Stats 170A: Project in Data Science

Text Analysis and Classification

Padhraic Smyth
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

(Acknowledgements to Prof Mark Steyvers and Prof Sameer Singh, UCI, for various slides)
Reading, Homework, Lectures

• Reference reading:
  – Python
    • http://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

• Homework 7
  – Due by 2pm Wednesday next week

• Next Lectures
  – Today: text analysis and classification
  – Next week: discussion of 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)
第13届上海国际电影节开幕...

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
Housing prices rose

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

Part-of-speech (POS) tagging

Let’s go to Agra!

Buy V1AGRA ...

Colorless green ideas sleep furiously.

Named entity recognition (NER)

Einstein met with UN officials in Princeton
Room for improvement....

“您需要开始理解我，Siri。”

“我将记下这个。”

“是的，你最好记下这个。”

Got it:

Of that
## Predicting the Sentiment of Tweets

<table>
<thead>
<tr>
<th>Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>wow :o this was included in the playlist #awesome</td>
</tr>
<tr>
<td>I could not be happier with my life right now</td>
</tr>
<tr>
<td>HAPPY BIRTHDAY TO THE PANDERIFIC PANDA IK.@TheOrionSound</td>
</tr>
<tr>
<td>I'm tired as hell I never get a off day during the week anymore</td>
</tr>
<tr>
<td>The movie was really dull and stupid</td>
</tr>
<tr>
<td>@washingtonpost @silvajanes How awful!!!!!!</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“Documents”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Labels</td>
</tr>
</tbody>
</table>
## Predicting the Scores of Movie Reviews

<table>
<thead>
<tr>
<th>Review</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>I liked this movie. Well done. Great acting and direction</td>
<td>4</td>
</tr>
<tr>
<td>Terrific movie, saw it 3 times. Loved the scenes in Mexico.</td>
<td>5</td>
</tr>
<tr>
<td>Boring as hell. Who gets paid to create this stuff?</td>
<td>1</td>
</tr>
<tr>
<td>Not one of the director’s best efforts. Tom Cruise was terrible.</td>
<td>2</td>
</tr>
</tbody>
</table>

“Documents”

<table>
<thead>
<tr>
<th>Ratings</th>
</tr>
</thead>
</table>
## Entity Recognition

<table>
<thead>
<tr>
<th>Review</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>I liked this movie. Well done. Great acting and direction</td>
<td>4</td>
</tr>
<tr>
<td>Terrific movie, saw it 3 times. Loved the scenes in <strong>Mexico.</strong></td>
<td>5</td>
</tr>
<tr>
<td>Boring as hell. Who gets paid to create this stuff?</td>
<td>1</td>
</tr>
<tr>
<td>Not one of the director’s best efforts. <strong>Tom Cruise</strong> was terrible.</td>
<td>2</td>
</tr>
</tbody>
</table>

**“Documents”**
Examples of User Location Fields in Twitter

- User location: Deutschland
- User location: Mountain View, CA
- User location: Florida, USA
- User location: United States
- User location: West Virginia, Appalachia
- User location: Columbia Mo
- User location: South Florida
- User location: Germany/Berlin
- User location: St Louis, MO
- User location: St. Louis
- User location: Central Virginia
- User location: United States
- User location: Sea Holme (of Norse legend) Oz
- User location: California, USA
- User location: USA
- User location: Cambodia
- User location: San Antonio, TX
- User location: Between my ears
- User location: Chicago, IL
Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, and an active discussion forum.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The book is being updated for Python 3 and NLTK 3. (The original Python 2 version is still available at http://nltk.org/book_1ed.)

Some simple things you can do with NLTK

Tokenize and tag some text:

```python
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning ...
  Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
```
scikit-learn
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification
Identifying to which set of categories a new observation belong to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, ...

Regression
Predicting a continuous value for a new example.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ...

Clustering
Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes.
Algorithms: k-Means, spectral clustering, mean-shift, ...

Dimensionality reduction
Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, Isomap, non-negative matrix factorization.

Model selection
Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics.

Preprocessing
Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.

News
On-going development: What's new (changelog)

Community
Questions? See stackoverflow # scikit-learn
Mailing list: scikit-learn-

Who uses scikit-learn?
Basic Concepts in Text Representation
Representing Text Documents Numerically

• How do we represent text (strings) for input to machine learning algorithms? (which generally require numeric representations)

• Define a vocabulary = set of words or terms

• Simple method: Bag of Words
  – Document represented as a vector of counts of words in the vocabulary

• More complex: Real-Valued Embeddings
  – Document (or word) represented as a real-valued vector in “embedding space”

• Even more complex: Sequential Representations
  – Sequential state-machine model for sequences of words
## 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>
Concepts and Terminology

• Document:
  – Collection of words
    • A book, a news article, a report, a Web page, an email, a tweet, etc
  – May contain both text and metadata.
  – Examples of metadata: author name(s), date, where published, etc

