Sentiment Analysis

Technical Report · January 2015

CITATIONS
0

READS
366

1 author:

Isuru Suranga Wijesinghe
University of Leeds
12 PUBLICATIONS **15** CITATIONS

SEE PROFILE

Sentiment Analysis on Movie Reviews

UNIVERSITY OF MORATUWA

Faculty of Engineering

Module CS4622: Machine Learning

Department of Computer Science and Engineering

Introduction and Background

Sentiment analysis refers to the use of natural language processing, text analysis and computational linguistics to extract and identify subjective information in source materials. It aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude can be,

- His or her judgment or evaluation.
- Affective state (that is to say, the emotional state of the author when writing).
- The intended emotional communication (that is to say, the emotional effect the author wishes to have on the reader).

In last decade there is a rise of social media such as blogs and social networks, which has fueled the interest in sentiment analysis. Online opinion has turned into a kind of virtual currency with the proliferation of reviews, ratings, recommendations and other forms of online expression, for businesses that are looking to market their products, identify new opportunities and manage their reputations. In order to automate the process of filtering out the noise, understanding the conversations, identifying the relevant content and following appropriate actions, many are now looking to the field of sentiment analysis. The problem of most sentiment analysis algorithms is that they use simple terms to express sentiment about a product or service. However, cultural factors, sentence negation, sarcasm, terseness, language ambiguity and differing contexts make it extremely difficult to turn a string of written text into a simple pro or con sentiment.

A fundamental task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect. It focuses on whether the expressed opinion in a document, a sentence or a feature/aspect is positive, negative, or neutral. Sometimes it goes beyond polarity and looks at emotional states such as "angry," "sad," and "happy."

Another task in sentiment analysis is subjectivity/objectivity identification where it focuses on classifying a given text (usually a sentence) into one of the two classes (objective or subjective). As the subjectivity of words and phrases may depend on their context and an objective document may contain subjective sentences (a news article quoting people's opinions), this problem can sometimes be more difficult than polarity classification.

Existing approaches to sentiment analysis can be grouped into four main categories. They are keyword spotting, lexical affinity, statistical methods, and concept-level techniques. Keyword spotting classifies text by affect categories based on the presence of unambiguous affect words such as happy, sad, afraid, and bored. Lexical affinity improves keyword based approach by considering not only obvious affect words. It also assigns arbitrary words a probable “affinity” to particular emotions. Statistical methods influence on elements from machine learning such as latent semantic analysis, support vector machines, bag of words and Semantic Orientation. Unlike above mentioned purely syntactical techniques, concept-level approaches leverage on elements from knowledge representation such as ontologies and semantic networks so that they also able to detect semantics that are expressed in a subtle manner i.e. through the analysis of concepts that do not explicitly convey relevant information, but which are implicitly linked to other concepts that do so.
University of Stanford has proposed a novel approach of sentiment analysis. Most of the conventional sentiment prediction systems work just by looking at words in isolation, giving positive points for positive words and negative points for negative words and then summing up these points. In that approach, the order of words is ignored and important information is lost. In contrast, new deep learning model of this approach actually builds up a representation of

whole sentences based on the sentence structure. It computes the sentiment by considering how words compose the meaning of longer phrases. By using that kind of approach the model is not as easily fooled as previous models.

