HOMEWORK 1: Sentiment Analysis (Text Categorization)

Team:

Rishi Josan
SBU ID: 108996773

Abhinav Mishra
SBU ID: 109272245

1. Introduction:

The task of text categorization or classification based on the sentiment of the text is generally tougher than classification based on the topic of the text. A clear demonstration of this difference is given in the paper [1]. The authors here evaluated the three famous machine learning techniques to classify documents based on the sentiment and their evaluations showed that the performance of these three techniques decreases for sentiment based classification. So, as the homework task we examined three more machine learning techniques namely language model classifier, perceptron and support vector machines, to classify the documents based on the sentiment.

2. Dataset description

The evaluations were done for the movie review domain. The dataset was obtained from [2]. It consisted of 1000 positive movie reviews and 1000 negative movie reviews.

3. Evaluation

3.1 Experimental Setup

The dataset consisted of 2000 movie reviews with equal number of positive as well as negative reviews. For the evaluation for each of the three techniques we used 5 fold cross validation technique. Each of the fold consisted of 20% of the dataset with even distribution of positive and negative reviews. The training of the classifier was done on 4 folds while testing was done on remaining 1 fold of the dataset and this process was repeated five times while rotating the test dataset in each iteration.

The entire project has been coded in Python. We have used the **Python NLTK (Natural Language Toolkit)** extensively [3]. In most cases we have also included our own implementation of the NLTK routines used. These have been commented.

We have also used **SciKit's SVM** classifier which essentially provides wrappers in Python for **libSVM** for the third experiment [4].

3.2 Language Model Classifier

For the evaluations of the language model classifier we used the following feature vectors:

1. Unigram Frequency
2. Bigram Frequency

The high level summary is as follows:

1. NLTK **Plain Text Corpora** were created for test/train data.
2. There were separate corpora for Positive and Negative training sets
3. Frequency distributions with **Laplace smoothing** for both positive and negative training sets was calculated. This was done for both unigram and bigram data.
4. Using these frequency distributions **Perplexity** was calculated for unigram as well as bigram model for each document in the test set.
5. For Perplexity calculation, we used the logarithm of the probabilities to **prevent underflow**
6. The outcome for both the positive and negative test data were stored in an **excel file** for evaluation.
7. This process was **repeated for five times** with the test data being rotated in each iteration.

3.3 Perceptron Classifier

For the Perceptron classifier we used unigram frequency as the feature vector.

1. For the training data, feature vectors were generated for each file
2. The feature vector for each file were the unigram frequencies. These were stored in a Python Dictionary which is essentially a HashMap.
3. The weight vector too was a Python Dictionary. For each unique word as key, the values were initialized to zero
4. **Training**: A temporary variable held the dot product of the weight and feature vector for each training instance. The weight vector was increased by the feature vector if the Perceptron incorrectly predicted the document as negative and decreased it by the feature vector if it was incorrectly predicted as Positive.
5. We evaluated this approach for 5, 10, 20 and 40 iterations. We observed that the weight vector converged at 37 iterations.
6. After training, the classifier was used to predict the sentiment of the test data. For this, the feature vector for test data was generated which was then used along with the weight vector to predict the outcome.

3.4 Support Vector Machine (SVM)

1. For the feature vector we used:
    a. Count **of unique words**.
    b. Count of **1000 most frequent words** in the training set.
2. For training, we created a NumPy array of dimensions: **[NumSamples x NumFeatures].**
3. We also created another one dimensional NumPy array with the actual labels for the Samples. We labeled positive reviews as 1 and negative documents as 0.
4. This was passed to the libSVM library to fit the SVM.
5. For testing, we created a NumPy array of dimensions [**NumSamples x NumFeatures]** again.
6. We evaluated using both L1 and L2 regularization.

4. Results:

4.1 Accuracy

All figures represent accuracy, i.e. the proportion of test documents predicted correctly

LMClassifier	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Unigram	0.805	0.8175	0.8075	0.825	0.79	0.809
Bigram	0.7525	0.765	0.7675	0.81	0.785	0.776

Perceptron	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Accuracy	0.815	0.845	0.83	0.86	0.8475	0.8395

SVM	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
L2	0.805	0.8325	0.8275	0.8575	0.8475	0.834
L1	0.795	0.8025	0.7975	0.785	0.835	0.803

We also compare the accuracy for perceptron varying the iterations. The weight vector had converged at 37 iterations.

