Announcements

• IMPORTANT: new room, starting next Wednesday, DBH 1422

• Monday: no class or office hours (university holiday)

• Tuesday: office hours from 9:30 to 11 (slightly extended)

• Assignment 1 due Wednesday, in Dropbox on EEE
Class Organization

• Class Website:  www.ics.uci.edu/~smyth/courses/cs175
  – This is where to find assignments, links to software, project guidelines, etc

• Piazza Website:
  – https://piazza.com/uci/winter2016/compsci175/home
  – Use this to post questions related to assignments, projects, etc
  – Piazza is where we will post announcements, answers to questions, etc

• Office Hours
  – Monday 9:30 to 10:30, Tuesday 10 to 11 (except next week)
Today’s Lecture

• Projects
  – Examples and ideas

• Text preprocessing
  – Tokenization, stopword removal, part of speech tagging, ...

• Python examples and Assignment 1
Examples of Possible Projects
Language Technology

making good progress

mostly solved

Sentiment analysis
- Best roast chicken in San Francisco!
- The waiter ignored us for 20 minutes.

Coreference resolution
- Carter told Mubarak he shouldn’t run again.

Word sense disambiguation (WSD)
- I need new batteries for my mouse.

Parsing
- I can see Alcatraz from the window!

Machine translation (MT)
- The 13th Shanghai International Film Festival...
- The Dow Jones is up
- The S&P500 jumped
- Housing prices rose

Information extraction (IE)
- You’re invited to our dinner party, Friday May 27 at 8:30

still really hard

Question answering (QA)
- Q. How effective is ibuprofen in reducing fever in patients with acute febrile illness?

Paraphrase
- XYZ acquired ABC yesterday
- ABC has been taken over by XYZ

Summarization
- The Dow Jones is up
- The S&P500 jumped
- Housing prices rose
- Economy is good

Dialog
- Where is Citizen Kane playing in SF?
- Castro Theatre at 7:30. Do you want a ticket?

Spam detection
- Let’s go to Agra!
- Buy V1AGRA...

Part-of-speech (POS) tagging
- ADJ ADJ NOUN VERB ADV
- Colorless green ideas sleep furiously.

Named entity recognition (NER)
- PERSON ORG LOC
- Einstein met with UN officials in Princeton
Example 1: Information Extraction about Organizations

• Problem statement:
  – Input: the name of an organization, e.g., Apple Inc, University of California
  – Output: structured record of facts about the organization, e.g.,
    • Location: where is the organization is located
    • Employees: how many people work for the organization
    • Management: who the head of the organization
    • Age: when was the organization founded?
    • And so on and so on...

• Approach:
  – Take a large set of documents where the organization is mentioned
    • E.g., Web pages from a search on the organization’s name, news articles, etc
  – Use information extraction algorithms to extract the relevant information
    (can’t use sources such as Google or Wikipedia: problem is too easy)
Example 1: Information Extraction about Organizations

• Challenges
  – Ambiguity in names:
    • need to recognize Apple as the name of the US company, not a person’s name or the fruit, etc
  – Need to extract relations between the entity and the property (location, age) of the entity
    • Relations can be stated in many different ways, e.g.,
      – “Tim Cook is CEO of Apple”
      – “Cook is the head of Apple”
      – “Apple’s leader, Mr. Timothy Cook, said…”
  – Information may be contradictory
    • Out of date: “Steve Jobs, CEO of Apple,…”
    • Incorrect: “Mark Zuckerberg, who heads Apple,…”
  – Need to build a web-crawler to find relevant documents
    • Using the results from an existing search engine would be ideal
    • Building a system that can do this in (near) real-time would be challenging
Apple Inc.
Company

Apple is an American multinational technology company headquartered in Cupertino, California, that designs, develops, and sells consumer electronics, computer software, and online services. Its hardware products include the i... +

Stock price: AAPL (NASDAQ) 118.90 ▼ -0.20 (-0.17%)
Jan 11, 1:38 PM EST · 20 min delayed

Customer service: +1 800-692-7753

Representative: Chat online with a representative

Founded: Apr 01, 1976 · Cupertino, CA

CEO: Tim Cook (2011)

Founders: Steve Jobs · Steve Wozniak · Ronald Wayne
Example 2: Extracting Relevant Aspects of Reviews

• Problem statement:
  – Input: large set of reviews (e.g., Yelp, Amazon, Rotten Tomatoes, Trip Advisor) of items (restaurants, products, movies, hotels)
  – Output: for each product or item
    • For each important aspects of an item, a summary of
      – A summary of the text that typically surrounds the item
      – A distribution of how positive or negative the text for that aspect

• Example:
  – Italian restaurant “Roma Garden”:
  – Aspects = food, service, parking, wait-time
  – Text summary: “food is excellent”, “service is slow”, “parking is a problem”
  – Emotion: for each aspect report the proportion of reviews that have negative, neutral, or positive reviews associated with each item
Example 2: Extracting Relevant Aspects of Reviews

• Possible Approach:
  – Take a large set of publicly-available reviews for a particular type of item or product (e.g., Yelp + restaurants)
    • Or build a Web crawler to download this information from a Website
  – Identify the different items (restaurants, products, etc)
    • Usually available in the metadata
  – Define the “aspects” for these types of items
  – For each item: for sentences within each review perform, for each aspect
    • information extraction to detect sentences mentioning the aspect
    • use an emotion classifier to classify the sentence as positive, neutral, or negative
    • use text summarization to summarize the positive and the negative sentences