Note that the definition of a document is flexible
  e.g., a book could be a single document, or ..... each section of a book could be considered a “document”

• Corpus: a collection of documents
  – e.g., all news articles from the Los Angeles Times since 1990
  – e.g., all Wikipedia Web pages
  – e.g., all Yelp reviews for restaurants in Chicago
  – e.g., a random sample of Tweets from Dec 2017
Concepts and Terminology

- **Tokens:**
  - Groupings of characters in the raw text
  - Individual words (or “word types”) + possibly numbers, punctuation, etc

- **Words 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 specific set of unique terms used by an algorithm or application
  - The English language has order of 1 million unique words
  - In a particular application we might use a vocabulary of only 10k to 50k terms
    - E.g., relevant/common words (unigrams)
    - Bigram, trigrams, …, ngrams, can also be part of the vocabulary
Typical Pipeline for Building a Text Classifier

Document (string)

- Tokenization
- StopWord Removal
- Vocabulary Definition
- Create Bag of Words
- Build Text Classifier

Note that 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.
Example of a Text Analysis Pipeline (for Information Extraction)

Example

Document = ‘Chapter 1: The Beginning. In the beginning, life was tough!...........


(Punctuation and white spaces are usually ignored)
Example

Document = ‘Chapter 1: The Beginning. In the beginning, life was tough!.........’

Tokens = {'chapter', '1', 'the', 'beginning', 'in', 'the', 'beginning', 'life', 'was', 'tough', ...}
(Punctuation and white spaces are usually ignored)

Vocabulary = {'chapter', '1', 'the', 'beginning', 'in', 'life', 'was', 'tough', ...}
Example

Document = ‘Chapter 1: The Beginning. In the beginning, life was tough!.........’

(Punctuation and white spaces are usually ignored)


Bag of Words = {[‘chapter’, 1], [‘1’, 1], [‘the’, 2], [‘beginning’, 2], …}
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
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:
  – Its 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 ngrams.
  – 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 its useful to retain this information.
Sentence Detection: Example using a Decision Tree

From https://web.stanford.edu/~jurafsky/slp3/
_Speech and Language Processing_, 3rd ed, Jurafsky and Martin
Example: Tokenization of Tweets

There are special-purpose tokenizers for different types of text, e.g., for Tweets

```python
from nltk.tokenize import TweetTokenizer
from nltk.tokenize import word_tokenize

tknzr = TweetTokenizer()

for i in range(10):
    print("NLTK Twitter Tokenizer: ", tknzr.tokenize(tweets.text[i]) )
    print("NLTK Standard Tokenizer:", word_tokenize(tweets.text[i]))
```
Sample Output from the 2 Tokenizers

**NLTK Twitter Tokenizer:** ['RT', '@ProudResister', ':', 'Trump', 'to', 'Comey', ':', 'Let', 'Flynn', 'go', ',', 'Trump', 'to', 'Russians', ':', 'Firing', 'Nut', 'Job', 'Comey', 'took', 'great', 'pressure', 'off', 'me', ',', 'Trump', 'to', 'McCabe', ':', 'Who', '…']


**NLTK Twitter Tokenizer:** ['Mueller', 'to', 'interview', 'former', 'spokesman', 'of', 'Trump', 'legal', 'team', ':', 'source', 'https://t.co/olA35uCt89']

**NLTK Standard Tokenizer:** ['Mueller', 'to', 'interview', 'former', 'spokesman', 'of', 'Trump', 'legal', 'team', ':', 'source', 'https://t.co/olA35uCt89']

**NLTK Twitter Tokenizer:** ['RT', '@IndivisibleTeam', ':', 'Here's', 'what', 'you', 'need', 'to', 'know', 'about', '#ReleaseTheMemo', '1', '1️⃣', 'It', '"', 's', 'not', 'really', 'a', 'memo', ',', 'They're', 'talking', 'points', 'by', 'Devin', 'Nunes', '…']

**NLTK Standard Tokenizer:** ['RT', '@', 'IndivisibleTeam', ':', 'Here', '"s', 'what', 'you', 'need', 'to', 'know', 'about', '#', 'ReleaseTheMemo ️⃣', 'It’s', 'not', 'really', 'a', 'memo', ',', 'They', '"re', 'talking', 'points', 'by', 'Devin', 'Nunes…']
Example: Defining a Vocabulary