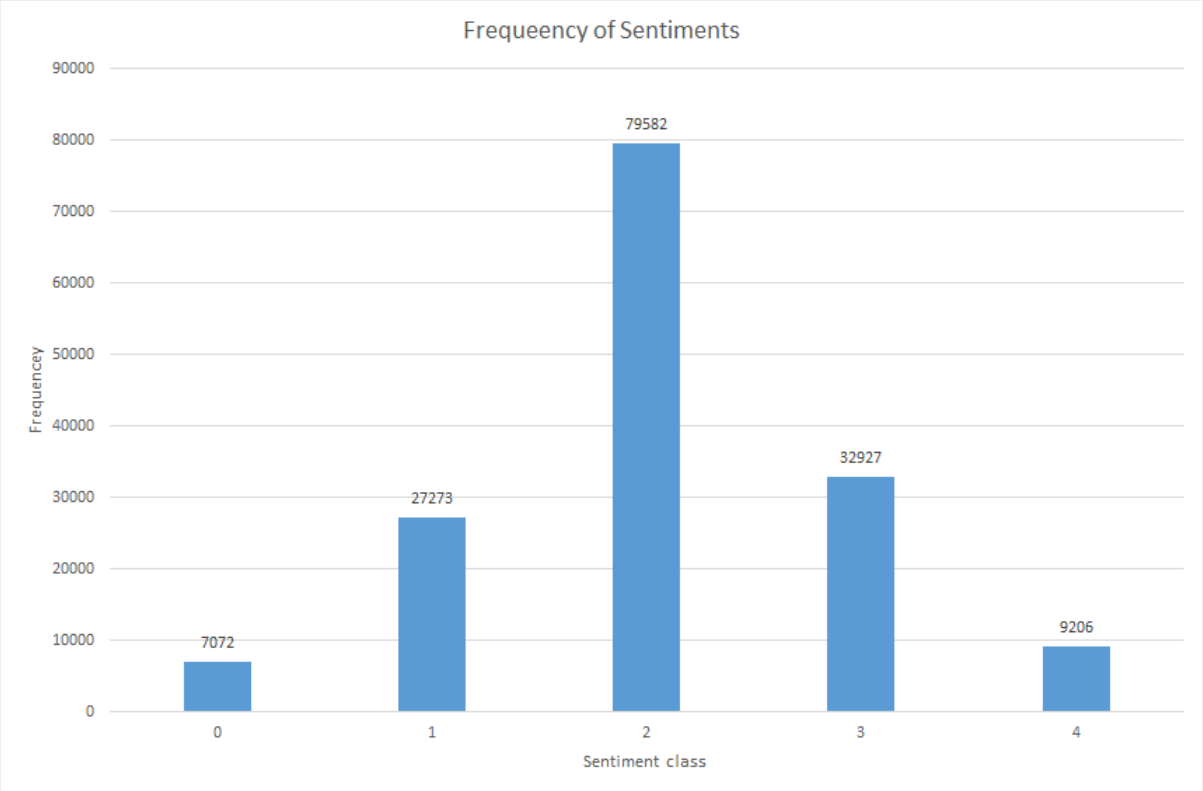
Data Preprocessing

The dataset provided in this competition is comprised of tab-separated files with phrases from the Rotten Tomatoes dataset. The dataset has been divided to training and test set for the purpose of benchmarking, but the sentences have been shuffled from their original order. Each Sentence in the dataset has been parsed into many phrases by the Stanford parser. Each phrase has a phrase Id. Each sentence has a sentence Id. Phrases that are repeated (such as short/common words) are only included once in the data.

The train.tsv contains the phrases and their associated sentiment labels. The test.tsv contains just phrases. Sentiment label to each phrase in test file should be assigned.
The sentiment labels used in the data set are,

- 0 - negative
- 1 - somewhat negative
- 2 - neutral
- 3 - somewhat positive
- 4 - positive

Training data set consists of 156059 phrases and their sentiment values are distributed as follows,



From graph it is clear that majority of the phrases has sentiment value of 2 which corresponds to neutral sentiment. Majority of the machine learning algorithms expect input data to be

vectors with numerical features. Therefore initially it is required to convert each phrase into feature vector. In this project tf-idf transformation is used in order to develop feature vectors for each phrase. Tf means term-frequency while tf-idf means term-frequency into inverse document-frequency. This is a common term weighting scheme in information retrieval, which has also found good use in document classification.
The main purpose of using tf-idf instead of using raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.
Scikit python library provides several transformers for tf-idf transformation such as count vectorizer, tfidf transformer and tfidf vectorizer. There are two approaches for transforming text data to tf-idf data.

