CSC 5930/9010: Text Mining
Lab 3

Dr. Paula Matuszek
Paula.Matuszek@villanova.edu
Paula.Matuszek@gmail.com
(610) 647-9789
Goals

- Goals for this lab are:
  - More Python
  - Run a naive Bayes classifier
  - Evaluate the results
• The Natural Language Processing with Python book covers a lot of Python, interspersed with a lot of NLP.

• We are mostly interesting in the parts relevant to text mining, so we are skipping a lot.

• Unfortunately that means we skip a lot of the Python, some of which we might want.
(Very) Brief Python Overview

• Borrowing a presentation:  [www.cis.upenn.edu/~matuszek/cit591-2012/Lectures/01-python.ppt](http://www.cis.upenn.edu/~matuszek/cit591-2012/Lectures/01-python.ppt)

• [http://www.cis.upenn.edu/~matuszek/Concise Guides/Concise Python.html](http://www.cis.upenn.edu/~matuszek/Concise Guides/Concise Python.html)

• To use the NLTK and do the homework assignments, you don’t actually need a lot of Python. Just plunge in.

• If you need more (for your project, for instance), there is a good tutorial at [http://docs.python.org/tutorial/](http://docs.python.org/tutorial/)

• You can also work through more of the NLTK book.
One More Data Type

- Python has a data type called a dictionary
- Sometimes called an associative array
- A dictionary is a set of key, value pairs
  - It is indexed by the key; rather than getting the $n$th element of the set, you get the key$^\text{th}$ element.
  - Each key within one dictionary must be unique
  - The value can be a list, so more than one thing can be associated with a given key
  - It is a set -- it is unordered.
- This data type is often useful in text mining
Dictionary Examples

- Empty dictionary: {}
- Dictionary of language features:
  
  ```json
  {'length': 27506,
   'language': 'English',
   'avgSent': 'R'}
  ```

- Dictionary of students
  
  ```json
  {'Matuszek': ['CS', 'MS', 'Paula'],
   'Smith': ['IS', 'BS', 'John'],
   'Gupta': ['CS', 'BS', 'Rajani']}
  ```
Getting Your Documents In

• First step is to get documents into your program.

• Hopefully you have all done this.

• You can give complete paths. If you’re working in Windows, either use / instead of \ or use \\ (because \ is the escape character)

• At this point you have one long string.
Breaking It Down

- Most of our operations expect a list of tokens, not a single string.
- NLTK has a decent default tokenizer
- We might also want to do things like stem it.
- Hopefully you have all done this as well.
The code

- Let’s look through some code.
- Sep11In.py
Some nltk methods expect an NLTK text object as input, so we might want to turn our tokens into one.

\[
\text{mytext} = \text{nltk.Text(tokens_amtrak)}
\]

At this point if you check the type you will discover that text is of type `<class 'nltk.text.Text'>`.

Whether you need to do this depends on the inputs that the NLTK method you are using wants.
Classifying with NLTK

The basic process for classifying documents with NLTK is:

- Define our problem. Classify ?? into ??.
- Choose a feature set. These are the features we think might be useful in classifying.
- Choose training and test documents. These will include the features we choose and the class for each document.
- Run a classifier
- Look at the results.

We are working through part of chapter 6 of the NLTK book.
Classifying

• Defining our problem: we are going to classify names into male and female

• Defining our feature set: We will start simple. Can we classify based on the last letter of the name?

• Choosing training and test documents. NLTK corpus includes a names corpus with two files: male.txt and female.txt. We will choose randomly from those two files.
Feature Set

- NLTK classifiers expect a training set of cases, each of which is a dictionary of features followed by a class
- \((\{'lastletter\': 'm'\}, 'male')\)
  - One feature, lastletter, with the value m, and a class male
- \((\{'length\': 4, 'lastletter': 'b', 'firstletter': 'R'\}, 'male')\)
  - Three features: length, lastletter, firstletter, and a class male
Creating the Feature Set

names = [(name, 'male') for name in names.words('male.txt')] + [(name, 'female') for name in names.words('female.txt')]

random.shuffle(names)

def gender_features(name):
    features = {}
    features['lastletter'] = name[-1]
    return features

featuresets = [(gender_features(n), g) for (n,g) in names]
Training and Test Sets

- We have about 7500 names altogether
- A reasonable test set is 500, chosen randomly
- Our featureset is already randomized, since we randomized the names.
- So we just pick off the first 500 as our test set and the rest become our training set
- \( \text{test\_set}, \text{train\_set} = \text{featuresets}[:500], \text{featuresets}[500:] \)
Running a Classifier

- NLTK has a number of different classifiers; tonight we will use the Naive Bayes classifier.
- We create and name a trained classifier
  \[
  \text{myclassifier} = \text{nltk.NaiveBayesClassifier.train(train\_set)}
  \]
- Now we have a trained model, called myclassifier. We can use it in various classify methods.
  \[
  \text{nltk.classify.accuracy(myclassifier, test\_set)}
  \]
  \[
  \text{classifier.show\_most\_informative\_features(10)}
  \]
More Code

- So what does this all look like as code?
- Sep11Classify.py
More with our Classifier

• We have already seen two evaluations
  • accuracy: `nltk.classify.accuracy(classifier, test_set)`
  • most informative features:
    `classifier.show_most_informative_features(10)`

• And we have used it to classify one name.
  • `classifier.classify(gender_features('Neo'))`

• Let’s classify some more
Next Week:

- More evaluation methods in NLTK
- More Classifiers