Raw text (a string in Python)

```python
raw1 = "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)
Defining the Vocabulary (after Tokenization)

- **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”
Frequency of Word in English usage

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

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., the term “restaurant” in reviews
N-grams

• Useful n-grams are groups of n-words that commonly appear in sequence
  – “Computer Science”
  – “Los Angeles”
  – “New York City”

  – Note that we can have both terms like “computer” and “computer science”, e.g.,

    “I am studying computer science at UCI and I bought a new computer last week.”

- Adding an n-gram as a term in the vocabulary may be useful in a model
  - e.g., for a restaurant review classification problem, “wait time” may be more informative than “time”
Automatically Finding N-grams

- e.g., Bigrams
  - Keep track of all unique pairs that occur sequentially (exclude stopwords)

- I visited Microsoft Research for a computer science job interview last week.”
  - *visited Microsoft Research*  *computer science job interview last week.*

- Rank by frequency of occurrence (or number of docs a bigram appears in)
  - Can also rank by other metrics

- Keep the top K in the vocabulary
  - How large should K be? Depends on the application, amount of data, etc
  - Might need to search over different values of K (e.g., using cross-validation)

- Same idea for tri-grams, 4 grams, etc.

- See also [http://www.nltk.org/howto/collocations.html](http://www.nltk.org/howto/collocations.html) in NLTK
## 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>
### 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>
<th>Label</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>
<td>1</td>
</tr>
<tr>
<td>d2</td>
<td>32</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d3</td>
<td>12</td>
<td>16</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d4</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d5</td>
<td>43</td>
<td>31</td>
<td>20</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>7</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>d7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>12</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>d8</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>4</td>
<td>2</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>
<td>2</td>
</tr>
<tr>
<td>d10</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>4</td>
<td>23</td>
<td>2</td>
</tr>
</tbody>
</table>

Use labeled training data (supervised learning) to learn a classifier – Homework Assignment 7
Overview of Text Classification
Text Classification

- Text classification has many applications
  - Spam email detection
  - Classifying news articles, e.g., Google News
  - Classifying Web pages into categories

- Data Representation
  - “Bag of words/terms” most commonly used: either counts or binary
  - Can also use other weighting and additional features (e.g., metadata)

- Classification Methods
  - Naïve Bayes widely used baseline
    - Fast and reasonably accurate
  - Logistic Regression
    - Widely used in industry, accurate, excellent baseline
  - Neural networks and deep learning
    - Can be very accurate
    - Can require very large amounts of labeled training data
    - More complex than other methods
Example of Document by Term Matrix

<table>
<thead>
<tr>
<th></th>
<th>predict</th>
<th>finance</th>
<th>stocks</th>
<th>goal</th>
<th>score</th>
<th>team</th>
<th>Class Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d4</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>d7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>d8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>d9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>d10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
## Possible Weights for a Linear Classifier

<table>
<thead>
<tr>
<th>Weight</th>
<th>predict</th>
<th>finance</th>
<th>stocks</th>
<th>goal</th>
<th>score</th>
<th>team</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1</td>
<td>2.5</td>
<td>1.0</td>
<td>-2.0</td>
<td>-0.2</td>
<td>-1.5</td>
</tr>
<tr>
<td>d1</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>d3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d4</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>d5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>d6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>d7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>d8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>d9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>d10</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Class Label

1

1

1

1

1

2

2

2

2
### Real Example from Yelp Data

#### Yelp Dataset

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Reviews</td>
<td>706,693</td>
</tr>
<tr>
<td>Number of Reviews w/o Neutral Rating</td>
<td>595,468</td>
</tr>
<tr>
<td>Number of Tokens</td>
<td>85,392,376</td>
</tr>
<tr>
<td>Vocabulary Size w/o Stopwords</td>
<td>176,114</td>
</tr>
<tr>
<td>Array Dimensions</td>
<td>(595468, 176114)</td>
</tr>
<tr>
<td>Number of cells in the Array</td>
<td>104,870,251,352</td>
</tr>
<tr>
<td>Non-zero entries</td>
<td>28,357,001</td>
</tr>
<tr>
<td>Density</td>
<td>0.0027%</td>
</tr>
</tbody>
</table>
Example of a Pipeline for Document Classification

Training Documents (corpus) → Tokenization → Lists of Tokens → Bag of Words → Vocabulary → Frequency Counts → Document Classifier → Machine Learning Algorithm

- Stopword and rare word removal
- Frequency Counts
Example of a Pipeline for Document Classification

Training Documents (corpus)

Lists of Tokens

Bag of Words

Machine Learning Algorithm

Vocabulary

Stopword and rare word removal

Frequency Counts

New Document

Lists of Tokens

Bag of Words

Document Classifier

Label Prediction

Tokenization

Tokenization
Key Steps in Document Analysis Pipelines (for Bag of Words)

- **Tokenization**
  - Various options (e.g., with punctuation, non alphanumeric symbols, etc)

- **Vocabulary definition**