1. First use count vectorizer to build feature vectors with frequency values. Then convert them to tf-idf values using tfidf transformer.
2. Tfidf vectorizer combines the functionality of both count vectorizer and tfidf transformer. It directly converts text data to feature vectors with tf-idf values.

All tokens are converted to lowercase to avoid duplicates when transferring text data to tf-idf format. In machine learning and data mining usually stop words (words that do not have an impact on decision making) are removed when processing text data to reduce the complexity of data. In this project, stop word removal has not performed as each phrase consists only few words and some phrases only consist of stop words.

Comparison of Techniques

Several multi class classification algorithms have been evaluated with given training data to find out the best algorithm for the task.
When evaluating the accuracy of algorithms, same training set cannot be used as model may over fit to training data but it cannot predict anything useful for unseen data. To avoid this problem, it is common practice to divide given training data to train set and test set. There are various approaches to divide given training data set to train set and test set. Initially hold out approach was used where 60% of original training data set is used for training and remaining amount is used for testing. But there is still a risk of overfitting on the test set because the parameters can be pinched until the algorithm performs optimally. This way, knowledge about the test set may leak into the model and evaluation metrics no longer report on generalization performance. Another part of the dataset can be held out as validation set to solve that issue. So the work flow for evaluating is training proceeds on the training set, evaluation is done on the validation set, and when the experiment seems to be successful, final evaluation can be done on the test set.
However, by partitioning the available data into three separate sets (training, evaluation and validation) we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets.
To solve these issues, cross validation is used. In k-fold CV, the training set is split into k smaller sets. Following procedure is followed for each of the k folds,

1. A model is trained using k-1 of the folds as training data. The resulting model is validated on the remaining part of the data.
2. The performance measure reported by k-fold cross-validation is the average of the values computed in each fold.

This approach can be computationally expensive compare to hold out method, but this does not waste too much data which is a major advantage in problem where the number of samples is very small.
Given below are the description and results of each evaluated technique.

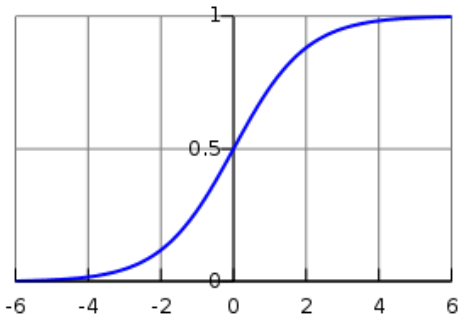
1. SVM
   Support vector machines are set of supervised learning methods used for classification, regression and outlier detection. It tries to find a hyperplane that can effectively divide the given training data into two parts. The major advantage of support vector machines is effectiveness in high dimensional spaces. Also it uses a subset of training points in the decision function called support vectors, so it is also memory efficient.
   The one drawback in SVM is when training data is highly unbalanced, resulting model tends to perform well on majority data but perform bad on minority data. In scikit library, different types of kernels such as linear, rbf and polynomial are provided. It implements the multi class classification using one against one approach. Performance have been evaluated using holdout approach (60% -training, 40%- testing) and 10-fold cross validation approaches and categorization accuracy values as follows,
   - Hold out method - 0.60
   - 10-fold cross validation – 0.612

2. SGD with SVM
   To improve the performance of SVM, Stochastic Gradient Descent (SGD) technique is used. It is a simple yet very efficient approach to discriminative learning of linear classifiers such as Support Vector Machines and Logistic Regression under convex loss functions. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention recently in the context of large-scale learning. The advantages of Stochastic Gradient Descent are efficiency, ease of implementation and there are a lot of opportunities for code tuning.
   One drawback of Stochastic Gradient Descent include SGD is it is sensitive to feature scaling. Performance have been evaluated using holdout approach (60% -training, 40%-testing) and 10-fold cross validation approaches and categorization accuracy values as follows,
   - Hold out method - 0.615
   - 10-fold cross validation – 0.62

