Text Analysis

Everything Data
CompSci 216 Spring 2019
Outline

• Basic Text Processing
  – Word Counts
  – Tokenization
    • Pointwise Mutual Information
  – Normalization
    • Stemming
Outline

• Basic Text Processing

• Finding Salient Tokens
  – TFIDF scoring
Outline

• Basic Text Processing
• Finding Salient Tokens

• Document Similarity & Keyword Search
  – Vector Space Model & Cosine Similarity
  – Inverted Indexes
Outline

• Basic Text Processing
  – Word Counts
  – Tokenization
    • Pointwise Mutual Information
  – Normalization
    • Stemming

• Finding Salient Tokens
• Document Similarity & Keyword Search
Basic Query: Word Counts

Google Flu

Emotional Trends

http://www.cos.neu.edu/home/amislove/twittermood/
Basic Query: Word Counts

• How many times does each word appear in the document?
Problem 1

• What is a word?
  – I’m … 1 word or 2 words {I’m} or {I, am}
  – State-of-the-art … 1 word or 4 words
  – San Francisco … 1 or 2 words

– Other Languages
  • French: l’ensemble
  • German: freundshaftsbezeichungen
  • Chinese: 這是一個簡單的句子 (no spaces)

This, one, easy, sentence
Solution: Tokenization

• For English:
  – Splitting the text on non alphanumerical characters is a reasonable way to find individual words.
  – But will split words like “San Francisco” and “state-of-the-art”

• For other languages:
  – Need more sophisticated algorithms for identifying words.
Finding simple phrases

• Want to find pairs of tokens that always occur together (co-occurrence)

• Intuition:
  Two tokens are co-occurent if they appear together more often than “random”
  – More often than monkeys typing English words

http://tmblr.co/ZiEAFywGEckV
Formalizing “more often than random”

• Language Model
  – Assigns a probability $P(x_1, x_2, ..., x_k)$ to any sequence of tokens.
  – More common sequences have a higher probability

  – Sequences of length 1: unigrams
  – Sequences of length 2: bigrams
  – Sequences of length 3: trigrams
  – Sequences of length $n$: $n$-grams
Formalizing "more often than random"

• Suppose we have a language model

  – \( P(\text{"San Francisco")}) \) is the probability that "San" and "Francisco" occur together (and in that order) in the language.
Formalizing “random”: Bag of words

• Suppose we only have access to the unigram language model
  – Think: all unigrams in the language thrown into a bag with counts proportional to their $P(x)$
  – Monkeys drawing words at random from the bag

  – $P(“San”) \times P(“Francisco”)$ is the probability that “San Francisco” occurs together in the (random) unigram model
Formalizing “more often than random”

- **Pointwise Mutual Information:**

\[
\text{PMI}(x_1, x_2) = \log_2 \frac{P(x_1, x_2)}{P(x_1) \cdot P(x_2)}
\]

- Positive PMI suggests word co-occurrence
- Negative PMI suggests words don’t appear together
What is $P(x)$?

- “Suppose we have a language model ...”

- Idea: Use counts from a large corpus of text to compute the probabilities
What is P(x)?

• **Unigram**: $P(x) = \frac{\text{count}(x)}{N}$
  - count(x) is the number of times token $x$ appears
  - N is the total # tokens.

• **Bigram**: $P(x_1, x_2) = \frac{\text{count}(x_1, x_2)}{N}$
  - count($x_1,x_2$) = # times sequence $(x_1,x_2)$ appears

• **Trigram**: $P(x_1, x_2, x_3) = \frac{\text{count}(x_1, x_2, x_3)}{N}$
  - count($x_1,x_2,x_3$) = # times sequence $(x_1,x_2,x_3)$ appears
What is $P(x)$?

• “Suppose we have a language model …”

• Idea: Use counts from a large corpus of text to compute the probabilities
Large text corpora

- [http://corpus.byu.edu/coca/](http://corpus.byu.edu/coca/)

The most widely used online corpora -- more than 130,000 distinct researchers, teachers, and students each month.

