CS 175, Project in Artificial Intelligence

Lecture 2: Basic Concepts in Text Analysis

Padhraic Smyth
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine
<table>
<thead>
<tr>
<th>Week</th>
<th>Monday</th>
<th>Wednesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 8</td>
<td>Lecture: Introduction and course outline</td>
<td>Lecture: Basic concepts in text analysis</td>
</tr>
<tr>
<td>Jan 15</td>
<td>No class (university holiday)</td>
<td>Lecture: Text classification, part 1 Assignment 1 due, 5pm</td>
</tr>
<tr>
<td>Jan 22</td>
<td>Lecture: Text classification, part 2</td>
<td>Lecture: Discussion of class projects Assignment 2 due, 5pm</td>
</tr>
<tr>
<td>Jan 29</td>
<td>Lecture: Neural networks for text, part 1</td>
<td>Lecture: Neural networks for text, part 2 Project proposal due, Friday 6pm</td>
</tr>
<tr>
<td>Feb 5</td>
<td>Office hours (no lecture)</td>
<td>Lecture: Algorithm evaluation methods</td>
</tr>
<tr>
<td>Feb 12</td>
<td>Office hours (no lecture)</td>
<td>Lecture: Unsupervised learning algorithms</td>
</tr>
<tr>
<td>Feb 19</td>
<td>No class (university holiday)</td>
<td>Lecture: Discussion of progress reports Progress report due, Friday 6pm</td>
</tr>
<tr>
<td>Feb 26</td>
<td>Office hours (no lecture)</td>
<td>Office hours (no lecture)</td>
</tr>
<tr>
<td>Mar 5</td>
<td>Office hours (no lecture)</td>
<td>Lecture: Discussion of final reports</td>
</tr>
<tr>
<td>Mar 12</td>
<td>Project Presentations (in class) Upload slides by 4pm</td>
<td>Project Presentations (in class) Upload slides by 4pm</td>
</tr>
<tr>
<td>Mar 19</td>
<td>Final project reports due (day/time TBD)</td>
<td></td>
</tr>
</tbody>
</table>
Announcements

- Friday: office hours from 10:30 to noon (different times this week)
- Thursday: Eric will have office hours in DBH 4222 from 1 to 3
- Monday: no class or office hours (university holiday)
- Assignment 1 due Wednesday next week before class, 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/winter2018/compsci175/home
  – Use this to post questions related to the class
  – Piazza is where we will post announcements, answers to questions, etc

  – If interested in finding other students to join a team, please post relevant information under “forming_teams” category
Assignment 1

Available on the class Web page

Due Wednesday Jan 17th 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
Today’s Lecture

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

• Python examples using NLTK

• Initial Discussion of Projects (if there is time)
  – Examples and ideas
Basic Concepts in Text Representation

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

• **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 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
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!.........’

(Punctuation and white spaces are usually ignored)

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], …}
## How many Words?

\[ N = \text{number of tokens} \]
\[ V = \text{vocabulary} = \text{set of unique tokens} \]
\[ |V| \text{ is the size of the vocabulary} \]

|                      | Tokens N   | Vocabulary | \(|V|\)          |
|----------------------|------------|------------|-----------------|
| Switchboard phone conversations | 2.4 million | 20 thousand |
| Shakespeare          | 884,000    | 31 thousand |
| Google N-grams       | 1 trillion | 13 million  |
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
    - Sequence of tokens -> part of speech tagger -> part of speech tags for tokens
    - Sequence of tokens -> stopword removal -> new sequence of tokens
    - New sequence of tokens -> vocabulary definer -> vocabulary list
    - {New sequence, vocab list} -> counting -> bag of words
    - Bag of words -> machine learning algorithm -> classifier

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

**Label** = basketball

**Class label**
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.
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:
  - 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.
Sentence Detection: Example using a Decision Tree

- Lots of blank lines after me?
  - YES: E-O-S
  - NO: Final punctuation is ?!, or ?:
    - YES: E-O-S
    - NO: Final punctuation is period
      - YES: E-O-S
      - NO: Not E-O-S
        - YES: I am “etc” or other abbreviation
        - NO: Not E-O-S

From https://web.stanford.edu/~jurafsky/slp3/
Speech and Language Processing, 3rd ed, Jurafsky and Martin
Stemming (Optional Step)