3. Logistic Regression

Despite its name, it is a linear model for classification rather than regression. It is also known in the literature as logit regression, log-linear classifier and maximum-entropy classification (MaxEnt). In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function. A logistic function or logistic curve is a common S shape (sigmoid curve), with equation:

$$f(x) = \frac{L}{1 + \mathrm{e}^{-k(x-x_0)}}$$



Performance have been evaluated using holdout approach (60% -training, 40%- testing) and 10-fold cross validation approaches and categorization accuracy values as follows,

- Hold out method - 0.625
- 10-fold cross validation – 0.63

4. SGD with logistic regression

   Like in SVM, stochastic descent gradient approach has been used to improve the performance of logistic regression. Performance have been evaluated using holdout approach (60% -training, 40%- testing) and 10-fold cross validation approaches and categorization accuracy values as follows,

   - Hold out method - 0.63
   - 10-fold cross validation – 0.638

5. K Nearest Neighbors Classifier

   Neighbors-based classification is a type of instance-based learning or lazy learning as it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is performed from a simple majority vote of the nearest neighbors of test point. Scikit implements two different nearest neighbor classifiers, K Neighbors Classifier and Radius Neighbors Classifier. Among techniques, K Neighbors Classifier is the more commonly used technique. Here k denotes the number of neighbors considered for the decision. The basic nearest neighbor classification uses uniform weights where the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. But sometimes it is better to weight the neighbors such that nearer neighbors contribute more to the fit. In this project ten neighbors are considered with equal weights. Performance have been evaluated using holdout approach (60% -training, 40%- testing) and 10-fold cross validation approaches and categorization accuracy values as follows,

   - Hold out method - 0.625
   - 10-fold cross validation – 0.638

6. Random Forest

   A random forest is a Meta estimator that fits a number of decision tree classifiers on different sub samples of the dataset and use averaging method to improve the predictive accuracy and control over-fitting. Each tree in the ensemble is built from bootstrap sample from the training set. When splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases compare to the bias of a single non-random tree. It is balanced by decreasing variance through averaging hence yielding an overall better model. Performance have been evaluated using holdout approach (60% -training, 40%-

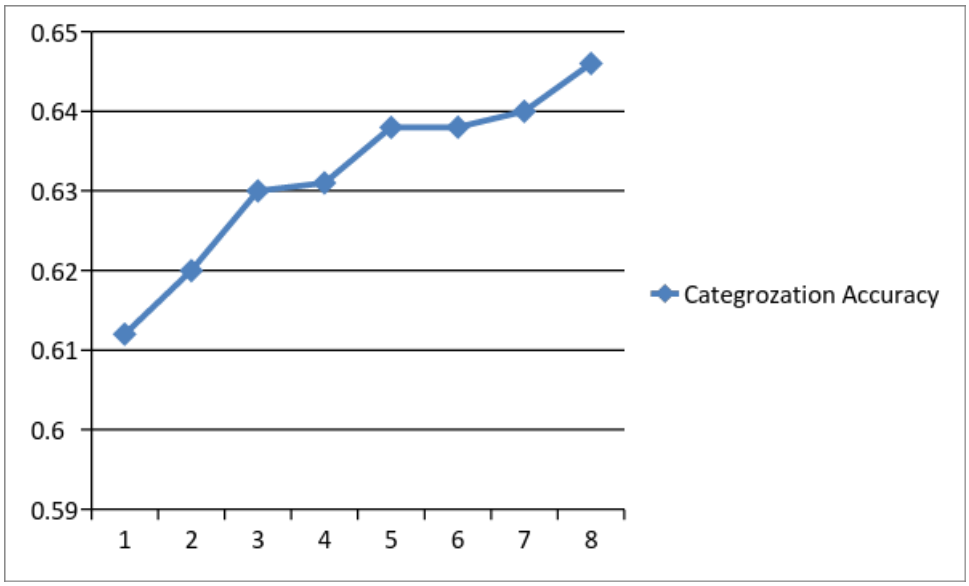
testing) and 10-fold cross validation approaches and categorization accuracy values as follows,