• Challenges
  – How are the “aspects” defined? (Manually? Automatically? By the user?)
  – How will you evaluate the results in terms of quality/accuracy
  – This is complex: instead could do text summarization of a set of reviews
Example 3: Document Classification.....Many Possible Projects

Preprocessing Methods
- Bag of Words
- Stemming
- Ngrams/Phrases
- Word embedding

Classification Models
- Naïve Bayes
- K-nearest neighbor
- Logistic regression
- Neural network
- Deep neural network

X
Example 3: Document Classification.....Many Possible Projects

Preprocessing Methods
- Bag of Words
- Stemming
- Ngrams/Phrases
- Word embedding

Classification Models
- Naïve Bayes
- K-nearest neighbor
- Logistic regression
- Neural network
- Deep neural network

Classification Problems
- Binary Classification
- Multiclass Classification
- Multi-label, Multiclass Ranking
Example 3: Document Classification.....Many Possible Projects

- **Preprocessing Methods**
  - Bag of Words
  - Stemming
  - Ngrams/Phrases
  - Word embedding

- **Classification Models**
  - Naïve Bayes
  - K-nearest neighbor
  - Logistic regression
  - Neural network
  - Deep neural network

- **Classification Problems**
  - Binary Classification
  - Multiclass Classification
  - Multi-label, Multiclass Ranking

- **Training Algorithms**
  - Gradient Descent
  - Stochastic Gradient
  - Regularization

- **Application Areas**
  - Sentiment Classification
  - News Article Classification
  - Wikipedia Page Classification
  - ....and more
Examples of Past CS 175 Student Projects

<table>
<thead>
<tr>
<th>Description</th>
<th>Data Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentiment analysis</td>
<td>Twitter (text + sentiment labels)</td>
</tr>
<tr>
<td>Star/score prediction from text</td>
<td>Yelp, Movie Reviews: text + scores</td>
</tr>
<tr>
<td>Predict number of upvotes for a Reddit post</td>
<td>Reddit posts + votes + timestamps</td>
</tr>
<tr>
<td>Predict if a restaurant will close in the next month</td>
<td>Yelp reviews (text, timestamps, metadata)</td>
</tr>
<tr>
<td>Simulate realistic text from an author/speaker/character</td>
<td>Gutenberg books, tweets, movie scripts</td>
</tr>
<tr>
<td>Automated poetry or song lyrics generation</td>
<td>Text from song lyrics or poetry</td>
</tr>
<tr>
<td>Automated essay grading</td>
<td>Text for student essays with human scores</td>
</tr>
</tbody>
</table>
Important Components of Projects

• Clear definition of the problem: inputs, outputs

• Data
  – Make sure you will be able data you need, e.g., labeled data for classifications

• Self-written components
  – Which parts of the code will you write and what will be existing code?

• Evaluation
  – How will you evaluate the quality of your system?
  – Think of ways to compare version A versus B

• Run-time
  – Do you want a system/demo that can run in real-time, or one that operates off-line? Different design decisions for each.
Project Tips: Plan in Stages

Plan your project in stages so that the overall project is not dependent on the riskier elements working.

Example:

PHASE 1

- Original Documents
- Standard Bag of Words
- Standard Logistic Regression
- Cross-Validation Experiments
Project Tips: Plan in Stages

Plan your project in stages so that the overall project is not dependent on the riskier elements working.

Example:

**PHASE 1**
- Original Documents
- Standard Bag of Words
- Bag of Phrases (ngrams)
- Standard Logistic Regression
- Cross-Validation Experiments

**PHASE 2**
Project Tips: Plan in Stages

Plan your project in stages so that the overall project is not overly dependent on the riskier elements.

Example:

PHASE 1
- Original Documents

PHASE 2
- Bag of Words

PHASE 3
- Standard Logisitic Regression
- Cross-Validation Experiments
- Deep Neural Network
Basic Concepts in Text Representation

(Thanks to Prof Mark Steyvers, UCI, for some of the slides in this section)
Basic Concepts and Terminology

• Tokens:
  – Groupings of characters in the raw text
  – individual words (or “word types”) + possibly numbers, punctuation, etc

• Word Types or Terms
  – Unique tokens (or combinations of tokens) that are meaningful
  – e.g., words such as “cat”, “dog”, terms such as “U.S” or “92697”
  – Can also include bigrams (“San Diego”), trigrams (“New York City”), etc

• Vocabulary
  – The set of terms we use in a specific application
  – The English language has order of 1 million words
  – In a particular text analysis application we might use a vocabulary of only 20,000 “relevant/common” words (unigrams) (plus some ngrams)
How many words?

\( N \) = number of tokens

\( V \) = vocabulary = set of types

\(|V|\) is the size of the vocabulary

Church and Gale (1990): \(|V| > O(N^{1/2})\)

|                          | Tokens = \( N \) | Types = \(|V|\) |
|--------------------------|------------------|-----------------|
| Switchboard phone       | 2.4 million      | 20 thousand     |
| conversations           |                  |                 |
| Shakespeare             | 884,000          | 31 thousand     |
| Google N-grams          | 1 trillion       | 13 million      |

From https://web.stanford.edu/~jurafsky/slp3/

Speech and Language Processing, 3rd ed, Jurafsky and Martin
Frequency of Word in English usage

Graph from www.prismnet.com/~dierdorf/wordfrequency.png

Very long “tail” of rare words
Pipelines in Text Processing

- Pipelines, consisting of different analysis steps, are widely used in text analysis applications

- Example pipeline
  - Raw text as a string -> tokenizer -> original sequence of tokens
  - Original sequence of tokens -> stopword removal -> sequence of tokens
  - Sequence of tokens -> part of speech tagger -> part of speech tags for tokens
  - Sequence of tokens -> vocabulary definer
    -> vocabulary list and counts (bag of words)
  - Bag of words -> machine learning algorithm -> document labels

- Note that the steps in the pipeline, and how these are implemented (e.g., how tokenization is done) will vary from application to application depending on the problem
  - There is no single “correct” way....as we will see in our examples
Example of a Pipeline for Document Classification

Original document

... [ (NBA, 7), (Lakers, 3), basket (2)....] 