- Can reduce all morphological variants of a word to a single term
  - Variants of word such as *fish* and *fisher* and *fishing*
  - Also remove plurals

- Stemming - reduce words to their root form
  - e.g. *fish* – becomes a new term

- Porter stemming algorithm (1980)
  - Relies on a preconstructed suffix list with associated rules
    - e.g. if suffix=IZATION and prefix contains at least one vowel followed by a consonant, replace with suffix=IZE
      - BINARIZATION => BINARIZE
    - Not always desirable: e.g., {university, universal} -> univers (in Porter’s)

- WordNet: dictionary-based approach

- Can help classification performance, but not always
  - May be more useful for information retrieval/search than classification
Example: Defining a Vocabulary

Raw text (a string in 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)
NLTK Tokenizer

from nltk.tokenize import word_tokenize

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

>>> 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
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
```
Example in NLTK (continued)

(see Section 4.1 in Chapter 1 of NLTK Book)

```
>>> vocab1 = set(tokens)
>>> len(vocab1)
16

>>> vocab2 = set(word.lower() for word in tokens)
>>> len(vocab2)
12

>>> vocab3 = set(word.lower() for word in tokens if word.isalpha())
>>> len(vocab3)
10
```
Frequency of Word in English usage

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

Very long "tail" of rare words
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”
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 ...
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.

NLTK StopWord List

```python
>>> from nltk.corpus import stopwords
>>> stopwords.words('english')
['the', 'a', 'an', 'and', 'are', 'as', 'at', 'be', 'been', 'because', 'but', 'by', 'can', 'can', 'cannot', 'could', 'de', 'do', 'does', 'doth', 'during', 'else', 'ever', 'for', 'from', 'get', 'has', 'had', 'have', 'has', 'have', 'he', 'him', 'his', 'how', 'if', 'in', 'into', 'just', 'keep', 'like', 'likely', 'may', 'might', 'more', 'most', 'must', 'my', 'myself', 'nor', 'not', 'of', 'on', 'off', 'on', 'or', 'other', 'our', 'ours', 'ourselves', 'out', 'over', 'past', 'per', 'put', 'quod', 'real', 'so', 'some', 'than', 'their', 'them', 'then', 'there', 'this', 'those', 'to', 'too', 'under', 'up', 'yet', 'you', 'your', 'yours', 'yourself', 'yourselves', 'your', 'you', 'you', 'you']
```
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 – Week 2 and Assignment 2
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

### 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’d 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>
</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
Examples of Analyzing Texts in NLTK and Python
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_ied.)

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
```
Analyzing Different Texts in NLTK
(from Section 1 in Chapter 2 of NLTK Book)

Python input:

```python
import nltk
from nltk.corpus import gutenberg

files = gutenberg.fileids()
```

Python output:

```python
['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...

```python
# import various texts and list them...
from nltk.corpus import gutenberg

# import the tokenize function
from nltk import word_tokenize

define_vocabulary()