- Hold out method - 0.63
- 10-fold cross validation – 0.631

7. Naive Bayesian
   Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the naive assumption of independence between every pair of features. Performance have been evaluated using holdout approach (60% -training, 40%- testing) and 10-fold cross validation approaches and categorization accuracy values as follows,
   - Hold out method - 0.64
   - 10-fold cross validation – 0.646

8. Adaboost
   The core principle used in this algorithm is to fit a sequence of weak learners on repeatedly modified versions of the data. Then predictions from all of these weak learners are combined through a weighted majority vote to produce the final prediction. The data is modified at each boosting iteration by applying weights w_1, w_2,--, w_n to each of the training samples. Initially, those weights are all set to 1/N (N= number of tuples in data set), so that the first step simply trains a weak learner on the original data. In successive iterations, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. Weight modification happens in such a way that those training examples that were incorrectly predicted have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict have higher weights so that each subsequent weak learner is forced to concentrate on the examples that are missed by the previous ones in the sequence. Performance have been evaluated using holdout approach (60% -training, 40%- testing) and 10-fold cross validation approaches and categorization accuracy values as follows,
   - Hold out method - 0.638
   - 10-fold cross validation – 0.64

Summary of categorization accuracy with corresponding algorithm is given in following graph,



1. SVM
2. SGD-SVM
3. Logistic regression
4. Random forest
5. K nearest neighbour
6. SGD-logistic regression
7. Adaboost
8. Naive Bayes

Libraries Used

Even though above mentioned methods can be used to classify the entries into the 5 given classes after performing the sentiment analysis separately to train the classifiers and perform classifications, we used some higher level libraries which provides support for sentiment analysis tasks. We have followed 2 separate approaches. We have used Vowpal Wabbit library for python language and then we have used Stanford Core NLP library for java.

1. Vowpal Wabbit

Vowpal Wabbit is an open source fast out-of-core learning system library. It is specifically use for data mining and substantial as a well-organized scalable implementation of online machine learning and support for a number of machine learning reductions, importance weighting, and a selection of different loss functions and optimization algorithms. It also had descriptive documentation and tutorials to learn and to gain knowledge. There are certain features that are more powerful in Vowpal Wabbit tool such as input format, speed, scalability and feature pairing.

2. Stanford Core NLP

Stanford CoreNLP is a java library for natural language processing tasks which which can take raw text input and give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, and mark up the structure of sentences in terms of phrases and word dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, etc.
Stanford CoreNLP integrates many NLP tools, including the part-of-speech (POS) tagger, the named entity recognizer (NER), the parser, the coreference resolution system, the sentiment analysis, and the bootstrapped pattern learning tools. The basic distribution provides model files for the analysis of English, but the engine is compatible with models for other languages.

Use of ML Techniques

Implementation using Vowpal Wabbit

Tool and language Selection

Within this approach we used python for the implementation. We built Vowpal Wabbit on Windows machines with CygWin library. We used version 7.1 to do the project. We executed it using the command prompt.

Steps

Input format

The input format for Vowpal Wabbit is significantly more flexible than might be expected. Examples can have features consisting of free form text, which is interpreted in a bag-of-words way. There can even be multiple sets of free form text in different namespaces. Vowpal Wabbit has a very flexible and human readable input format. It can handle raw text as well. Therefore we do not need to vectorize it.
In this project it has multiple categorical class labels. Vowpal Wabbit assumes labels to be positive integers, beginning from one. It could be identified by five labels namely very negative, negative, neutral, positive, very positive as [1, 2, 3, 4, 5] respectively.

Data transformation and Feature Generation

The next phase of the project involves a process called Data transformation and Feature Generation. According to Vowpal Wabbit we couldn't feed tab separated files without any context. Here at the beginning what we had to do was to turn the row data sets into a Vowpal Wabbit-friendly format. In the data transformation step we decided the number of temporal features generated from the original feature set such words of a phrase and the length of the phrase.

Building training and testing set

To build the training and testing set ran a script in the command line to transform the .tsv data sets into .vw data sets. After that we created a training set and a test set in Vowpal Wabbit format.