Document features (e.g., a “bag of words”)

Label = basketball
Class label

Padhraic Smyth, UC Irvine: CS 175, Winter 2017
Example of a Pipeline for Document Classification

Original document

Tokenization, Stemming, Part of Speech Tagging, etc

[ (NBA, 7), (Lakers, 3), basket (2), … ]

Document features (e.g., a “bag of words”)

Label = basketball

Class label

Classification model, e.g.,
- naïve Bayes
- logistic regression
- neural network

(Assignment 2 and next week’s lectures)
Tokenization

- Split up text into individual tokens (words/terms)
- Simplest approach is to ignore all punctuation and white space and use only unbroken strings of alphabetic characters as tokens

If you had a magic potion I’d love to have it.

If you had a magic potion I’d love to have it.
Issues in Tokenization

- Finland’s capital → Finland Finlands Finland’s ?
- what’re, I’m, isn’t → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??

From https://web.stanford.edu/~jurafsky/slp3/

Speech and Language Processing, 3rd ed, Jurafsky and Martin
Issues in Tokenization

• Other languages:
  
  – Accents: résumé vs. resume.

  – L'ensemble → one token or two? L ? L’ ? Le ?
Sentence Tokenization: Example using a Decision Tree

From https://web.stanford.edu/~jurafsky/slp3/  
Speech and Language Processing, 3rd ed, Jurafsky and Martin
Frequency of Word in English usage

Rank of Word by Frequency

Graph from www.prismnet.com/~dierdorf/wordfrequency.png

Very long “tail” of rare words
Stopword Removal

- Remove words that are likely to be irrelevant to our problem
- Keep content words (typically verbs, adverbs, nouns, adjectives)

Example:

If you had a magic potion I’d love to have it. If that makes sense

But what about this?

[Prince Hamlet] To be or not to be ...
NLTK StopWord List

```python
>>> from nltk.corpus import stopwords
>>> stopwords.words('english')
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']
```

Note: in many applications there may be additional domain-specific “stop words” that are very common and that we may want to remove since they contain little information, e.g., “restaurant” in reviews.
Example: Defining a Vocabulary

Raw text (a string in Python)
\[
\text{raw1} = \text{“The dog chased the cat and the mouse. Why did the dog do this?”}
\]

There are 14 word tokens in the string raw1 (if we ignore punctuation and spaces)
The, dog, chased, the, cat, and, the, mouse, Why, did, the, dog, do, this

The vocabulary (the unique tokens, normalizing to lower case) is:
the, dog, chased, cat, and, mouse, why, did, do, this
The vocabulary size is 10.

The counts for a bag of words representation is:
the (3), dog (2), chased (1), cat (1), and (1), mouse (1), why (1), did (1), do (1), this (1)

If we remove stopwords we decrease our vocabulary size to 4
dog (2), chased (1), cat (1), mouse (1)
NLTK Tokenizer

```python
from nltk.tokenize import word_tokenize

>>> s = """Good muffins cost $3.88\n in New York.
 Please buy me ... two of them.\n \n Thanks.""

>>> word_tokenize(s)
['Good', 'muffins', 'cost', '$', '3.88', 'in', 'New', 'York', '.
', 'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.']
```

Note that this tokenizer retains lower and upper case tokens and also returns punctuation as tokens
Doing the Example in NLTK

(see Section 4.1 in Chapter 1 of NLTK Book)

```python
>> raw1 = "The dog chased the cat and the mouse. Why did the dog do this?"
>> type(raw1)
str
>> len(raw1)
63
>>
>> from nltk import word_tokenize
>>
>> tokens = word_tokenize(raw1)
>>
>> type(tokens)
List
>>
>> len(tokens)
16
```
Doing the Example in NLTK (continued)

(see Section 4.1 in Chapter 1 of NLTK Book)

```python
>> vocab1 = set(tokens)
>> len(vocab1)
16
>>
>> vocab2 = set(word.lower() for word in tokens)
>> len(vocab2)
12
>>
>> vocab3 = set(word.lower() for word in tokens1 if word.isalpha())
>> len(vocab3)
10
```
Defining the Vocabulary

- Vocabulary
  - Set of terms (words) used to construct the document-term matrix

- Basic approach: use single words (unigrams) as terms

- Remove very common terms (e.g., stop words)

- Remove very rare terms: e.g., remove all terms that occur in fewer than K documents in the corpus (e.g., K = 10)
  - Gets rid of misspellings, unusual names, etc

- Can extend term list with n-grams
  - Frequent word combinations (2-grams, 3-grams,...)
    "feel good" / "New York City"
## Example of Bag-of-Words Matrix

<table>
<thead>
<tr>
<th></th>
<th>database</th>
<th>SQL</th>
<th>index</th>
<th>calculus</th>
<th>derivative</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>24</td>
<td>21</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>d2</td>
<td>32</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>d3</td>
<td>12</td>
<td>16</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d4</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d5</td>
<td>43</td>
<td>31</td>
<td>20</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>d6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>d7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>d8</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>d9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>34</td>
<td>27</td>
<td>25</td>
</tr>
<tr>
<td>d10</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>4</td>
<td>23</td>
</tr>
</tbody>
</table>
## Another Example of Bag of Words