  - N-grams, stopword removal, rare word removal, stemming

- **Feature definition**
  - Binary (term present or not?)
  - Counts
  - Weighted counts, e.g., TD-IDF (see later in the slides)

- **Classifier selection**
  - Naïve Bayes, logistic, SVMs, neural networks, etc
TF-IDF Weighting of Features

In practice the inputs can be weighted

- It can be helpful to use “TF-IDF weights” instead of counts

\[
TF(t,d) = \text{term frequency} = \text{count} = \text{number of times term t occurs in doc d}
\]

\[
IDF(t,d) = \text{inverse document frequency}
= \log\left(\frac{N}{\text{number of docs with term t}}\right)
\]

(\text{where N = total number of docs in the corpus})

\[
\text{TF-IDF}(t,d) = TF(t,d) \times IDF(t,d)
\]

The IDF term has the effect of upweighting terms that occur in few docs
TF-IDF Example

N = 1000 in a corpus of news articles

Term 1: $t = \text{“city”}$, appears in 500 documents

$$\text{IDF}(t) = \log(1000/500) = \log(2) = 1$$

$log$ is base 2, not important

Term 2: $t = \text{“freeway”}$, appears in 10 documents

$$\text{IDF}(t) = \log(1000/10) = \log(100) = 6.64$$

So occurrences of “freeway” will get upweighted by a factor of 6.64 compared to occurrences of “city”
Classification Methods for Text

• Logistic regression
  – widely used for bag-of-words
  – Input dimensionality (vocabulary size) is high (e.g., 5k to 500k)
  – regularization is useful

• Recurrent neural networks are widely used for sequential models (see later)
  – More complex, but can pick up sequential information
This starts to make more sense. Columns are target classes. In each column there are features and their weights. Intercept (bias) feature is shown as $<\text{BIAS}>$ in the same table. We can inspect features and weights because we're using a bag-of-words vectorizer and a linear classifier (so there is a direct mapping between individual words and classifier coefficients). For other classifiers features can be harder to inspect.

from: brian@ucsd.edu (brian kantor) subject: re: help for kidney stones organization: the avant-garde of the now, ltd. lines: 12 nntp-posting-host: ucsd.edu as i recall from my bout with kidney stones, there isn’t any medication that can do anything about them except relieve the pain. either they pass, or they have to be broken up with sound, or they have to be extracted surgically. when i was in, the x-ray tech happened to mention that she’d had kidney stones and children, and the childbirth hurt less. demerol worked, although i nearly got arrested on my way home when i barfed all over the police car parked just outside the er. - brian
y=soc.religion.christian (probability 0.001, score -7.157) top features

<table>
<thead>
<tr>
<th>Contribution</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.258</td>
<td>&lt;BIAS&gt;</td>
</tr>
<tr>
<td>-6.899</td>
<td>Highlighted in text (sum)</td>
</tr>
</tbody>
</table>

from: brian@ucsd.edu (brian kantor) subject: re: help for kidney stones .............. organization: the avant-garde of the now, ltd. lines: 12 nntp-posting-host: ucsd.edu as i recall from my bout with kidney stones, there isn't any medication that can do anything about them except relieve the pain. either they pass, or they have to be broken up with sound, or they have to be extracted surgically. when i was in, the x-ray tech happened to mention that she'd had kidney stones and children, and the childbirth hurt less. demerol worked, although i nearly got arrested on my way home when i barfed all over the police car parked just outside the er. - brian

ELI5 is a Python package which helps to debug machine learning classifiers and explain their predictions.

It provides support for the following machine learning frameworks and packages:

- **scikit-learn.** Currently ELI5 allows to explain weights and predictions of scikit-learn linear classifiers and regressors, print decision trees as text or as SVG, show feature importances and explain predictions of decision trees and tree-based ensembles. ELI5 understands text processing utilities from scikit-learn and can highlight text data accordingly. It also allows to debug scikit-learn pipelines which contain HashingVectorizer, by undoing hashing.
- **xgboost** - show feature importances and explain predictions of XGBClassifier and XGBRegressor.
- **lightning** - explain weights and predictions of lightning classifiers and regressors.
- **sklearn-crfsuite.** ELI5 allows to check weights of sklearn_crfsuite.CRF models.

From: https://github.com/TeamHG-Memex/eli5
Sentiment Analysis
Sentiment Symposium Tutorial

Sentiment Analysis Symposium, San Francisco, November 8-9, 2011

Instructor: Christopher Potts (Stanford Linguistics)

Overview

1. Overview: goals, plan, and applications
2. Sentiment in language and cognition
3. Text preparation:
   a. Tokenization
   b. Stemming
   c. Advanced linguistic structure
4. Sentiment lexicons
5. Classifier models for sentiment
6. Vector space models
7. Context-dependency and social relationships
8. Sentiment summarization
9. Bibliography

Very useful tutorial, many practical tips and ideas

Online at http://sentiment.christopherpotts.net/
Example: Predicting if Review Text is Positive or Negative

- Two basic approaches
  - Use lexicons of positive and negative words
  - Use labeled data to learn a classification model
    (can also combine both approaches)

- Challenges, e.g., how to handle negation?
  - Positive review:
    - “This movie is not depressing the way I thought it would be. I don’t think I have anything negative to say about it.”
    - “I wasn’t disappointed at all with the acting – in fact the acting was excellent.”

  - Negative reviews:
    - “Sitting through this movie was not something I enjoyed doing”
    - “This wasn’t a very good script. The director is usually excellent (some really great movies over the years that I really enjoyed), but this leaves me cold”
Negation Marking (during Tokenization)

- Idea: append a _NEG suffix to every word appearing between a negation and a clause-level punctuation mark

**Definition: Negation**

A negation is any word matching the following regular expression:

```regex
(?:
  ^(?::never|no|nothing|nowhere|noone|none|not|
  havent|hasnt|hadnt|cant|couldnt|shouldnt|
  wont|wouldnt|dont|doesnt|didnt|isnt|arent|aint
  )$
)
| n't
```

**Definition: Clause-level punctuation**

A clause-level punctuation mark is any word matching the following regular expression:

```regex
^[.::!?]$
```

From http://sentiment.christopherpotts.net/lingstruc.html
Examples of Negation Marking

No one enjoys it.

I don't think I will enjoy it: it might be too spicy.

From http://sentiment.christopherpotts.net/lingstruc.html
Improvements in Classification Accuracy

OpenTable; 6000 reviews in test set (1% = 60 reviews)

From http://sentiment.christopherpotts.net/lingstruc.html
Sentiment Lexicons (Dictionaries)

- Counts, percentages, weights of words in various categories,
  - e.g., degree to which a word tends to positive or negative

- 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
  - NRC Word-Emotion Association Lexicon (next slide)
Yelp Dataset Challenge
Discover what insights lie hidden in our data.

**The Challenge**
We challenge students to use our data in innovative ways and break ground in research. Here are some examples of topics we find interesting, but remember these are only to get you thinking and we welcome novel approaches!

**Photo Classification**
Maybe you've heard of our ability to identify hot dogs (and other foods) in photos. Or how we can tell you if your photo will be beautiful or not. Can you do better?

![Photo Classification Images](image-url)

**Natural Language Processing & Sentiment Analysis**
What's in a review? Is it positive or negative? Our reviews contain a lot of metadata that can be mined and used to infer meaning, business attributes, and sentiment.

**Graph Mining**
We recently launched our Local Graph but can you take the graph further? How do user's relationships define their usage patterns? Where are the trend setters eating before it becomes popular?

---

**Round 11**
Our dataset has been updated for this iteration of the challenge - we're sure there are plenty of interesting insights waiting there for you. This set includes information about local businesses in 11 metropolitan areas across 4 countries. Round 11 has kicked off starting January 18, 2018 and will run through June 30, 2018.

[Download Dataset]
Yelp Challenge DataSet
Jupyter Notebook Example with Yelp Dataset
Vector Representations for Text
Vector Representations allow Generalization

- **Example:**
  - Cat: \[1.5, 1.5, -0.2, 0.0, 0.0, 0.0]\]
  - Kitten: \[1.4, 1.5, -0.2, 0.0, 0.0, 0.0]\]
  - Pet: \[2.0, 1.6, -0.3, 0.0, 0.0, 0.0]\]

- If the word “cat” occurs in a document then we know that other words like “kitten” and “pet” are similar.

- Why is this useful in learning?
  - Consider a classifier based on weights (e.g., logistic regression, neural network).
  - Traditional approach: each word has its own separate weight.
  - If we represent words by their vectors, and learn weights on the vectors, then words that are close together will produce similar outputs.
  - This can help with generalization.
Another Example of Word Embedding
Publicly-Available Pre-Trained Word Embeddings

- **Word2Vec**: https://code.google.com/archive/p/word2vec/
  - Google News dataset, 3M vocab (words and phrases), from 100B tokens
  - Entity vectors: 1.4M vectors for entities, trained on 100B words from news articles, entity names from Freebase

- **Glove**: http://nlp.stanford.edu/projects/glove/
  - Wikipedia: 400k vocab, 50d to 300d vectors, based on 6B tokens
  - Common Crawl, 2.2M vocab, 300d vectors, from 840B tokens
  - Twitter: 1.2M vocab, 25d to 200d vectors, based on 2B tweets

- **Note**: you may want to consider using these in your projects