Creating the model

Now we have our data sets in the correct format we can create our model with Vowpal Wabbit. Following figure will show the command we used to create the model.

```
vw rotten.train.vw -c -k --passes 300 --ngram 7 -b 24 --ect 5 -f
rotten.model.vw
```

Where:

- vw is the Vowpal Wabbit executable
- rotten.train.vw is the train set
- -c -k means to use a cache for multiple passes, and kill any existing cache
- --passes 300 means to make 300 passes over our data set
- --ngram 7 tells Vowpal Wabbit to create n-grams (7-grams in this case).
- -b 24 tells Vowpal Wabbit to use 24-bit hashes (18-bit hashes is default)
- -f rotten.model.vw means "save model as 'rotten.model.vw'".
- --ect (error correcting tournament [pdf]) in very simple terms tells Vowpal Wabbit that there are 5 possible labels and we want it to pick one.

Multiple passes over the data allows Vowpal Wabbit to better fit the model. N-grams increased performance because a phrase similar to “this movie was not good” would score positive sentiment for the token “good”. If 2-grams were used the model could detect negative sentiment in the token “not good”.

Vowpal Wabbit is extremely fast in part due to the hashing trick. With many features and a small-sized hash collision start occurring. These collisions may influence to the final results. Therefore we used multiple features sharing the same hash can have a PCA (Principle Component Analysis) for dimensionality reduction. PCA is a data reduction technique that can transform variables into orthogonal (uncorrelated) factors. Each factor is a weighted linear combination of the original feature set.

Making Predictions on the Test set

After building the model then give the test set as an input to the model and generated a predictions file. Then transform the predictions file to Kaggle Submission format.
To run Vowpal Wabbit in test mode and create predictions:

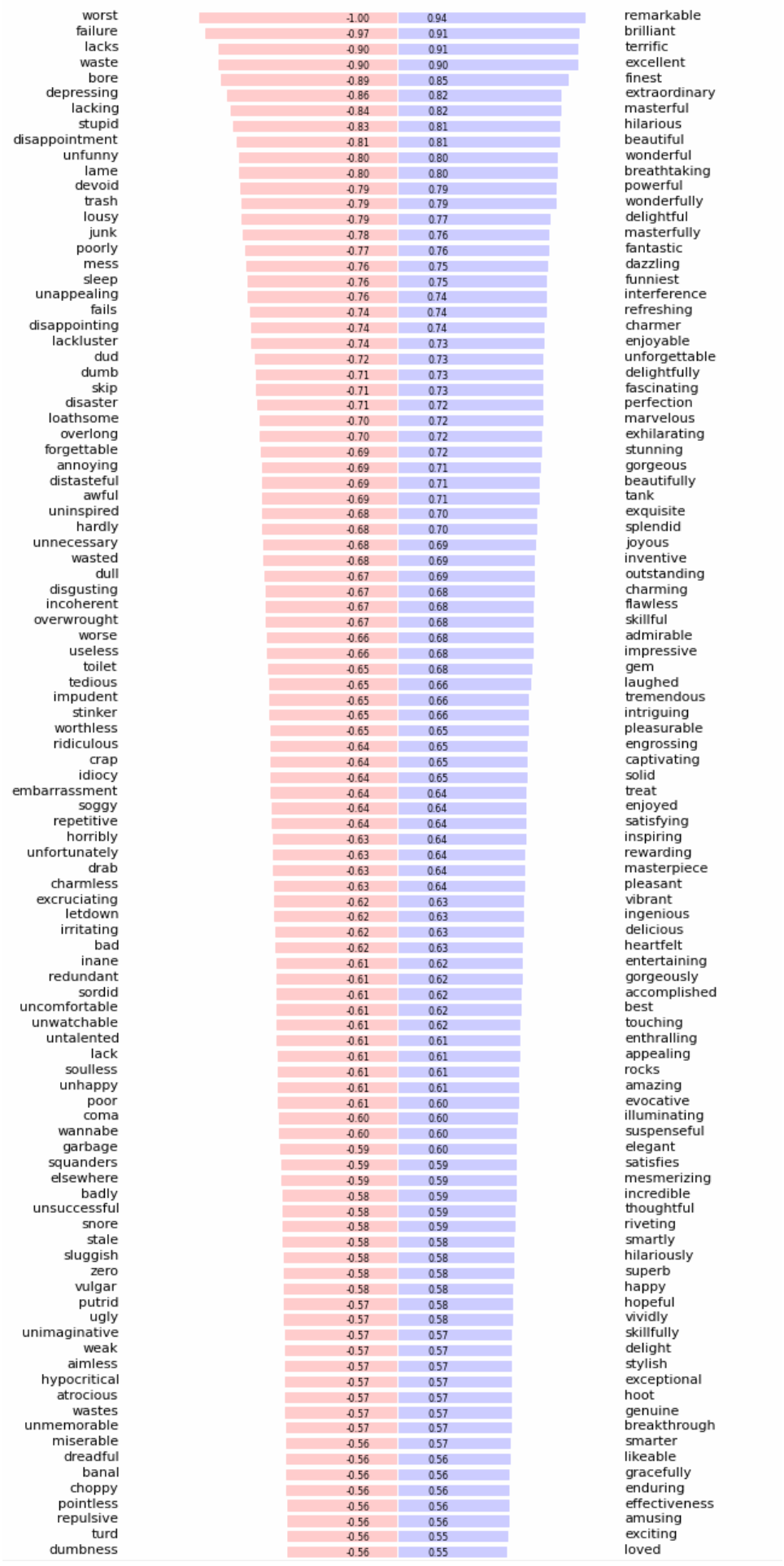
vw rotten.test.vw -t -i rotten.model.vw -p rotten.preds.txt

