NLP Lab Session, Week 12, November 16, 2017
Reviewing Final Project Options
Reading Final Project Data for Classification
Using External Classifiers Weka or SciKitLearn

Getting Started

Download the file FinalProjectData.zip from Blackboard and not only unzip the outer level, but also the inner level for any data that you are interested in reviewing.

In this lab, we will be reviewing data and programs for (two of) the three options for final projects:

- design a plan understanding module in LUIS
- running some stand-alone python programs that demonstrate how to work with the three datasets available for the classification option of the Final Projects, which I have downloaded for you from the web. The programs and data are zipped together and placed on blackboard. In addition, there is a separate folder with programs to process features for a classification problem in NLTK, save to a file, and use that file in an external classifier.
- description of the programming project to compute lexical chains. For this you are responsible to assemble 2 small sets of documents for testing.

1. Design a plan understanding module in LUIS (half a chatbot)

There is no further information at this time. Please start by thinking about the chatbot user design and by watching the Microsoft LUIS tutorial videos to get started. Note that all syr.edu email accounts should be eligible to sign up for a LUIS account.

2. Classification task: looking at and reading the datasets provided

Each folder contains a python program that we will run to process the data in the folder.

- First look at the raw data.
- Run the program and observe some of the tokenized data.

Notes:

1. The limitations of using the classifier in NLTK is that you may be limited by the number of documents and the number of features that you can use in order for the classifier to run in a reasonable time and memory space.
   a. each program allows you to limit the number of documents (emails, phrases or tweets)
   b. you can limit the number of features in the python program

2. In the Enron email dataset, if you want to run the POS tagger, you must first separate the text into sentences, using NLTK's sentence tokenizer, for example, this gets a list of sentences, each of which can then be tokenized and run in the POS tagger.

```
sentences = nltk.tokenize.sent_tokenize(text)
```

3. Sentiment Lexicons: These lexicons and the programs are found in the kaggle folder. To read subjectivity words, I added a function to read all three types of positive, negative, and neutral words. If you want to try using positive and negative sentiment words from the LIWC lexicon, I have given a python program that will read them in

```
sentiment_read_LIWC_pos_neg_words.py
```

External 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.) Sci-Kit Learn should already be installed in Anaconda.

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.

Writing Featuresets to a .csv file

Before using an external classifier package, you will first write a program to generate featuresets as usual with NLTK, for example, using one of the template programs for the spam, phrase sentiment or twitter sentiment classification problems. Then you will include a function definition for writeFeatureSets() in that program and call it to write your featuresets to a file. And then file can be read with Weka or sklearn in order do the cross-validation classification and report performance.

For this part, I have written the 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.

The file save_features.py has two parts:

- the function definition of writeFeatureSets()
- a main program demonstrating how to use the writeFeatureSets() function by creating simple BOW feature sets on the movie reviews in NLTK and then writing those featuresets to a file.
    - the program also contains the feature definition function needed to create the BOW featuresets.

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 the program

python run_sklearn_model_performance.py <path to feature file>

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

3. Programming Option

We will briefly review that programming project document to discuss this option.

For the lab exercise, fill in a discussion with which option you have chosen for the final project. If you are doing the classification option, also give which of the provided datasets you plan to use or if you plan to find your own dataset and use it. 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.