CS1573, Spring 2011
Project 1
Document Classification: Sentiment Analysis

1 Introduction

This project in text classifcation will focus on features, experimental set up and evaluation, and give you experience applying an existing ML package to a text classification task.

The task is document-level sentiment analysis, specifically recognizing whether on-line movie reviews are positive or negative.

Your task will be to create feature (attribute) files representing the documents in the dataset, to be input to the Weka Machine Learning Toolkit. Weka will run a 10-fold cross validation experiment on the data, and output evaluation statistics. You will experiment with different feature sets, perform ablation studies, and submit your feature files and a report.

This paper carries out comparative experiments, and would be useful. Don't worry about details of the actual subsumption hierarchy and linguistic features. We will discuss aspects of this paper in class.

```
@InProceedings{riloff-patwardhan-wiebe:2006:EMNLP,
  author    = {Riloff, Ellen  and  Patwardhan, Siddharth  and  Wiebe,
  Janyce},
  title     = {Feature Subsumption for Opinion Analysis},
  booktitle = {Proceedings of the 2006 Conference on Empirical Methods
  in Natural Language Processing},
  year      = {2006},
  address   = {Sydney, Australia},
  pages     = {440--448},
  url       = {http://www.aclweb.org/anthology/W/W06/W06-1652.pdf}
}
```

2 Data

Download **polarity dataset v2.0** from

`http://www.cs.cornell.edu/people/pabo/movie-review-data/`

This data consists of 1000 positive and 1000 negative movie reviews from the Internet Movie Database (IMDb) archive. The positive and negative classes are derived from author ratings expressed in stars or numerical values. If you are interested, the particular criteria to classify posts as positive or negative are found in the readme file accompanying the data.

Important Note: you should not look at the text of any of the documents, in case you notice a frequent feature this particular dataset happens to have. If you want to get ideas for features,

it is acceptable to look at similar data distinct from the experiment corpus. This is called *development* data. For example, download some current reviews from rottentomatoes.com You can be sure there is no overlap between these documents and those in the experiment corpus, since the experiment corpus was created in 2004.

3 Features and Feature Selection

In the Segaran text, the method for mapping an input item to its features is applied inside the training method (see p. 121). Instead, you will write code to create a file in which items are represented as feature vectors. That file will then be input to Weka, for training and evaluation.

You should start with the simple bag of words features, as in Segaran. Then, you should experiment with adding other features as well.

You could experiment with several possibilities, for example:

- things: words, parts of speech, hits of positive words, hits of negative words, punctuation, hits of words in a list you compile yourself by looking at development data
- counts of things
- sequences of things (could be mixtures of types)
- occurrences within a window (could be mixtures of types)
- positions of things in the document
- combinations & variations of the above

An additional possibility is to experiment with feature selection strategies.
You will be provided with two wordlist files:

- *stopwords_mysql.txt* contains a list of stopwords used by MySQL when evaluating full-text queries. Stopwords are frequently occuring words that are usually filtered out of queries in order to improve performance. Filtering out stopwords may help to improve your results (then again, it may not). Each stopword is on a separate line. **Feel free to modify this list.**
- A second file, "subjclueslen1_simplified.txt", contains a list of single-word subjectivity clues. A subjectivity clue is a word or phrase that is frequently used when a person is expressing an internal state, such as a belief, opinion, or sentiment. The kind of subjective expression in which each word typically appears is also given ("positive", "negative", or "neutral"). Incorporating subjectivity into some of your features may also improve performance (but again, maybe not).

The NLTK toolkit is a set of NLP tools widely used in classes. The support on the website is good and the toolkit isn't difficult to use. You may want to run one or more of the tools on the input documents to help you create your features.

4 Partnering

There are several aspects to the project. Meet with your partner early to split up your tasks and create a schedule. You will both need to stay on schedule so you do not hold each other up.

Also, get the basics working first, and then work on extensions only if there is time. Don't feel you need to try every variation or every resource.

5 Grading

Grading:

- Fundamentals, the baseline bag of words approach: 35%
- At least three other types of features: You will be graded on how interesting your features are and your motivation for why you think they are promising. Note that it is difficult to beat bag of words on this dataset, so your grade will not suffer if your features do not improve on its performance (Though improvements will be awarded): 40%
- Your report: 25%

6 Report

Your report should describe, motivate, and illustrate your new features.

It should report results of the baseline bag of words approach, the results of your full system, and the results of ablation studies, in which the system is avaluated as features are systematically added or removed (your choice). All results should be of 10-fold cross validation experiments. There is no need to carry out significance tests.

Present your results in readable tables with informative captions, and discuss your results in the text.

Describe your ideas for what you would try next, if you had more time to work on the project.

7 Submission Instructions

To submit the project, please email the TA an archive containing all of your source code, feature files, report and README file before the deadline. The README file should specify how to execute your code.

Please send the archive to alexander.p.conrad@gmail.com.

8 Tools

8.1 Weka

WEKA is a free machine learning software package written in Java with following capabilities: data preprocessing, classification, clustering, regression, visualization, and feature selection.

8.1.1 Installing WEKA

The project homepage is located at

`http://www.cs.waikato.ac.nz/ml/weka/.`

Follow the download link and choose the right version depending on your platform. For the Windows platform, double-clicking on the self-extracting executable will install WEKA toolkit on your system. For the Mac and Linux platforms, you just need to extract the downloaded zip file. You need to have Java VM (Java 1.5 or later) installed on your system in order to run the WEKA toolkit.