  - Easier and faster than training your own embedding model
Cosine Distance between two Vectors

In this example, \( y \) and \( z \) are more similar under cosine similarity than \( y \) and \( x \) because the angle between \( y \) and \( z \) is much smaller.

Cosine distance = 1 - Cosine similarity: measures cosine of the angle between 2 vectors, where \( \cos(0) = 1 \), and \( \cos(90) = 0 \).
## Examples of Similarity between Words

Examples from https://code.google.com/archive/p/word2vec/

### Cosine similarity to “France”

<table>
<thead>
<tr>
<th>Country</th>
<th>Cosine Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>spain</td>
<td>0.678515</td>
</tr>
<tr>
<td>belgium</td>
<td>0.665923</td>
</tr>
<tr>
<td>netherlands</td>
<td>0.652428</td>
</tr>
<tr>
<td>italy</td>
<td>0.633130</td>
</tr>
<tr>
<td>switzerland</td>
<td>0.622323</td>
</tr>
<tr>
<td>luxembourg</td>
<td>0.610033</td>
</tr>
<tr>
<td>portugal</td>
<td>0.577154</td>
</tr>
<tr>
<td>russia</td>
<td>0.571507</td>
</tr>
<tr>
<td>germany</td>
<td>0.563291</td>
</tr>
<tr>
<td>catalonia</td>
<td>0.534176</td>
</tr>
</tbody>
</table>

### Cosine similarity to “San Francisco”

<table>
<thead>
<tr>
<th>Location</th>
<th>Cosine Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>los_angeles</td>
<td>0.666175</td>
</tr>
<tr>
<td>golden_gate</td>
<td>0.571522</td>
</tr>
<tr>
<td>oakland</td>
<td>0.557521</td>
</tr>
<tr>
<td>california</td>
<td>0.554623</td>
</tr>
<tr>
<td>san_diego</td>
<td>0.534939</td>
</tr>
<tr>
<td>pasadena</td>
<td>0.519115</td>
</tr>
<tr>
<td>seattle</td>
<td>0.512098</td>
</tr>
<tr>
<td>taiko</td>
<td>0.507570</td>
</tr>
<tr>
<td>houston</td>
<td>0.499762</td>
</tr>
<tr>
<td>chicago_illinois</td>
<td>0.491598</td>
</tr>
</tbody>
</table>

**Cosine similarity:**
measures cosine of the angle between 2 vectors, where \( \cos(0) = 1 \), and \( \cos(90) = 0 \)
Most Similar Vectors to ‘cat’ from word2vec 300d embeddings

<table>
<thead>
<tr>
<th>Rank</th>
<th>Word</th>
<th>Cosine Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>cat</td>
<td>1.00</td>
</tr>
<tr>
<td>2:</td>
<td>cats</td>
<td>0.81</td>
</tr>
<tr>
<td>3:</td>
<td>dog</td>
<td>0.76</td>
</tr>
<tr>
<td>4:</td>
<td>kitten</td>
<td>0.75</td>
</tr>
<tr>
<td>5:</td>
<td>feline</td>
<td>0.73</td>
</tr>
<tr>
<td>6:</td>
<td>beagle</td>
<td>0.72</td>
</tr>
<tr>
<td>7:</td>
<td>puppy</td>
<td>0.71</td>
</tr>
<tr>
<td>8:</td>
<td>pup</td>
<td>0.69</td>
</tr>
<tr>
<td>9:</td>
<td>pet</td>
<td>0.69</td>
</tr>
<tr>
<td>10:</td>
<td>felines</td>
<td>0.68</td>
</tr>
</tbody>
</table>
## Most Similar Vectors to ‘cat’ from word2vec 300d embeddings

<table>
<thead>
<tr>
<th>Rank</th>
<th>Word</th>
<th>Cosine Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>cat</td>
<td>1.00</td>
</tr>
<tr>
<td>2:</td>
<td>cats</td>
<td>0.81</td>
</tr>
<tr>
<td>3:</td>
<td>dog</td>
<td>0.76</td>
</tr>
<tr>
<td>4:</td>
<td>kitten</td>
<td>0.75</td>
</tr>
<tr>
<td>5:</td>
<td>feline</td>
<td>0.73</td>
</tr>
<tr>
<td>6:</td>
<td>beagle</td>
<td>0.72</td>
</tr>
<tr>
<td>7:</td>
<td>puppy</td>
<td>0.71</td>
</tr>
<tr>
<td>8:</td>
<td>pup</td>
<td>0.69</td>
</tr>
<tr>
<td>9:</td>
<td>pet</td>
<td>0.69</td>
</tr>
<tr>
<td>10:</td>
<td>felines</td>
<td>0.68</td>
</tr>
<tr>
<td>11:</td>
<td>chihuahua</td>
<td>0.67</td>
</tr>
<tr>
<td>12:</td>
<td>pooch</td>
<td>0.67</td>
</tr>
<tr>
<td>13:</td>
<td>kitties</td>
<td>0.67</td>
</tr>
<tr>
<td>14:</td>
<td>dachshund</td>
<td>0.67</td>
</tr>
<tr>
<td>15:</td>
<td>poodle</td>
<td>0.66</td>
</tr>
<tr>
<td>16:</td>
<td>stray_cat</td>
<td>0.66</td>
</tr>
<tr>
<td>17:</td>
<td>Shih_Tzu</td>
<td>0.66</td>
</tr>
<tr>
<td>18:</td>
<td>tabby</td>
<td>0.66</td>
</tr>
<tr>