Iterations	Accuracy
5	0.78
10	0.7825
15	0.7525
20	0.8225
40	0.8395

4.2 Trends Observed

4.2.1 Performance Trends

We see that we get best results for the Perceptron. This is marginally higher than SVM's accuracy. Given the simplicity of Perceptron when compared to SVM, this result is somewhat surprising.

We notice that the accuracy for the Language model classifier reduces when we use Bigrams instead of unigrams. The reason being our train set not being large enough to incorporate a high proportion of bigrams observed during testing.

We also notice that the SVM with L2 regularization performs better than the one with L1 regularization

4.2.2 Feature Vectors

The performance of Language Model Classifier using unigrams and bigrams has been discussed above.

For SVM, we initially started with the relative frequencies of the words as the feature vector. We observed bad performance for this. We then shifted over of the count of instances of words for the feature vector which resulted in much better performance. We also experimented with using only the 1000 most frequent words in the training set as the feature vector. The performance using this approach was actually very comparable to the original strategy which used all ~19000 unique words. The computation too was much faster.

4.2.3. Pros and Cons of Approaches

We personally feel that the perceptron performs very well, is intuitive and easy to understand and is easy to code. No third party libraries are required and it can be tweaked as desired. Therefor it is the best approach.

The SVM libraries do give a lot of ability to tweak varied parameters, but doing this mostly had a negative effect on the performance

The language model classifier is a good beginning point. What is good is that even though it is the simplest model, it performs reasonably well.

4.2.4. Reviews that failed in all approaches

There were 150 files for which all approaches failed. Given the high accuracy for all approaches, this implies that there is close correlation between failures for all three techniques. A subset of the files is listed below

Positive:

cv000_29590.txt, cv001_18431.txt, cv022_12864.txt, cv024_6778.txt, cv050_11175.txt, cv057_7453.txt, cv079_11933.txt, cv082_11080.txt

Negative:

cv221_27081.txt, cv232_16768.txt, cv237_20635.txt, cv267_16618.txt, cv305_9937.txt, cv333_9443.txt, cv376_20883.txt

One striking observation in all these files is that, all of these had **detailed descriptions** or **summaries** of the movies. These reviews tend to mention **quite a few nouns**: Names of characters, Actors and other movies. We also notice that these reviewers **tend to compare** these movies with others and use **fewer adjectives**. This is most evident in cv000_29590.txt. The reviewers in these reviews tend to describe and compare actors and/or movies instead of actually talking about his/her opinion.

Another interesting aspect we notice is the “Thwarted Expectation”. For example in the file cv221_27081.txt, we see the following sentences:

“the premise had great promise .”

“the computer-generated world within carl's mind could have been a bizarre , surreal universe governed by insanity and symbolism rather than logic .”

The author goes on to describe events and concludes with:

"the psychological importance of the horse and its fate is left to the viewer to ponder . "

"another element that should have been developed is the effect on catharine of merging with the mind of a psychopath ."

Clearly, we see the reviewer's thwarted expectation.

4.2.2. Reviews that passed in all approaches

Some examples are:

Positive:

cv003_11664.txt, cv008_29435.txt, cv010_29198.txt, cv011_12166.txt, cv014_13924.txt. cv015_29439.txt, cv016_4659.txt

Negative:

cv002_17424.txt, cv008_29326.txt, cv011_13044.txt, cv018_21672.txt, cv034_29446.txt, cv040_8829.txt. cv041_22364.txt, cv044_18429.txt, cv048_18380.txt

We see clear trends for a lot of adjectives being used. The authors in these movies express their opinion. For example in cv002_17424.txt the reviewer mentions:

*"the film is **way too juvenile** for the older mindset . "*

*"information on the characters is **literally spoon-fed** to the audience (would it be that hard to show us instead of telling us ?) , dialogue is **poorly** written , and the plot is **extremely predictable** ."*

Most of the review consists of what the reviewer actually thought about the movie, instead of comparing it to something else

5. Conclusion

As the result of this extensive evaluation of the three machine learning techniques we have observed that perceptron, even though simpler in approach, performs marginally better than the SVM as well as the language model classifier. We also found some cases where all the three techniques failed to make a correct classification and the analysis of such cases highlighted the fact that often the classifiers get confused by the way the document has been written.

6. References

[1] *Thumbs up? Sentiment Classification using Machine Learning Techniques* by Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, EMNLP, 2002

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[3] http://www.nltk.org/

[4] http://scikit-learn.org/stable/index.html