8.1.2 Using WEKA

There are three ways to interact with the WEKA toolkit: (1) the graphical user interface (GUI), (2) the command line interface, and (3) the API. For the first project and getting used to WEKA toolkit, the graphical user interface is quite sufficient. For advanced manipulation, you might need to use the API or the command line interface. In this document, you will find how to use the GUI to conduct a 10-fold crossvalidation experiment for classification on a sample dataset. For a tutorial on accessing the API, see the following wiki entry

`http://weka.wikispaces.com/Use+Weka+in+your+Java+code.`

To start up the Weka GUI, type

```
java -jar weka.jar
```

on the command prompt or the unix shell. You need to be in the directory where weka.jar resides or include weka.jar in your classpath (You can find weka.jar in the installation directory). We will make use of the Explorer interface. Thus, select Explorer from the list of applications.

Note that depending on the size of the data you process, you might get an out of memory error. In this case, you need to increase the maximum heap size of the java virtual machine

`(e.g. "java -Xmx1024m -jar weka.jar").`

8.1.3 Loading a dataset

After starting the Explorer interface, you will see the preprocess tab. In this tab, you can load and save datasets, apply filters to them, and convert the file format. Let's use the iris dataset as our sample dataset. To load the iris dataset, select "Open file", and choose iris.arff in the data directory. Arff is the default file type used by WEKA to store datasets. A dataset is a collection of instances. Each Instance consists of a number of attributes, any of which can be nominal (= one of a predefined list of values), numeric (= a real or integer number) or a string (= an arbitrary long list of characters, enclosed in "double quotes"). When you load the iris dataset, you can see basic statistics about the dataset. The dataset has 150 instances and 5 attributes (sepallength, sepalwidth, petallength, petalwidth, and class). When you choose an attribute, you will see detailed statistics about it (e.g. minimum and maximum value of the atribute) and the distribution of its values to existing classes.

8.1.4 10-fold crossvalidation

Click on the Classify tab and choose a classifier from the list. Many machine learning algorithms are available. Some of them are:

`NaiveBayes (bayes-->NaiveBayes), SVM (functions-->SMO), and Decision Trees (trees-->J48).`

Make sure that Cross-validation is selected and 10 Folds are required. Then click on Start. After cross-validation finishes, you will see the output of the classifier summarized via various performance metrics (e.g. precision, recall, f-measure for each class value, correctly classified instances (accuracy), confusion matrix). If you want to see also the individual predictions of the classifier, click on "More options", check "Output predictions" box and start cross-validation again. The instances in the prediction list appear in the same order as they appear in the data file.

8.1.5 File Format

Arff is the default file type used by WEKA to store datasets. See

`http://weka.wikispaces.com/ARFF+%28stable+version%29`

for a description of the arff format. Many sample arff files can be found in the data directory under the weka installation directory.

WEKA can also read csv files, which are easier to create. You might want to start with csv files in your first project. A csv file consists of a header (a comma seperated list of attribute names) followed by individual instances (each instance is a comma seperated list of attribute values). Make sure that you define the class attribute as the last attribute. Below is a part of a csv file storing the iris dataset.

```
sepallength,sepalwidth,petallength,petalwidth,class
5.1,3.5,1.4,0.2,Iris-setosa
```

```
4.9,3,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
7.7,3,6.1,2.3,Iris-virginica
6.3,3.4,5.6,2.4,Iris-virginica
6.4,3.1,5.5,1.8,Iris-virginica
5.5,2.4,3.8,1.1,Iris-versicolor
5.5,2.4,3.7,1,Iris-versicolor
5.8,2.7,3.9,1.2,Iris-versicolor
```

8.2 NLTK NLP Tookkit

Here's some example code and explanation for tokenizing / stemming / POS-tagging using NLTK.

NLTK supports robust tokenizing, stemming, and part-of-speech tagging functionality. Below are code snippets illustrating very simple ways of performing each of these tasks.

```
To split up a string into tokens, you can use the

   nltk.word_tokenize(‘‘some string’’) method:

>>> import nltk
>>> someString = ‘‘Some strings of text.’’
>>> tokens = nltk.word_tokenize(someString)
>>> tokens
[’Some’, ’strings’, ’of’, ’text’, ’.’]

To stem a list of tokens, you will need to create a PorterStemmer
object and then call its stem(‘‘token’’) method:

>>> stemmer = nltk.stem.porter.PorterStemmer()
>>> stems = []
>>> for token in tokens:
...     stems.append(stemmer.stem(token))
...
>>> stems
[’Some’, ’string’, ’of’, ’text’, ’.’]

To perform part-of-speech tagging, you can use the nltk.pos_tag([list
  of tokens]) method:

>>> tokens_pos = nltk.pos_tag(tokens)
>>> tokens_pos
[(’Some’, ’DT’), (’strings’, ’NNS’), (’of’, ’IN’), (’text’, ’NN’),
```

```
>>> ('.', '.')]

This method replaces all tokens in the list with a tuple containing
the token and an abbreviation representing the token's part of
speech. The full list of part-of-speech appreviations is available at
http://www.computing.dcu.ie/~acahill/tagset.html
<http://www.computing.dcu.ie/%7Eacahill/tagset.html> .

Note: you may need to install the NumPY package (available at
http://www.nltk.org/download) in order for the above code snippets to
work correctly.

For those interested in learning more about NLTK, an entire book on this
package is available online through PittCat at
http://pittcatplus.pitt.edu/?itemid=|library/marc/voyager|6336387
```