<table>
<thead>
<tr>
<th></th>
<th>water</th>
<th>farming</th>
<th>cattle</th>
<th>agriculture</th>
<th>land rights</th>
<th>use</th>
<th>stock</th>
<th>portfolio</th>
<th>bonds</th>
<th>return</th>
<th>loan</th>
<th>interest rate</th>
<th>profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc2</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>doc10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
## Learning a Document Classifier

Use labeled training data (supervised learning) to learn a classifier that maps a feature vector to a label (see lectures in Week 2)

<table>
<thead>
<tr>
<th></th>
<th>water</th>
<th>farming</th>
<th>cattle</th>
<th>agriculture</th>
<th>land rights</th>
<th>use</th>
<th>stock</th>
<th>portfolio</th>
<th>bonds</th>
<th>return</th>
<th>loan</th>
<th>interest rate</th>
<th>profit</th>
<th>Class Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>doc2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>doc3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>doc5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>doc6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>doc7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>doc8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>doc9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>doc10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
Part of Speech (POS) Tagging

- Common POS categories (or tags) in English:
  - Noun, verb, article, preposition, pronoun, adverb, conjunction, interjection

- However there are many more specialized categories
  - E.g., proper nouns: e.g., ‘Toronto’, ‘Smith’,....
  - E.g., comparative adverb: e.g., ‘bigger’, ‘smaller’,...
  - E.g., symbol: ‘3.12’, ‘$’,...

- Assigning POS categories to words in text is known as **tagging**
Universal Tagset (as used in NLTK)

12 universal tags:
VERB - verbs (all tenses and modes)
NOUN - nouns (common and proper)
PRON - pronouns
ADJ - adjectives
ADV - adverbs
ADP - adpositions (prepositions and postpositions)
CONJ - conjunctions
DET - determiners
NUM - cardinal numbers
PRT - particles or other function words
X - other: foreign words, typos, abbreviations
. - punctuation

See "A Universal Part-of-Speech Tagset" by Slav Petrov, Dipanjan Das and Ryan McDonald for more details: http://arxiv.org/abs/1104.2086
and http://code.google.com/p/universal-pos-tags/
# Examples

## Universal Part-of-Speech Tagset

<table>
<thead>
<tr>
<th>Tag</th>
<th>Meaning</th>
<th>English Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>adjective</td>
<td>new, good, high, special, big, local</td>
</tr>
<tr>
<td>ADP</td>
<td>adposition</td>
<td>on, of, at, with, by, into, under</td>
</tr>
<tr>
<td>ADV</td>
<td>adverb</td>
<td>really, already, still, early, now</td>
</tr>
<tr>
<td>CONJ</td>
<td>conjunction</td>
<td>and, or, but, if, while, although</td>
</tr>
<tr>
<td>DET</td>
<td>determiner, article</td>
<td>the, a, some, most, every, no, which</td>
</tr>
<tr>
<td>NOUN</td>
<td>noun</td>
<td>year, home, costs, time, Africa</td>
</tr>
<tr>
<td>NUM</td>
<td>numeral</td>
<td>twenty-four, fourth, 1991, 14:24</td>
</tr>
<tr>
<td>PRT</td>
<td>particle</td>
<td>at, on, out, over, per, that, up, with</td>
</tr>
<tr>
<td>PRON</td>
<td>pronoun</td>
<td>he, their, her, its, my, I, us</td>
</tr>
<tr>
<td>VERB</td>
<td>verb</td>
<td>is, say, told, given, playing, would</td>
</tr>
<tr>
<td>.</td>
<td>punctuation marks</td>
<td>. , ; !</td>
</tr>
<tr>
<td>X</td>
<td>other</td>
<td>ersatz, esprit, dunno, gr8, univeristy</td>
</tr>
</tbody>
</table>

(from Section 2.3 in Chapter 5 of NLTK Book)
POS Tagging Algorithms

• Tagging Algorithms: “POS Taggers”
  – Tagging is often done automatically with algorithms
  – These algorithms often use sequential (Markov) models
    • The tag for a particular token can depend on words/tags before and after it
  – These models are trained using machine learning
    • using various word features and dictionaries as input
  – Trained on manually labeled documents

• Tagging performance is best on material that the tagger was originally trained on (often news documents)

• Tags can be helpful “downstream” for various applications
  – E.g., for document classification we might want to only use nouns, adjectives, and verbs and ignore everything else
  – E.g., for information extraction we might focus only on nouns
Challenges in POS Tagging

• Tagging words in text with their correct POS tags is not simply assigning words to tags using a lookup table

• Semantic context
  – *The negotiator was able to bridge the gap between the 2 sides*
  – Here ‘bridge’ is used as a verb even though we ordinarily think of it as a noun
  – The other words in the sentence and the grammatical structure allow us to interpret ‘bridge’ here as a verb

• Ambiguity, e.g.,
  – *The president was entertaining last night*
    • Both the adjective and verb tag for “entertaining” work here, i.e., there is ambiguity

• Tokenization issues
  – The algorithm must be able to deal with tokens such as I’ld or ‘pre-specified’
Software for Part of Speech Tagging

- Many software packages and online tools available

- NLTK POS Tagger

- Stanford natural language group provides excellent POS taggers in several languages (English, German, Chinese, French, etc)
  - Uses the Penn Treebank tagset