<td>19:</td>
<td>basset_hound</td>
<td>0.65</td>
</tr>
<tr>
<td>20:</td>
<td>golden_retriever</td>
<td>0.65</td>
</tr>
</tbody>
</table>
Most Similar Vectors to ‘oxycontin’ from word2vec 300d embeddings

<table>
<thead>
<tr>
<th>Rank</th>
<th>Word</th>
<th>Cosine Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>oxycontin</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>Oxycontin</td>
<td>0.79</td>
</tr>
<tr>
<td>3</td>
<td>Oxycodone</td>
<td>0.71</td>
</tr>
<tr>
<td>4</td>
<td>OxyContin</td>
<td>0.69</td>
</tr>
<tr>
<td>5</td>
<td>morphine_pills</td>
<td>0.67</td>
</tr>
<tr>
<td>6</td>
<td>hydrocodone</td>
<td>0.67</td>
</tr>
<tr>
<td>7</td>
<td>Lortab</td>
<td>0.67</td>
</tr>
<tr>
<td>8</td>
<td>oxycodone</td>
<td>0.65</td>
</tr>
<tr>
<td>9</td>
<td>OxyContin_pills</td>
<td>0.65</td>
</tr>
<tr>
<td>10</td>
<td>Hydrocodone</td>
<td>0.65</td>
</tr>
<tr>
<td>11</td>
<td>Alprazolam</td>
<td>0.64</td>
</tr>
<tr>
<td>12</td>
<td>Oxycodone_pills</td>
<td>0.63</td>
</tr>
<tr>
<td>13</td>
<td>Xanax</td>
<td>0.62</td>
</tr>
<tr>
<td>14</td>
<td>Lorcet</td>
<td>0.62</td>
</tr>
<tr>
<td>15</td>
<td>Percocet_pills</td>
<td>0.61</td>
</tr>
<tr>
<td>16</td>
<td>Valium</td>
<td>0.61</td>
</tr>
<tr>
<td>17</td>
<td>Diazepam</td>
<td>0.60</td>
</tr>
<tr>
<td>18</td>
<td>methadone_pills</td>
<td>0.60</td>
</tr>
<tr>
<td>19</td>
<td>hydrocodone_pills</td>
<td>0.60</td>
</tr>
<tr>
<td>20</td>
<td>prescription_pills</td>
<td>0.59</td>
</tr>
<tr>
<td>Rank</td>
<td>Word</td>
<td>Cosine Similarity</td>
</tr>
<tr>
<td>------</td>
<td>---------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>1:</td>
<td>iran</td>
<td>1.00</td>
</tr>
<tr>
<td>2:</td>
<td>pakistan</td>
<td>0.66</td>
</tr>
<tr>
<td>3:</td>
<td>israel</td>
<td>0.64</td>
</tr>
<tr>
<td>4:</td>
<td>lebanon</td>
<td>0.62</td>
</tr>
<tr>
<td>5:</td>
<td>russia</td>
<td>0.62</td>
</tr>
<tr>
<td>6:</td>
<td>iraq</td>
<td>0.61</td>
</tr>
<tr>
<td>7:</td>
<td>egypt</td>
<td>0.60</td>
</tr>
<tr>
<td>8:</td>
<td>Irans</td>
<td>0.59</td>
</tr>
<tr>
<td>9:</td>
<td>afghanistan</td>
<td>0.59</td>
</tr>
<tr>
<td>10:</td>
<td>Iran</td>
<td>0.59</td>
</tr>
<tr>
<td>11:</td>
<td>arabs</td>
<td>0.58</td>
</tr>
<tr>
<td>12:</td>
<td>israeli</td>
<td>0.57</td>
</tr>
<tr>
<td>13:</td>
<td>obama</td>
<td>0.56</td>
</tr>
<tr>
<td>14:</td>
<td>arab</td>
<td>0.56</td>
</tr>
<tr>
<td>15:</td>
<td>iraqi</td>
<td>0.56</td>
</tr>
<tr>
<td>16:</td>
<td>japan</td>
<td>0.56</td>
</tr>
<tr>
<td>17:</td>
<td>sri_lanka</td>
<td>0.56</td>
</tr>
<tr>
<td>18:</td>
<td>america</td>
<td>0.55</td>
</tr>
<tr>
<td>19:</td>
<td>korea</td>
<td>0.55</td>
</tr>
<tr>
<td>20:</td>
<td>cuba</td>
<td>0.55</td>
</tr>
</tbody>
</table>

Most Similar Vectors to ‘iran’ from word2vec 300d embeddings
**Most Similar Vectors to ‘Iran’ from word2vec 300d embeddings**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Word</th>
<th>Cosine Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>Iran</td>
<td>1.00</td>
</tr>
<tr>
<td>2:</td>
<td>Tehran</td>
<td>0.83</td>
</tr>
<tr>
<td>3:</td>
<td>Iranian</td>
<td>0.82</td>
</tr>
<tr>
<td>4:</td>
<td>Islamic_republic</td>
<td>0.81</td>
</tr>
<tr>
<td>5:</td>
<td>Islamic_Republic</td>
<td>0.81</td>
</tr>
<tr>
<td>6:</td>
<td>Iranians</td>
<td>0.78</td>
</tr>
<tr>
<td>7:</td>
<td>Teheran</td>
<td>0.76</td>
</tr>
<tr>
<td>8:</td>
<td>Syria</td>
<td>0.75</td>
</tr>
<tr>
<td>9:</td>
<td>Ahmadinejad</td>
<td>0.69</td>
</tr>
<tr>
<td>10:</td>
<td>Iran</td>
<td>0.67</td>
</tr>
<tr>
<td>11:</td>
<td>Larijani</td>
<td>0.66</td>
</tr>
<tr>
<td>12:</td>
<td>North_Korea</td>
<td>0.64</td>
</tr>
<tr>
<td>13:</td>
<td>Mottaki</td>
<td>0.63</td>
</tr>
<tr>
<td>14:</td>
<td>clerical_regime</td>
<td>0.63</td>
</tr>
<tr>
