NLP Lab Session
Week 13, November 20, 2013
Text Processing and Twitter Sentiment for the Final Projects

Getting Started

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

LabExamplesWeek13FinalProject.zip

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

General Text Processing

Check the file Labweek13.textprocess.py.

Classifying Twitter Sentiment

There have been quite a few research papers in recent years that discuss various aspects of finding Twitter Sentiment. I have chosen to follow one by Koulompis, Wilson and Johanna Moore "Twitter Sentiment Analysis: The Good the bad and the OMG!", in ICWSM (International Conference on Weblogs and Social Media) in 2011. Note that Theresa Wilson is one of the researchers from the Subjectivity Lexicon project. This paper is available on-line and also in the Content -> Final Project Resources -> Twitter Sentiment Classification folder on Blackboard, along with two other papers on Twitter Sentiment.

For training data, we are going to use the Sentiment 140 Emoticon data set, available at http://help.sentiment140.com/, where you can click on the link "For Academics" to read about the data and their process. The data set was collected by searching Twitter for positive and negative emoticons. The emoticons were removed from the tweets, and each tweet was labeled positive (4) or negative (0) according to its emoticon. The data set has 1,600,000 tweets. This dataset is good for experiments, but note that in a practical situation where we want to detect sentiment in tweets, we would want to have training data for three polarities: positive, negative and neutral. Also, since this training set has emoticons removed, they can't be used for features.

In order to randomly select a smaller training set, I wrote a program called sentiment__classify_select_training_data.py that selects 8,000 tweets of each polarity and saves them in a .csv file called training.16000.mixed.csv. I have included the program for your information and use if you wish to have larger or smaller training sets. (We can look at some examples in this file, either using Excel or NotePad.) Note that to run the program, you should use easy_install to install the python package csv.

First, if we don't already have easy_install, we should install it. Next we open a terminal (command prompt) window and type

% easy_install csv

This installs the python csv package in its folder of external packages.

Next, I have written a baseline sentiment classifier program that uses NLTK to train a classifier using word features and subjectivity features from the training set. The baseline classifier uses a twitter tokenizer from Tweet Motif and does some pruning to remove non-ascii characters, tokens of length 1 and tokens on a tweet stopword list. In order to run this program, open a terminal window (or command prompt window) and navigate to the folder where you put all the programs from this week's lab examples. Now run the classify train model program:

$ python sentiment__classify_train_model.py
(or whatever command you need to use on a PC)

This will take a little time to make the feature sets and train the model.

Now to work on this classification task for the final project, you can extend this program in several ways:

- Work on preprocessing to improve the tokens
- Work on features to add useful ones
- Experiment with different classifiers in weka

For ideas on preprocessing and features, we can look at a summary of what the Wilson et al paper did. Note that the baseline program already uses the three subjectivity lists as they did, but that you could experiment with using the LIWC or ANEW lists instead of, or in addition to, the subjectivity lexicon.

Here are the main points for successful preprocessing and features from the Wilson paper:

Preprocessing:

- Tokenize emoticons, hashtags, mentions and URLs and replace them by their token type (HAPPY, SAD, URL, etc.)
- Expand abbreviations to their meanings, brb -> "be right back"
- Normalize character repetitions to two, "happyyyyy" -> "happyy"
- Note intensifiers such as all caps, "I LOVE this show"

Features:

- Vocabulary: removed stopwords, replaced words that either directly preceded or directly succeeded a negation word with the word with NOT, "happy" -> "NOT_happy" and keep only the top occurring 1000 words

- Subjectivity Lexicon: three features denoting the presence or absence of positive, neutral or negative emotion words.
- (Part-of-speech features were tried, but didn't add to the accuracy.)
- Micro-blogging features: emoticons, presence of intensifiers (using the Internet Lingo Dictionary and other internet slang dictionaries).

There are other ideas in the other two Twitter Sentiment papers that I placed in the Blackboard folder.

Note that 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_pos_neg_words.py in the zip file.