- Online demo
  - [http://demo.ark.cs.cmu.edu/parse](http://demo.ark.cs.cmu.edu/parse)
Examples in Python and Assignment 1
Announcements

- **IMPORTANT:** new room, starting next Wednesday, DBH 1422

- Monday: no class or office hours (university holiday)

- Tuesday: office hours from 9:30 to 11 (slightly extended)

- Assignment 1 due Wednesday, in Dropbox on EEE
Assignment 1

Available on the class Web page

Due Wednesday Jan 18th by noon (to dropbox on EEE)

Outline

– Read Sections of Chapter 1 and 3 of the online NLTK book
– Install Anaconda/NLTK/…
– Write simple functions in Python for text analysis
  • Compute percentage of alphabetic characters in a string
  • Detect the first K words on a Web page
  • Parse text into parts of speech (nouns, verbs, etc)
– Submit your code as a single python file via EEE
Examples of Analyzing Texts in NLTK and Python
Analyzing Different Texts in NLTK
(from Section 1 in Chapter 2 of NLTK Book)

Python input:

```
import nltk
from nltk.corpus import gutenberg

gutenberg.fileids()
```

Python output:

```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```
def define_vocabulary(raw_text, textname):
    # function to extract a vocabulary from raw text in different ways
    print('Analyzing raw text and extracting vocabularies in', textname)
    print('The number of characters in the raw text is', len(raw_text))

    # now tokenize the text....
    tokens = word_tokenize(raw_text)
    print('The number of tokens is', len(tokens))

    # define a vocabulary as the number of unique tokens
    vocab1 = set(tokens)
    print('Initial vocabulary size is', len(vocab1))

    # normalizing to lower case
    vocab2 = set(word.lower() for word in tokens)
    print('Vocabulary size after normalizing to lower case is', len(vocab2))

    # removing punctuation
    vocab3 = set(word.lower() for word in tokens if word.isalpha())
    print('Vocabulary size after removing punctuation is', len(vocab3))
    print('\n')
Invoking the vocabulary function...

# import various texts and list them...
from nltk.corpus import gutenberg
gutenberg.fileids()

# import the tokenize function
from nltk import word_tokenize

# test the vocabulary function with some of the large raw texts

textname1 = 'bible-kjv.txt'
raw1 = gutenberg.raw(textname1)
define_vocabulary(raw1, textname1)

textname2 = 'melville-moby_dick.txt'
raw2 = gutenberg.raw(textname2)
define_vocabulary(raw2, textname2)

textname3 = 'shakespeare-hamlet.txt'
raw3 = gutenberg.raw(textname3)
define_vocabulary(raw3, textname3)
Analyzing raw text and extracting vocabularies in bible-kjv.txt
The number of characters in the raw text is 4332554
The number of tokens is 946812
Initial vocabulary size is 18192
Vocabulary size after normalizing to lower case is 17188
Vocabulary size after removing punctuation is 12560

Analyzing raw text and extracting vocabularies in melville-moby_dick.txt
The number of characters in the raw text is 1242990
The number of tokens is 254989
Initial vocabulary size is 20755
Vocabulary size after normalizing to lower case is 18717
Vocabulary size after removing punctuation is 16509

Analyzing raw text and extracting vocabularies in shakespeare-hamlet.txt
The number of characters in the raw text is 162881
The number of tokens is 36326
Initial vocabulary size is 5540
Vocabulary size after normalizing to lower case is 4812
Vocabulary size after removing punctuation is 4560
Python code for computing various statistics related to the texts

```python
# loop over each text (each fileid)...
for fileid in gutenberg.fileids():
    # compute basic statistics
    num_chars = len(gutenberg.raw(fileid))
    num_word_tokens = len(gutenberg.words(fileid))
    num_sents = len(gutenberg.sents(fileid))
    vocab_size = len(set(w.lower() for w in gutenberg.words(fileid)))

    # compute derived statistics of interest
    average_word_length = round(num_chars/num_word_tokens)
    average_sentence_length = round(num_word_tokens/num_sents)
    lexical_diversity = round(num_word_tokens/vocab_size)

