NLP Lab Session
Week 14, December 9, 2015
Classification for the Final Projects

Getting Started

In this lab, we will be doing some work in the Python IDLE window and also running weka. All of the programs and data are zipped together in one zip file on Blackboard:

LabExamplesWeek14_FinalProjectClassifiers.zip

Download this zip file to your NLP class folder in the lab and unzip it there.

Classifiers

In the NLTK, we have been using the Naïve Bayes classification algorithm, which has reasonable performance on many smaller classification problems. For your final project, you may continue to work in NLTK, using Naïve Bayes for the dataset that you choose. But if you find that it takes too long to train models using a large number of examples, or if NLTK runs out of memory, one solution is to use NLTK to prepare the features and then use an outside classifier.

On text classification problems, it is also often the case that the Support Vector Machines (SVM) algorithms or the MaxEnt (aka logistic regression) algorithms have the best classification performance, particularly for problems with large numbers of features and large numbers of training examples. NLTK does also have a MaxEnt classifier, in addition to Naïve Bayes and Decision Tree classifiers, but I believe that classifier is being phased out in favor of using the one in SciKit Learn.

In this lab, we will explore how to prepare the features in NLTK and then use one of the external classifier packages to do the actual classification.

One choice is to use the Weka software package, which is an open software application that collects together many data mining algorithms, including classification algorithms. Weka is particularly useful for classification experiments, for example, to compare different types of features (and less useful for creating a classifier for unlabeled data). Weka is written in Java and is used through a GUI.

Another choice would be to use the Sci-Kit Learn software package, which is an open source software application that also collects together many data mining algorithms and processes. Sci-Kit Learn is written as a collection of Python packages and is used through the command line, either as a sequence of commands in an IDLE or Python interpreter window, or collected into a Python program. It is imported as sklearn and is also useful for classification experiments. (You are also supposed to be able to use sklearn from within NLTK by loading sklearn as nltk packages, but I didn't explore that option.) However, using Sci-Kit Learn directly does require more downloading of software and uses NLTK 3.x.

Both Weka and Sci-Kit Learn have Text to Feature Vector software that will automatically tokenize text and possibly apply other transformations such as removing stop words and then generate a feature vector. But this is **not** what you are supposed to do for projects!! Instead, we will be using what we know about NLP to process text and produce our own feature vectors. So for both Weka and Sci-Kit Learn, I am going to demonstrate how to write a .csv file of feature sets, i.e. each instance of the classification problem is represented as a set of features and a class label for training.

Then we will show how read those .csv files into either Weka to perform experiments with other classifiers such as SVM or MaxEnt /logistic regression.

Creating data for classification

To demonstrate data preparation, we will use the NLTK to create feature sets as usual. In this lab, we will just use the NLTK movie reviews. You'll be using other data for your final projects, but the format of the feature sets should be the same.

In order to demonstrate this process, let's load the movie reviews in NLTK.

```
>>> from nltk.corpus import movie_reviews
>>> import random

>>> documents = [(list(movie_reviews.words(fileid)), category)
        for category in movie_reviews.categories()
        for fileid in movie_reviews.fileids(category)]
```

Since the documents are in order by label, we mix them up for later separation into training and test sets.

```
>>> random.shuffle(documents)
```

We need to define the set of words that will be used for features. This is essentially all the words in the entire document collection, except that we will limit it to the 1000 most frequent words.

In previous labs, we used the convenient function movie_reviews.words() which already gets all the words in the movie review corpus and then put them into a frequency dictionary.

```
    all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
```

Instead, let's define all the words by iterating over the documents that we just created. This should also work for all the final project datasets.

```
>>> all_words = nltk.FreqDist(w.lower() for (words,tag) in documents for w in words)
```

Then we take as many word features as we have room and time for.

```
>>> word_features = all_words.keys()[:1000]

[ Or for NLTK 3.x, use the most_common function which returns (word,frequency) pairs
of the most frequent words
>>> word_items = all_words.most_common(1500)
>>> word_features = [word for (word, freq) in word_items]   # just the words
]
```

We give the same feature definitions as before for all words, except that we have to make some restrictions for the .csv file and for weka. For the .csv file, we'll remove any word that contains a comma. For weka, we'll remove words that contain either a single quote/apostrophe or a double quote. Use a shorter prefix for the vocabulary feature names.

```
>>> def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        if ("'" not in word) and ("," not in word) and ('"' not in word):
            features['V_%s' % word] = (word in document_words)
    return features
```

Define the feature sets for the documents.

```
>>> featuresets = [(document_features(d), c) for (d,c) in documents]
```

Writing Featuresets to a .csv file

For this part, I have written a function that will write the feature sets to a file that can be used for classification testing in Weka or Sci-Kit Learn. One file format that Weka can use is a csv (comma separated values) file where the first line should contain the names of the features (which Weka calls attributes) separated by commas and the remaining lines have one line per training example, with the feature values separated by commas. Note that you should not allow any feature name or value containing quotes, apostrophes, or commas. Also, Weka treats the class label (for example, 'pos' or 'neg') as one of the features, and it is customarily the last one.

Here is a function definition that takes featuresets and a path to where you want to write the csv file, and converts the featuresets to be a weka input file in csv format. This file format will also be suitable to read for using Sci-Kit Learn.

```
>>> def writeFeatureSets(featuresets, outpath):
    # take featuresets defined in the nltk and convert them to weka input csv file
    #    and write the file to the outpath location
    # open outpath for writing - it should include the name of the csv file
    f = open(outpath, 'w')
    # get the feature names from the feature dictionary in the first featureset
```

```
    featurenames = featuresets[0][0].keys()
    # create the first line of the file as comma separated feature names
    #    with the word class as the last feature name
    featurenameline = ''
    for featurename in featurenames:
        featurenameline += featurename + ','
    featurenameline += 'class'
    # write this as the first line in the csv file
    f.write(featurenameline)
    f.write('\n')
    # convert each feature set to a line in the file
    # with comma separated feature values, for booleans just write the words true and false
    for featureset in featuresets:
        featureline = ''
        for key in featurenames:
            featureline += str(featureset[0][key]) + ','
        featureline += featureset[1]
        # write each feature set values to the file
        f.write(featureline)
        f.write('\n')
    f.close()
```

Now we need to make a path to the output file (which will be a .csv file). Include the name of the file and the file extension .csv. (On my Mac, an example of a path is outpath = "/Users/njmccrac1/AAAdocs/NLPspring2015/labs/LabExamplesWeek13_FinalProjects_Spring2015/weka/movies_words.csv"

```
>>> outpath =   <put your path here for the lab>
```

Now we can call the function:

```
>>> writeFeatureSets (featuresets, outpath)
```

And we can go to the file, open it either in Excel or in NotePad++ or some other editor to see the contents.

Now we can classify in Weka or Sci-Kit Learn.
If you have not used Weka before, go to the Weka document for instructions.
If you want to use Sci-Kit Learn, use anaconda to install everything you need. Note that this will automatically upgrade your path to use the anaconda separate installation of NLTK 3.x. Use the program

```
        python run_sklearn_model_performance.py <path to feature file>
```

and choose one of the classifiers in comments to perform the classification.

There is no lab exercise for this week, but fill in a discussion with which option you have chosen for the final project. If you are doing the classification option 2, alse give which dataset you plan to use. If you are planning to work in a group, only one post needs to be made per group, and it should contain all the team member names.