Where:

- vw is the Vowpal Wabbit executable
- rotten.test.vw the location to our test set
- -t tells to test only (no learning)
- -i rotten.model.vw says to use rotten.model.vw as the model
- -p rotten.preds.txt means “save predictions as ‘rotten.preds.txt’”

Visualization

Vowpal wabbit can give quick overview of feature relevance. It generates the output as follows.


worst -1.00 0.94 remarkable
failure -0.97 0.91 brilliant
lacks -0.90 0.91 terrific
waste -0.90 0.90 excellent
bore -0.89 0.85 finest
depressing -0.86 0.82 extraordinary
lacking -0.84 0.82 masterful
stupid -0.83 0.81 hilarious
disappointment -0.81 0.81 beautiful
unfunny -0.80 0.80 wonderful
lame -0.80 0.80 breathtaking
devoid -0.79 0.79 powerful
trash -0.79 0.79 wonderfully
lousy -0.79 0.77 delightful
junk -0.78 0.76 masterfully
poorly -0.77 0.76 fantastic
mess -0.76 0.75 dazzling
sleep -0.76 0.75 funniest
unappealing -0.76 0.74 interference
fails -0.74 0.74 refreshing
disappointing -0.74 0.74 charmer
lackluster -0.74 0.73 enjoyable
dud -0.72 0.73 unforgettable
dumb -0.71 0.73 delightfully
skip -0.71 0.73 fascinating
disaster -0.71 0.72 perfection
loathsome -0.70 0.72 marvelous
overlong -0.70 0.72 exhilarating
forgettable -0.69 0.72 stunning
annoying -0.69 0.71 gorgeous
distasteful -0.69 0.71 beautifully
awful -0.69 0.71 tank
uninspired -0.68 0.70 exquisite
hardly -0.68 0.70 splendid
unnecessary -0.68 0.69 joyous
wasted -0.68 0.69 inventive
dull -0.67 0.69 outstanding
disgusting -0.67 0.68 charming
incoherent -0.67 0.68 flawless
overwrought -0.67 0.68 skillful
worse -0.66 0.68 admirable
useless -0.66 0.68 impressive
toilet -0.65 0.68 gem
tedious -0.65 0.66 laughed
impudent -0.65 0.66 tremendous
stinker -0.65 0.66 intriguing
worthless -0.65 0.65 pleasurable
ridiculous -0.64 0.65 engrossing
crap -0.64 0.65 captivating
idiocy -0.64 0.65 solid
embarrassment -0.64 0.64 treat
soggy -0.64 0.64 enjoyed
repetitive -0.64 0.64 satisfying
horribly -0.63 0.64 inspiring
unfortunately -0.63 0.64 rewarding
drab -0.63 0.64 masterpiece
charmless -0.63 0.64 pleasant
excruciating -0.62 0.63 vibrant
letdown -0.62 0.63 ingenious
irritating -0.62 0.63 delicious
bad -0.62 0.63 heartfelt
inane -0.61 0.62 entertaining
redundant -0.61 0.62 gorgeously
sordid -0.61 0.62 accomplished
uncomfortable -0.61 0.62 best
unwatchable -0.61 0.62 touching
untalented -0.61 0.61 enthralling
lack -0.61 0.61 appealing
soulless -0.61 0.61 rocks
unhappy -0.61 0.61 amazing
poor -0.61 0.60 evocative
coma -0.60 0.60 illuminating
wannabe -0.60 0.60 suspenseful
garbage -0.59 0.60 elegant
squanders -0.59 0.59 satisfies
elsewhere -0.59 0.59 mesmerizing
badly -0.58 0.59 incredible
unsuccessful -0.58 0.59 thoughtful
snore -0.58 0.59 riveting