    # print the results for each fileid
    print(average_word_length, average_sentence_length, lexical_diversity, vocab_size, fileid)
```
<table>
<thead>
<tr>
<th>Word Count</th>
<th>Number of Words</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>26</td>
<td>austen-emma.txt</td>
</tr>
<tr>
<td>26</td>
<td>17</td>
<td>austen-persuasion.txt</td>
</tr>
<tr>
<td>28</td>
<td>22</td>
<td>austen-sense.txt</td>
</tr>
<tr>
<td>34</td>
<td>79</td>
<td>bible-kjv.txt</td>
</tr>
<tr>
<td>19</td>
<td>5</td>
<td>blake-poems.txt</td>
</tr>
<tr>
<td>19</td>
<td>14</td>
<td>bryant-stories.txt</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>burgess-busterbrown.txt</td>
</tr>
<tr>
<td>20</td>
<td>13</td>
<td>carroll-alice.txt</td>
</tr>
<tr>
<td>20</td>
<td>12</td>
<td>chesterton-ball.txt</td>
</tr>
<tr>
<td>23</td>
<td>11</td>
<td>chesterton-brown.txt</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>chesterton-thursday.txt</td>
</tr>
<tr>
<td>21</td>
<td>25</td>
<td>edgeworth-parents.txt</td>
</tr>
<tr>
<td>26</td>
<td>15</td>
<td>melville-moby_dick.txt</td>
</tr>
<tr>
<td>52</td>
<td>11</td>
<td>milton-paradise.txt</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>shakespeare-caesar.txt</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>shakespeare-hamlet.txt</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>shakespeare-macbeth.txt</td>
</tr>
<tr>
<td>36</td>
<td>12</td>
<td>whitman-leaves.txt</td>
</tr>
</tbody>
</table>
## Average sentence length

<table>
<thead>
<tr>
<th>Length</th>
<th>Count</th>
<th>Sentence Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>25</td>
<td>26 7344</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>17 5835</td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td>22 6403</td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>79 12767</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>5 1535</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>14 3940</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>12 1559</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>13 2636</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>12 8335</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>11 7794</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>11 6349</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>25 8447</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>15 17231</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>11 9021</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>9 3032</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>8 4716</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>7 3464</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>12 12452</td>
</tr>
</tbody>
</table>
Lexical diversity

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>25</td>
<td>26</td>
<td>7344</td>
<td>austen-emma.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>17</td>
<td>5835</td>
<td>austen-persuasion.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td>22</td>
<td>6403</td>
<td>austen-sense.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>79</td>
<td>12767</td>
<td>bible-kjv.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>5</td>
<td>1535</td>
<td>blake-poems.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>14</td>
<td>3940</td>
<td>bryant-stories.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>12</td>
<td>1559</td>
<td>burgess-busterbrown.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>13</td>
<td>2636</td>
<td>carroll-alice.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>12</td>
<td>8335</td>
<td>chesterton-ball.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>11</td>
<td>7794</td>
<td>chesterton-brown.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>11</td>
<td>6349</td>
<td>chesterton-thursday.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>25</td>
<td>8447</td>
<td>edgeworth-parents.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>15</td>
<td>17231</td>
<td>melville-moby_dick.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>11</td>
<td>9021</td>
<td>milton-paradise.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>9</td>
<td>3032</td>
<td>shakespeare-caesar.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>8</td>
<td>4716</td>
<td>shakespeare-hamlet.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>7</td>
<td>3464</td>
<td>shakespeare-macbeth.txt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>12</td>
<td>12452</td>
<td>whitman-leaves.txt</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Vocabulary size

<table>
<thead>
<tr>
<th>Vocabulary size</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 25 26 7344</td>
<td>austen-emma.txt</td>
</tr>
<tr>
<td>5 26 17 5835</td>
<td>austen-persuasion.txt</td>
</tr>
<tr>
<td>5 28 22 6403</td>
<td>austen-sense.txt</td>
</tr>
<tr>
<td>4 34 79 12767</td>
<td>bible-kjv.txt</td>
</tr>
<tr>
<td>5 19 5 1535</td>
<td>blake-poems.txt</td>
</tr>
<tr>
<td>4 19 14 3940</td>
<td>bryant-stories.txt</td>
</tr>
<tr>
<td>4 18 12 1559</td>
<td>burgess-busterbrown.txt</td>
</tr>
<tr>
<td>4 20 13 2636</td>
<td>carroll-alice.txt</td>
</tr>
<tr>
<td>5 20 12 8335</td>
<td>chesterton-ball.txt</td>
</tr>
<tr>
<td>5 23 11 7794</td>
<td>chesterton-brown.txt</td>
</tr>
<tr>
<td>5 18 11 6349</td>
<td>chesterton-thursday.txt</td>
</tr>
<tr>
<td>4 21 25 8447</td>
<td>edgeworth-parents.txt</td>
</tr>
<tr>
<td>5 26 15 17231</td>
<td>melville-moby_dick.txt</td>
</tr>
<tr>
<td>5 52 11 9021</td>
<td>milton-paradise.txt</td>
</tr>
<tr>
<td>4 12 9 3032</td>
