 0
- Easy and fast
- Even when the Naïve Bayes estimates of Pr $(C_j)$ aren't very good numerically, the largest of these is often a good choice for classification
- "Naïve Bayes is to $p(C_j, \mathbf{x})$ as logit is to $p(C_j|\mathbf{x})$" say Ng and Jordan

# *n*-Gram Models

- An *n*-gram model tracks the frequencies of consecutive sets of *n* words
- Extends the "bag of words" (1-gram) model
- "Words" might be:
  - proteins in protein sequencing
  - sounds in speech recognition
  - {A, C, T, G} in DNA sequencing
  - letters in text, handwriting, image
  - words in text
- Word *n*-grams are straightforward to tabulate across a large corpus of text

# *n*-Gram Models

- If we tabulate all the 3-grams, say, in a corpus, then we can estimate the probability Pr (word | two preceding words)
  - And we can generate text: start with any two words, pick subsequent ones one at a time
- This is an order 2 **Markov model**
- Bigger $n \rightarrow$ better fidelity, except that bigger $n \rightarrow$ sparser data for estimating
- In its basic form, no mention of the position of the word inside a sentence

# Word 2 Vec

- This is a scheme due to Mikolov et al (2013) and apparently patented by Google
  – Open-source implementations are available
- Uses (e.g.) a neural network to embed words in vector spaces in a way that is more sophisticated than just the TDM
  – Extension: **sense embedding** differentiates between a single word with two distinct meanings, like “tank” or “general”

# Word 2 Vec, cont'd

- Under this scheme, vector **differences** between terms are preserved across the space
  - Though apparently there is no general agreement as to exactly why!
- So, e.g., France – Paris = Germany – Berlin
  - Which means the set of items close to (France – Paris + Berlin) includes, among other things, "Germany"

# Thoughts

- RSS feeds, streaming data, updated blogs, social media ever more common; we need schemes to digest and analyze large amounts of text
- Low-level tasks differ by language
- Clustering vs. classification (Unsupervised vs. supervised)
- Need to detect changes in time – evolution of new topics or terms, distributions of topics etc.

# Sites and Software

- Clustering: Yippy search engine (e.g., check out results for “Tiger”)
- GATE.ac.uk: the GATE project
- Lingpipe: Competitors, Demo
- Weka (open-source Java-based data mining project) has a text module
  - Companion MOA for streaming data
- Library `tm` in R
- Lots more