NLP Final Project

The final project will be a classification task, where you will develop features for the task and demonstrate that you can carry out experiments that show which sets of features are the best for that data.

Three datasets are provided for your choice. You may also propose your own data. While the task is designed to be appropriate for a single person, you may also work in groups of 2-3 people. For each person that is added to the group, plan on increasing the tasks by about 50%.

Processing and Classification: Three Datasets

For this project, you should choose a dataset to work on: the email spam data, the Kaggle movie review data, the SemEval Twitter data or a dataset of your choosing (with permission from instructor).

1. Detection of SPAM in email

The first dataset is one produced for detecting Spam emails from the Enron public email corpus. In addition to some small numbers of Spam already in the corpus, additional spam emails were introduced into each user's email stream in order to have a sufficient number of spam examples to train a classifier. The non-Spam emails are labeled "ham". (See this paper for details: http://www.aueb.gr/users/ion/docs/ceas2006_paper.pdf) The dataset that we have was gleaned from their web site at http://www.aueb.gr/users/ion/data/enron-spam/.

Although there are 3 large directories of both Spam and Ham emails, only the first one is used here with 3,672 regular emails in the "ham" folder, and 1,500 emails in the "spam" folder.

2. Kaggle competition movie review phrase data, labeled for sentiment

This dataset was produced for the Kaggle competition, described here: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews , and which uses data from the sentiment analysis by Socher et al, detailed at this web site: http://nlp.stanford.edu/sentiment/. The data was taken from the original Pang and Lee movie review corpus based on reviews from the Rotten Tomatoes web site. Socher's group used crowd-sourcing to manually annotate all the subphrases of sentences with a sentiment label ranging over: "negative", "somewhat negative", "neutral", "somewhat positive", "positive".

Although the actual Kaggle competition is over, the data is still available at https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data. We are going to use the training data "train.tsv", and some test data is also available "test.tsv". There appear to be 156,060 phrases in the training data file, and one of the challenges will be to choose an appropriate subset for processing and training.

3. SemEval shared task Twitter data, labeled for sentiment

The Semantic Evaluation conference, SemEval, has recently added sentiment detection in Twitter messages and other social media genres to its shared tasks. For example, in 2014, it ran as Task 9, sub-task B, Message Polarity Classification: http://alt.qcri.org/semeval2014/task9/. The data was manually labeled and each dataset is available as a Twitter id paired with the manual label. There are 5 labels used for each message: "positive", "negative", "objective", "neutral", "objective OR neutral".

To actually get the tweets, there is a script you can run to get them from Twitter itself. If a Twitter user has retracted their tweet, then Twitter will no longer send it out and it is marked as "Not Available".

The dataset given here was collected in the Spring 2014 from Twitter. You are given access to this data through our learning management system, and it should not be made publically available as it is subject to the Twitter terms of service.

The Text Classification Tasks

Step 1: For your choice of dataset, you will first process the text, tokenize it and choose whether to do further pre-processing or filtering. If you do some pre-processing or filtering, then using the text with and without it can be one of your experiments.

For each dataset, there is a program template that reads the data. You should run the program on your data of choice and investigate some of the data in order to choose pre-processing or filtering. If you choose your own dataset, then you should write a similar program to read the text data and the labels.

Step 2: The second step is to produce the features in the notation of the NLTK. For this you should write feature functions in Python as we have been doing in lab. You should start with the "bag-of-words" features where you collect all the words in the corpus and select some number of most frequent words to be the word features.

Now use the NLTK Naïve Bayes classifier to train and test a classifier on your feature sets. You should use **cross-validation to obtain precision, recall and F-measure scores.** Or you can choose to produce the features as a csv file and use Weka or Sci-Kit Learn to train and test a classifier, using cross-validation scores.

Step 3: Experiments

A. For a base level completion of experiments, carry out at least several experiments where you use two different sets of features and compare the results. For example, you may take the unigram word features as a baseline and see if the features you designed improve the accuracy of the classification. Here are some of the types of experiments that we have done so far:

- filter by stopwords or other pre-processing methods
- representing negation (if using twitter data, note the difference in tokenization)
- using a sentiment lexicon with scores or counts: Subjectivity
- different sizes of vocabularies

- POS tag features

You must define at least one "new" feature function not given in class. Also you should try to combine some of the earlier features, e.g. to use unigrams, bigrams, POS tag counts, and sentiment word counts all in one feature set. Examples of new features:

- Use the LIWC sentiment lexicon
- combine the use of sentiment lexicons
- use a different representation of negation, for example, carrying the scope of the negation work over to the next punctuation

B. Choose an additional, more advanced type of task from this list, or propose your own

- using classifiers (e.g., Weka, Sci Kit Learn, etc.) with features produced in NLTK
- using an additional type of lexicon besides Subjectivity or LIWC
- in addition to using cross-validation on the training set, train the classifier on the entire training set and test it on a separately available test set (only the SemEval data has these)
    - note that you must save the vocabulary from the training set and use the same for creating feature sets for the test data
- implement additional features
    - in the email dataset, use word frequency or tfidf scores as the values of the word features, instead of Boolean values
    - use POS tagging from the ARK on Twitter
    - twitter emoticons or other features based on internet usage or informal text, such as repeated letters, or all caps words

Step 4: Write a report that describes the data processing, the features and the classification experiment(s). As one (or more) of your experiments, you are to compare results from different classifier algorithms (e.g., Weka, Sci-Kit Learn, etc.).

What to Hand In

Each person should hand in a report with the description of all that you did and the discussion of the results. Also submit any python programs that you write, particularly to show additional features that you implement, or the annotation files that you did.

If you worked in a group, the report should include which tasks each person worked on. Each person in the group should submit the (same) report.