<td>shakespeare-caesar.txt</td>
</tr>
<tr>
<td>4 12 8 4716</td>
<td>shakespeare-hamlet.txt</td>
</tr>
<tr>
<td>4 12 7 3464</td>
<td>shakespeare-macbeth.txt</td>
</tr>
<tr>
<td>5 36 12 12452</td>
<td>whitman-leaves.txt</td>
</tr>
</tbody>
</table>
Suggested (Optional) Exercises

- Compute median rather than mean lengths (for word length, sentence length, etc)

- Extend the `define_vocabulary()` code to also remove stopwords

- Extend the `define_vocabulary()` code to also remove rare words

- Define a vocabulary across all the texts and write some code to identify the most likely text source given an input sentence (the input sentence could be from one of the actual sentences or a real sentence from one of the texts). Hint: you could use a naïve Bayes classifier (next week’s lecture topic) to do this.
Backup Slides
Tokenization Software

- Instead of writing your own tokenizer with a complex set of rules, use existing software
  - e.g., tokenizer function in NLTK
  - e.g., tokenizer from Stanford’s natural language group

- Practical tip:
  - It's useful to keep different representations of the data to use later on, e.g.,
    - Data with original sequence and formatting, tokenized list, bag of words, etc
  - Sequential order of words is needed for detecting n-grams.
  - Punctuation can contain useful information:

    If you had a magic potion I’d love to have it. If that makes sense

If we decide to extract n-grams later on, we know that “it” and “if” should not be combined. So it's useful to retain this information.
# Part of Speech Tagging

<table>
<thead>
<tr>
<th>WORD</th>
<th>Part of Speech (POS) Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>if</td>
<td>IN Preposition</td>
</tr>
<tr>
<td>you</td>
<td>PRP Personal Pronoun</td>
</tr>
<tr>
<td>had</td>
<td>VBD Verb, Past Tense</td>
</tr>
<tr>
<td>a</td>
<td>DT Determiner</td>
</tr>
<tr>
<td>magic</td>
<td>JJ Adjective</td>
</tr>
<tr>
<td>potion</td>
<td>NN Noun</td>
</tr>
<tr>
<td>i</td>
<td>PRP Personal Pronoun</td>
</tr>
<tr>
<td>‘d</td>
<td>MD Modal</td>
</tr>
<tr>
<td>love</td>
<td>VB Verb, base form</td>
</tr>
<tr>
<td>to</td>
<td>TO</td>
</tr>
<tr>
<td>have</td>
<td>VB Verb, base form</td>
</tr>
<tr>
<td>it</td>
<td>PRP Personal Pronoun</td>
</tr>
</tbody>
</table>

(based on Stanford POS tagger; on dialog data)
Using POS Tags for Exploratory Data Analysis

<table>
<thead>
<tr>
<th>PRP (pronouns)</th>
<th>IN (preposition)</th>
<th>UH (uh words)</th>
<th>NN (noun singular)</th>
<th>JJ (adjective)</th>
</tr>
</thead>
<tbody>
<tr>
<td>405616 i</td>
<td>109552 of</td>
<td>46819 yeah</td>
<td>23579 something</td>
<td>14983 good</td>
</tr>
<tr>
<td>217860 you</td>
<td>84844 that</td>
<td>15755 okay</td>
<td>20907 kind</td>
<td>12979 other</td>
</tr>
<tr>
<td>190642 it</td>
<td>81317 like</td>
<td>13307 oh</td>
<td>18518 way</td>
<td>8432 little</td>
</tr>
<tr>
<td>49022 me</td>
<td>62675 in</td>
<td>7237 uh</td>
<td>18474 time</td>
<td>6228 okay</td>
</tr>
<tr>
<td>47827 he</td>
<td>41304 about</td>
<td>1453 well</td>
<td>15303 thing</td>
<td>6109 sure</td>
</tr>
<tr>
<td>36887 she</td>
<td>40723 with</td>
<td>1178 wow</td>
<td>10352 lot</td>
<td>6051 last</td>
</tr>
<tr>
<td>33905 they</td>
<td>37129 for</td>
<td>535 yes</td>
<td>8969 anything</td>
<td>5872 much</td>
</tr>
<tr>
<td>31866 we</td>
<td>36475 because</td>
<td>507 ah</td>
<td>8616 sort</td>
<td>5320 yeah</td>
</tr>
<tr>
<td>16385 him</td>
<td>35092 if</td>
<td>263 no</td>
<td>7927 um</td>
<td>5169 same</td>
</tr>
<tr>
<td>15079 them</td>
<td>29040 on</td>
<td>241 hello</td>
<td>6888 day</td>
<td>4881 bad</td>
</tr>
<tr>
<td>12251 her</td>
<td>27464 at</td>
<td>120 nope</td>
<td>5851 feeling</td>
<td>4840 different</td>
</tr>
<tr>
<td>8533 myself</td>
<td>15052 so</td>
<td>48 uh-uh</td>
<td>4972 everything</td>
<td>4482 right</td>
</tr>
<tr>
<td>4709 yourself</td>
<td>13674 as</td>
<td>21 ok</td>
<td>4922 part</td>
<td>4369 able</td>
</tr>
<tr>
<td>3276 us</td>
<td>11238 from</td>
<td>17 aw</td>
<td>4802 stuff</td>
<td>3857 uh-huh</td>
</tr>
<tr>
<td>649 himself</td>
<td>6777 than</td>
<td>4 heck</td>
<td>4769 person</td>
<td>3855 like</td>
</tr>
<tr>
<td>494 herself</td>
<td>6466 into</td>
<td>3 bam</td>
<td>4672 week</td>
<td>3782 whole</td>
</tr>
<tr>
<td>472 themselves</td>
<td>6253 out</td>
<td>3 boy</td>
<td>4611 night</td>
<td>3714 first</td>
</tr>
</tbody>
</table>

Text data from dialog between 2 speakers
Removing Stop Words with a POS Tagger

- Filter out any term that does not belong to a (user-defined) target set of POS classes

<table>
<thead>
<tr>
<th>SPEAKER</th>
<th>WORD</th>
<th>POSTAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATIENT</td>
<td>if</td>
<td>IN</td>
</tr>
<tr>
<td>PATIENT</td>
<td>you</td>
<td>PRP</td>
</tr>
<tr>
<td>PATIENT</td>
<td>had</td>
<td>VBD</td>
</tr>
<tr>
<td>PATIENT</td>
<td>a</td>
<td>DT</td>
</tr>
<tr>
<td>PATIENT</td>
<td>magic</td>
<td>JJ</td>
</tr>
<tr>
<td>PATIENT</td>
<td>potion</td>
<td>NN</td>
</tr>
<tr>
<td>PATIENT</td>
<td>i</td>
<td>PRP</td>
</tr>
<tr>
<td>PATIENT</td>
<td>‘d</td>
<td>MD</td>
</tr>
<tr>
<td>PATIENT</td>
<td>love</td>
<td>VB</td>
</tr>
<tr>
<td>PATIENT</td>
<td>to</td>
<td>TO</td>
</tr>
<tr>
<td>PATIENT</td>
<td>have</td>
<td>VB</td>
</tr>
<tr>
<td>PATIENT</td>
<td>it</td>
<td>PRP</td>
</tr>
</tbody>
</table>

Example rule: extract adjectives and nouns only
Extracting Ngrams

Extracting bigram tokens:

... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
Filter out “bad” Ngrams

Extracting bigram tokens:

... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense
... magic potion I’d love to have it. If that makes sense

Allowable combinations (according to user-defined POS tag filter)

Excluded combinations
Stemming

• Maps several terms onto one base form
  – “alcohol”, “alcoholic”, “alcoholics” all reduced to same token “alcohol”

• Advantages of stemming
  – Increases overlap in terms across documents: can help in assessing similarity
  – Might be beneficial when analyzing small data sets

• Disadvantages
  – Correct morphological analysis is language specific and can be complex
  – Mixed effects of stemming have been found in information-retrieval
Stemming using Dictionaries

- Consult dictionary such as CELEX to create a list of words with alternate word endings.
- Example stemming list:
  - alcohol
  - alcoholic alcoholics
  - alcoholism
  - alert
  - alex
  - alhambra
  - alias
  - alice
  - alienated alienate alienating
  - alienation
  - aligned
  - alignment
  - alike
  - alirhght
  - alist
  - alive

Counts for the words “alienated”, “alienate” and “alienating” will be collapsed. This makes it easier to find correlations.
Porter Stemmer

- Simple procedure for removing known affixes in English without using a dictionary.

- Can produce unusual stems that are not English words:
  - “computer”, “computational”, “computation” all reduced to same token “comput”
  - This complicates interpretability

Marrying this guy was a big mistake

Marri thi guy wa a big mistak
Porter Stemmer Errors

• Errors of “commission”:
  – organization, organ → organ
  – police, policy → polic
  – arm, army → arm

• Errors of “omission”:
  – cylinder, cylindrical
  – create, creation
  – Europe, European
Syntactic and Semantic Parsing

Syntactic dependency parse

Frame-semantic parse

http://demo.ark.cs.cmu.edu/
Concordance

Analyze the context in which a term appears

tanglements begin to have a significant effect on the relaxation times. The undiluted syringe even more doses. Although its effect on the circulation of wild polioviruses had their properties would have a beneficial effect on the overall scheme, members heard. As rabbits or sheep, has a devastating effect on the fine-leaved bouncy turf rich in sedge, such groups must have had a major effect on the structure of the forest. The question whether artemether has a beneficial effect on the objective and unambiguous primal government and that has inevitably had an effect on the level of the charge. "This is frightfully bad: meat and biscuits had had a ruinous effect on the housekeeping. Happily Herbert has been talking about had had a very bad effect on the Quigleys. Mrs Quigley was hypertensive. Oleoresins of the dipterocarps have an effect on the bacteria of the fore-stomach of cows, but progressive and compensatory in effect. On the circumference of that circle are many possibilities of charging for more services. The effect on the demography of the inner cities continues in April 1988 have had a devastating effect on young people. At the stroke of a pen they can return to her to worry about the devastating effect Paula was having on Edward. Behind the scenes, and for public health activities. Thus in effect reference centres are indistinguishable from a matrix between 'knowledge of a cause/effect relationship between participation program...
Sentiment Lexicons

- Basic analysis of text:
  - Overall count and percentage of words in various categories

- Example lexicons
  - General inquirer ([http://www.wjh.harvard.edu/~inquirer](http://www.wjh.harvard.edu/~inquirer))
    - Words categorized according to Positive / Negative, Strong vs Weak, Active vs Passive, etc
  - Sentiwordnet ([http://sentiwordnet.isti.cnr.it/](http://sentiwordnet.isti.cnr.it/))
    - Synsets in WordNet3.0 annotated for degrees of positivity, negativity, and neutrality/objectiveness
    - Next slide
LIWC (Linguistic Inquiry and Word Count)


- 2300 words, >70 classes
- **Affective Processes**
  - negative emotion (*bad, weird, hate, problem, tough*)
  - positive emotion (*love, nice, sweet*)
- **Cognitive Processes**
  - Tentative (*maybe, perhaps, guess*), Inhibition (*block, constraint*)
- **Pronouns, Negation** (*no, never*), **Quantifiers** (*few, many*)
## LIWC Word Categories

<table>
<thead>
<tr>
<th>Category</th>
<th>1st person singular</th>
<th>1st person plural</th>
<th>2nd person</th>
<th>Pronouns</th>
<th>Positive emotions</th>
<th>Negative emotions</th>
<th>Affect</th>
<th>Hearing</th>
<th>Feeling</th>
<th>Body</th>
<th>Sexual</th>
<th>Motion</th>
<th>Space</th>
<th>Cognitive mechanisms</th>
<th>Time</th>
<th>Occupation</th>
<th>Achievement</th>
<th>Leisure</th>
<th>Home</th>
<th>Money</th>
<th>Religion</th>
<th>Death</th>
<th>Assent</th>
<th>Nonfluencies</th>
</tr>
</thead>
</table>
Pros and Cons of Dictionary Approaches such as LIWC

- **Pros**
  - Effective method for studying the various emotional, cognitive, structural and process components present in individual’s verbal and written speech.
  - Easy to use

- **Cons**
  - Sentiment lexicons are fixed in number of categories and words in categories
  - Word context is often ignored
  - Not domain specific
## Example of a Document-Term Matrix

<table>
<thead>
<tr>
<th></th>
<th>water</th>
<th>farming</th>
<th>cattle</th>
<th>agriculture</th>
<th>land rights</th>
<th>use</th>
<th>stock</th>
<th>portfolio</th>
<th>bonds</th>
<th>return</th>
<th>loan</th>
<th>interest rate</th>
<th>profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>doc6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>