<td>15:</td>
<td>nuclear_ambitions</td>
<td>0.63</td>
</tr>
<tr>
<td>16:</td>
<td>Khamenei</td>
<td>0.63</td>
</tr>
<tr>
<td>17:</td>
<td>Ayatollah_Khamenei</td>
<td>0.63</td>
</tr>
<tr>
<td>18:</td>
<td>Velayati</td>
<td>0.63</td>
</tr>
<tr>
<td>19:</td>
<td>Ahmedinejad</td>
<td>0.63</td>
</tr>
<tr>
<td>20:</td>
<td>Rafsanjani</td>
<td>0.62</td>
</tr>
</tbody>
</table>
Recurrent Neural Networks for Sequential Data
Network Model for Predicting the Next Word in a Sentence

Sentence: The dog saw the cat on the wall.

- Binary input, "One-hot" encoding
- Hidden unit activations (real-valued)
- Binary target output, "One-hot" encoding
Standard Recurrent Neural Network

\[
h_t = f_W(h_{t-1}, x_t)
\]

\[
h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)
\]

\[
y_t = W_{hy}h_t
\]

Different to a “feedforward” network
Hidden unit at position \( t \) now also
Has input from hidden vector at time \( t-1 \)

Figures from http://cs231n.stanford.edu/slides/2017/
State Computation in a Neural Network

Figures from http://cs231n.stanford.edu/slides/2017/
State Computation in a Neural Network

Key points:
- $f_w$ in each position is the RNN state
- This state is updated recursively from the previous state and the current input

Figures from http://cs231n.stanford.edu/slides/2017/
Key points:
- The $x \rightarrow f$ and $f \rightarrow h$ arrows are weight matrices
- The same weight matrices are (usually) used across the sequence

Figures from http://cs231n.stanford.edu/slides/2017/
Learning the Weights in a Recurrent Network

Learning:  - loss function compares predictions and targets
           - compute gradient (based on error) and update weights

Figures from http://cs231n.stanford.edu/slides/2017/
Example: Prediction of Sequences of Characters

Figures from http://cs231n.stanford.edu/slides/2017/
Simulating Sequences from an RNN

Figures from http://cs231n.stanford.edu/slides/2017/
Simulating Sequences from an RNN

Figures from http://cs231n.stanford.edu/slides/2017/
Simulating Sequences from an RNN

Figures from http://cs231n.stanford.edu/slides/2017/
Output from an RNN Model Trained on Shakespeare

*KING LEAR:*

O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.

*Second Senator:*

They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

*DUKE VINCENTIO: Well, your wit is in the care of side and that.*

---

Examples from “The Unreasonable Effectiveness of Recurrent Neural Networks”, Andrej Karpathy, blog, http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Output from an RNN Model Trained on Cooking Recipes

MMMMMM----- Recipe via Meal-Master (tm) v8.05

Title: CARAMEL CORN GARLIC BEEF
Categories: Soups, Desserts
Yield: 10 Servings

2 tb Parmesan cheese, ground
1/4 ts Ground cloves
--- diced
1 ts Cayenne pepper

Cook it with the batter. Set aside to cool. Remove the peanut oil in a small saucepan and pour into the margarine until they are soft. Stir in a mixer (dough). Add the chestnuts, beaten egg whites, oil, and salt and brown sugar and sugar; stir onto the brownly brown it.

The recipe from an oiled by fried and can. Beans, by Judil Cookbook, Source: Pintore, October, by Chocolates, Breammons of Jozen, Empt.com

MMMMMM

From https://gist.github.com/nylki/1efbaa36635956d35bcc
Different Recurrent Network Models

- **One to one**: Image classification
- **One to many**: Image captioning
- **Many to one**: Sentiment analysis
- **Many to many**: Machine translation
- **Many to many**: Synced sequence (video classification)
Part-of-Speech Tagging
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/
# Universal Tagset (used in NLTK)

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

Example rule: extract adjectives and nouns only