**Final Project**: Sentiment Analysis

Issued: 11/29/2018

Due: 12/13/2018 (hard deadline: 12/15/2018)

Total points: 100

The goal of this assignment is to give you hands on experience with a basic NLP pipeline for a real application: sentiment analysis for movie reviews. You have to implement a sentiment analyzer that automatically classifies input data as either positive or negative. Since we provide training data, it makes sense to use supervised machine learning to build the classifier.

Approach:

1. First, you should produce one or more baselines (i.e., simple, first-stab approach systems that you are fairly confident will produce a measurable result). Baselines should be as simple as possible and should prove the feasibility of your plan. However, as for baselines, the system performance does not need to be very high.
2. You should conduct a series of experiments which may improve the performance. Try to learn from your results and revise your experiments as you go. If you methodically follow a line of reasoning in your experiments, but do not ultimately have a huge improvement over your baseline system, that is actually OK (provided you justify it). Close-to-perfect results are nice, but we are also looking for good methodology in your experimentation.
3. The idea is to start with a good baseline system and find new good/informative features to improve it. So, at the end, we would like to see a performant system. However, as said before, we are more interested in the text analytics process / methodology you use to build such a system.

For this project you will experiment with Naïve Bayes. The performance metric to consider is accuracy.

**Dataset:**

The dataset used in this project is the [polarity dataset v2.0](http://www.cs.cornell.edu/people/pabo/movie-review-data/review_polarity.tar.gz) (3.0Mb) (includes [README v2.0](http://www.cs.cornell.edu/people/pabo/movie-review-data/poldata.README.2.0.txt)): 1000 positive and 1000 negative movie reviews. Introduced in Pang/Lee ACL 2004 and released June 2004, it is drawn from an archive of the rec.arts.movies.reviews newsgroup hosted at [IMDB](http://reviews.imdb.com/Reviews).

You have to download the dataset and read the README file. Familiarize yourself with the dataset and see how it is structured so that you prepare it for classification. Divide the corpus with a ratio 90:10 split (200 review files are used as test – i.e., files that start with cv9) – keep an even distribution of positive and negative labels.

Project questions: Here are some questions intended to guide you:

1. Problem1. In building the baselines, what shallow text features make sense for the task as a first-stab approach? (e.g., a language modeling approach is a standard way to tackle this problem, i.e., a bag-of-words approach). Experiment with the following baseline models (note that you have to run a POS tagger for models M3):

|  |  |  |
| --- | --- | --- |
| Model | Feature | Accuracy |
| M1 | Unigrams (absence/presence) |  |
| M2 | Unigrams with frequency count |  |
| M3 | Unigrams (only adjectives/adverbs) |  |
| M4 | Unigrams (sublinear tf-idf) |  |
| M5 | Bigrams (absence/presence) |  |

As we’ve seen before with other models, you have to decide if you need to clean your data before you send it to the classifier. For this, you have to look carefully at your reviews. As you can see, your data has already been tokenized and downcased.

Besides stop word removal (which here you should do only for model M4), do you need to do any more preprocessing before starting to build the application?

After this, you have to represent the text to identify the features you test under each model. Run the classifier for each baseline feature and report the performance. What is the best model and which model is the least performant? How do you interpret the differences in performance across these models (i.e., why do you think one model is better than another)? For the unigram models M1-M4, list the top 5 most informative words identified by the classifier. What do you notice here? (compare these lists).

Note: For model M4 use sublinear tf-idf. Here, besides removing stop words, you also have to remove those vocabulary words that occurred in less than 5 documents (i.e., files) and those that occurred in more than 80% of documents.

1. Problem2. Redo Problem1 but with stemming (use Porter’s stemmer). Is this worth doing? Compare the models’ performance with those at Problem1.
2. Problem 3. Here you have to experiment with the Pos/Neg ratio (use the EmoLex lexicon provided). Compare the performance of this model with those you got for Problems 1 and 2.

Note: The format and the terms of use of the EmoLex lexicon can be found here: <http://saifmohammad.com/WebDocs/README-NRC-Lex.txt>

1. Problem 4. How would you improve over the models you have experimented with so far? Meaning, what text features are most beneficial to the task of sentiment analysis? -- i.e., if you had to do it again, what would you change? For example, if you had to solve the problem in a different (and hopefully more efficient way), which features would you choose? Write 1-2 short paragraphs about the features you might want to try for this problem.

Note: Do not give example of features from the lecture. Study the data (i.e., the movie reviews) in detail and see what is needed there that the features you experimented with so far can not capture.

**Project Deliverables:**

This is an independent project (i.e., you must work alone although you can engage in design discussions with fellow colleagues).

This is what you have to upload into canvas:

1) all code you wrote to process the data (i.e., preprocessing / data cleaning code, etc.) plus a Readme file where you explain what the code does and how to run it.

2) You have to answer the questions at Problems 1 - 4 above and explain all the steps you took to build the system (place these in answers.pdf).