<table>
<thead>
<tr>
<th>English</th>
<th># words</th>
<th>language/dialect</th>
<th>time period</th>
<th>compare</th>
</tr>
</thead>
<tbody>
<tr>
<td>iWeb: The Intelligent Web-based Corpus</td>
<td>14 billion</td>
<td>US/CA/UK/IE/AU/NZ</td>
<td>2017</td>
<td>Info (中文)</td>
</tr>
<tr>
<td>News on the Web (NOW)</td>
<td>7.3 billion+</td>
<td>20 countries / Web</td>
<td>2010-last month</td>
<td></td>
</tr>
<tr>
<td>Global Web-Based English (GloWbE)</td>
<td>1.9 billion</td>
<td>20 countries / Web</td>
<td>2012-13</td>
<td></td>
</tr>
<tr>
<td>Wikipedia Corpus</td>
<td>1.9 billion</td>
<td>English</td>
<td>2014</td>
<td>Info</td>
</tr>
<tr>
<td>Hansard Corpus</td>
<td>1.6 billion</td>
<td>British (parliament)</td>
<td>1803-2005</td>
<td>Info</td>
</tr>
<tr>
<td>Early English Books Online</td>
<td>755 million</td>
<td>British</td>
<td>1470s-1690s</td>
<td></td>
</tr>
<tr>
<td>Corpus of Contemporary American English (COCA)</td>
<td>560 million</td>
<td>American</td>
<td>1990-2017</td>
<td>*** ***</td>
</tr>
<tr>
<td>Corpus of Historical American English (COHA)</td>
<td>400 million</td>
<td>American</td>
<td>1810-2009</td>
<td>**</td>
</tr>
<tr>
<td>The Movie Corpus [NEW]</td>
<td>200 million</td>
<td>US/CA/UK/IE/AU/NZ</td>
<td>1930-2018</td>
<td>Info</td>
</tr>
<tr>
<td>Corpus of US Supreme Court Opinions</td>
<td>130 million</td>
<td>American (law)</td>
<td>1790s-present</td>
<td></td>
</tr>
</tbody>
</table>

- Google N-gram viewer
  - [https://books.google.com/ngrams](https://books.google.com/ngrams)
Summary of Problem 1

• Word tokenization can be hard

• Space/non-alphanumeric words may oversplit the text

• We can find co-occurrent tokens:
  – Build a language model from a large corpus
  – Check whether the pair of tokens appear more often than random using pointwise mutual information.
Language models …

• … have many many applications

– Tokenizing long strings
– Word/query completion/suggestion
– Spell checking
– Machine translation (**NMT**)
– Question answering (**SQUAD**)
– Natural Language inference (**MultiNLI**)
– Conversational Question Answering (**CoQA**)
– …
BRACE YOURSELF

DEEP LEARNING IS COMING
Better Language Models and Their Implications

We've trained a large-scale unsupervised language model which generates coherent paragraphs of text, achieves state-of-the-art performance on many language modeling benchmarks, and performs rudimentary reading comprehension, machine translation, question answering, and summarization — all without task-specific training.
BERT (Google)

OpenAI GPT

ELMo

Problem 2

- A word may be represented in many forms
  - car, cars, car’s, cars’ \(\rightarrow\) car
  - automation, automatic \(\rightarrow\) automate

- Lemmatization: Problem of finding the correct dictionary headword form
Solution: Stemming

• Words are made up of
  – Stems: core word
  – Affixes: modifiers added (often with grammatical function)

• Stemming: reduce terms to their stems by crudely chopping off affixes
  – automation, automatic \(\rightarrow\) automat
Porter’s algorithm for English

- Sequences of rules applied to words

Example rule sets:

\/(.*).sses$/ → \1ss
\/(.*).ies$/ → \1i
\/(.*).ss$/ → \1s
\/(.*[^s])s$/ → \1

\/(.*[aeiou]+.*)ing$/ → \1
\/(.*[aeiou]+.*)ed$/ → \1
Any other problems?

• Same words that mean different things
  – Florence the person vs Florence the city
  – Paris Hilton (person or hotel)

• Abbreviations
  – I.B.M vs International Business Machines

• Different words meaning same thing
  – Big Apple vs New York

• …
Any other problems?

• Same words that mean different things
  – *Word Sense Disambiguation*

• Abbreviations
  – *Translations*

• Different words meaning same thing
  – *Named Entity Recognition & Entity Resolution*

• …
Word embeddings

Word2Vec

GloVe

Can convert words to vectors of numbers - at the hearth of most NLP applications with deep learning

http://slides.com/simonescardapane/the-dark-side-of-deep-learning
Outline

• Basic Text Processing

• Finding Salient Tokens
  – TFIDF scoring

• Document Similarity & Keyword Search
Summarizing text

Belinelli's late jumper gives Popovich his 1000th career w.