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)
```
### Average word length

<table>
<thead>
<tr>
<th>Length</th>
<th>Count</th>
<th>Word Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>79</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>12</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Row</th>
<th>焦虑</th>
<th>词数</th>
<th>文件名</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>25</td>
<td>7344</td>
<td>austen-emma.txt</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>5835</td>
<td>austen-persuasion.txt</td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td>6403</td>
<td>austen-sense.txt</td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>12767</td>
<td>bible-kjv.txt</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>1535</td>
<td>blake-poems.txt</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>3940</td>
<td>bryant-stories.txt</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>1559</td>
<td>burgess-busterbrown.txt</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>2636</td>
<td>carroll-alice.txt</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>8335</td>
<td>chesterton-ball.txt</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>7794</td>
<td>chesterton-brown.txt</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>6349</td>
<td>chesterton-thursday.txt</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>8447</td>
<td>edgeworth-parents.txt</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>17231</td>
<td>melville-moby_dick.txt</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>9021</td>
<td>milton-paradise.txt</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>3032</td>
<td>shakespeare-caesar.txt</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4716</td>
<td>shakespeare-hamlet.txt</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>3464</td>
<td>shakespeare-macbeth.txt</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>12452</td>
<td>whitman-leaves.txt</td>
</tr>
</tbody>
</table>
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)
第13届上海国际电影节开幕...
The 13th Shanghai International Film Festival...

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?
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 is
    • 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 entity and property (location, age) of 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 tends to be 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
1,244 Reviews from our TripAdvisor Community

Read reviews that mention:

- All reviews
- spa pool
- studio restaurant
- main pool
- lobby lounge
- kids club
- car service
- the ritz
- four seasons
- business conference
- beverly hills
- fitness center
- beautiful property
- pool staff
- southern california
- fire pit
- star resort
- spa services
- orange county
- overlooking the ocean
- loft

Traveler rating
- Excellent: 947
- Very good: 163
- Average: 59
- Poor: 32
- Terrible: 25

Traveler type
- Families (263)
- Couples (435)
- Solo (32)
- Business (273)
- Friends (84)

Time of year
- Mar-May (259)
- Jun-Aug (351)
- Sep-Nov (349)
- Dec-Feb (267)

Language
- All languages
- English (1,226)
- German (5)
- Portuguese (5)

Showing 1,226: English reviews

Start your review of Montage Laguna Beach

Click to rate
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
Example 3: Document Classification.....Many Possible Projects

<table>
<thead>
<tr>
<th>Preprocessing Methods</th>
<th>Classification Models</th>
<th>Classification Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bag of Words</td>
<td>Naïve Bayes</td>
<td>Binary Classification</td>
</tr>
<tr>
<td>Stemming</td>
<td>K-nearest neighbor</td>
<td>Multiclass Classification</td>
</tr>
<tr>
<td>Ngrams/Phrases</td>
<td>Logistic regression</td>
<td>Multi-label, Multiclass</td>
</tr>
<tr>
<td>Word embedding</td>
<td>Neural network</td>
<td>Ranking</td>
</tr>
</tbody>
</table>
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
- Standard Logistic Regression
- Cross-Validation Experiments

PHASE 2
- Bag of Phrases (ngrams)
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

PHASE 2

Bag of Phrases (ngrams) → Deep Neural Network

PHASE 3
Announcements

• Friday: office hours from 10:30 to noon (different time this week)

• Thursday: Eric will have office hours in DBH 4222 from 1 to 3

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

• Assignment 1 due Wednesday next week before class, in Dropbox on EEE
<table>
<thead>
<tr>
<th>Week</th>
<th>Monday</th>
<th>Wednesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 8</td>
<td>Lecture: Introduction and course outline</td>
<td>Lecture: Basic concepts in text analysis</td>
</tr>
<tr>
<td>Jan 15</td>
<td>No class (university holiday)</td>
<td>Lecture: Text classification, part 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Assignment 1 due, 5pm</strong></td>
</tr>
<tr>
<td>Jan 22</td>
<td>Lecture: Text classification, part 2</td>
<td>Lecture: Discussion of class projects</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Assignment 2 due, 5pm</strong></td>
</tr>
<tr>
<td>Jan 29</td>
<td>Lecture: Neural networks for text, part 1</td>
<td>Lecture: Neural networks for text, part 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Project proposal due, Friday 6pm</strong></td>
</tr>
<tr>
<td>Feb 5</td>
<td>Office hours (no lecture)</td>
<td>Lecture: Algorithm evaluation methods</td>
</tr>
<tr>
<td>Feb 12</td>
<td>Office hours (no lecture)</td>
<td>Lecture: Unsupervised learning algorithms</td>
</tr>
<tr>
<td>Feb 19</td>
<td>No class (university holiday)</td>
<td>Lecture: Discussion of progress reports</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Progress report due, Friday 6pm</strong></td>
</tr>
<tr>
<td>Feb 26</td>
<td>Office hours (no lecture)</td>
<td>Office hours (no lecture)</td>
</tr>
<tr>
<td>Mar 5</td>
<td>Office hours (no lecture)</td>
<td>Lecture: Discussion of final reports</td>
</tr>
<tr>
<td>Mar 12</td>
<td>Project Presentations (in class)</td>
<td>Project Presentations (in class)</td>
</tr>
<tr>
<td></td>
<td>Upload slides by 4pm</td>
<td>Upload slides by 4pm</td>
</tr>
<tr>
<td>Mar 19</td>
<td><strong>Final project reports due (day/time TBD)</strong></td>
<td></td>
</tr>
</tbody>
</table>