Yahoo Sports (blog) - 4 hours ago

Gregg Popovich of the San Antonio Spurs has already established himself ... Nevertheless, it's pretty cool and rare any time a coach hits 1,000 ...
Finding salient tokens (words)

• Most frequent tokens?
Document Frequency

• Intuition: Uninformative words appear in many documents (not just the one we are concerned about)

• Salient word:
  – High count within the document
  – Low count across documents
TF•IDF score

• Term Frequency (TF):

\[ TF(x) = \log_{10}(1 + c(x)) \quad \text{or} \quad c(x) \]

\[ c(x) \text{ is \# times } x \text{ appears in the document} \]

• Inverse Document Frequency (IDF):

\[ IDF(x) = \log_{10} \left( \frac{N_{docs}}{DF(x)} \right) \]

DF(x) is the number of documents x appears in.
Back to summarization

- Simple heuristic:
  - Pick sentences $S = \{x_1, x_2, \ldots, x_k\}$ with the highest:

$$\text{Salience}(S) = \frac{1}{|S|} \sum_{x \in S} \text{TF}(x) \cdot \text{IDF}(x)$$
Outline

• Basic Text Processing
• Finding Salient Tokens

• Document Similarity & Keyword Search
  – Vector Space Model & Cosine Similarity
  – Inverted Indexes
Document Similarity

Belinelli's late jumper gives **Popovich** his **1000th** career W...

Yahoo Sports (blog) - 4 hours ago

Gregg **Popovich** of the San Antonio Spurs has already established himself ... Nevertheless, it's pretty cool and rare any time a coach hits **1,000** ...

Spurs' Gregg **Popovich** becomes 9th NBA coach to win **1000** games

Sl.com - 20 hours ago

**SVG: Popovich's 1000** wins 'a great accomplishment'

Detroit Free Press - 2 minutes ago

**Gregg Popovich Wins 1000th** Game with Milestones Ahead & Other ...

In-Depth - Bleacher Report - 18 hours ago

**Six things to know about Gregg Popovich's 1000th win**

Blog - Washington Post (blog) - 20 hours ago

**Raptors Beat Spurs, Deny Popovich 1000th Win**

In-Depth - ABC News - Feb 8, 2015
Vector Space Model

• Let $V = \{x_1, x_2, \ldots, x_n\}$ be the set of all tokens (across all documents)

• A document is a $n$-dimensional vector

$$D = [w_1, w_2, \ldots, w_n]$$

where $w_i = \text{TFIDF}(x_i, D)$
Distance between documents

- Euclidean distance
  \[ D1 = [w_1, w_2, \ldots, w_n] \]
  \[ D2 = [y_1, y_2, \ldots, y_n] \]
  \[ d(D1, D2) = \sqrt{\sum_i (w_i - y_i)^2} \]

- Why?
Cosine Similarity

\[ d(D1, D2) = \cos(\theta) \]
Cosine Similarity

• $D_1 = [w_1, w_2, \ldots, w_n]$
• $D_2 = [y_1, y_2, \ldots, y_n]$

$$d(D_1, D_2) = \frac{\sum_i w_i \cdot y_i}{\sqrt{\sum_i w_i^2} \sqrt{\sum_i y_i^2}}$$
Keyword Search

• How to find documents that are similar to a keyword query?

• Intuition: Think of the query as another (very short) document
Keyword Search

• Simple Algorithm

For every document $D$ in the corpus
Compute $d(q, D)$

Return the top-$k$ highest scoring documents
Does it scale?

http://www.worldwidewebsize.com/
Inverted Indexes

• Let $V = \{x_1, x_2, \ldots, x_n\}$ be the set of all token

• For each token, store
  $<\text{token}, \text{sorted list of documents token appears in}>$
  – $<"\text{caeser"}, [1,3,4,6,7,10,\ldots]>$

• How does this help?
Using Inverted Lists

• Documents containing “caesar”
  – Use the inverted list to find documents containing “caesar”

  – What additional information should we keep to compute similarity between the query and documents?
Using Inverted Lists

• Documents containing “caesar” AND “cleopatra”
  – Return documents in the intersection of the two inverted lists.
  – Why is inverted list sorted on document id?

• OR? NOT?
  – Union and difference, resp.
Many other things in a modern search engine …

• Maintain positional information to answer phrase queries

• Scoring is not only based on token similarity
  – Importance of Webpages: PageRank (in later classes)

• User Feedback
  – Clicks and query reformulations
Summary

• Word counts are very useful when analyzing text
  – Need good algorithms for tokenization, stemming and other normalizations

• Algorithm for finding word co-occurrence
  – Language Models
  – Pointwise Mutual Information
Summary (contd.)

• Raw counts are not sufficient to find salient tokens in a document
  – Term Frequency x Inverse Document Frequency (TFIDF) scoring

• Keyword Search
  – Use Cosine Similarity over TFIDF scores to compute similarity
  – Use Inverted Indexes to speed up processing.