stale -0.58 0.58 smartly
sluggish -0.58 0.58 hilariously
zero -0.58 0.58 superb
vulgar -0.58 0.58 happy
putrid -0.57 0.58 hopeful
ugly -0.57 0.58 vividly
unimaginative -0.57 0.57 skillfully
weak -0.57 0.57 delight
aimless -0.57 0.57 stylish
hypocritical -0.57 0.57 exceptional
atrocious -0.57 0.57 hoot
wastes -0.57 0.57 genuine
unmemorable -0.57 0.57 breakthrough
miserable -0.56 0.57 smarter
dreadful -0.56 0.56 likeable
banal -0.56 0.56 gracefully
choppy -0.56 0.56 enduring
pointless -0.56 0.56 effectiveness
repulsive -0.56 0.56 amusing
turd -0.56 0.55 exciting
dumbness -0.56 0.55 loved

Implementation using Stanford Core NLP

Implementation in this approach was carried out using Java. As Stanford Core NLP library is also implemented in Java this choice is made mainly to make the implementation process much easier. After downloading the Stanford Core NLP library model training and data classification process was fairly straight forward.

Steps
Input format
In order to make the data input process easier and so that directly providing tab-separated files into a Stanford Classifier is possible we used a Column Data Classifier. Column Data Classifier is capable of doing context independent classification of a series of data items, where each data item is represented by a line of a file, as a list of String variables, in tab-separated columns. Then as by default the classifier does not know how to ignore the header record of the training data we had to manually remove the header record from the train.tsv file.
Then as the input data were required to be tokenized, the default tokenizer, which was provided with the Stanford Core NLP library, Stanford Tokenizer, was used to tokenize the input sentences. As a further preprocessing step on the input data lemmatization was carried out on the movie review data in order to convert all the words within those sentences into their base form using the Stanford Lemmatizer.

Creating the model
We built a linear classification model, in order to carry out the multiclass classification process, using Naive Bayes approach and the sentiment analysis support provided with Stanford Core NLP. Training of the model was carried out using the training set which was prepared as above. A properties file which was created in the following format was used to specify the required parameters to be used within the process of model creation.

```
trainFile = train.tsv
crossValidationFolds = 10
serializeTo = rottonmr-model.ser.gz
2.useClassFeature = true
2.splitWordsRegexp = \\s
2.useSplitWords = true
2.useSplitPrefixSuffixNGrams=true
2.maxNGramLeng=4
2.minNGramLeng=1
useNB = true
goldAnswerColumn = 3
```

Here,
trainFile - path of file to use as training data
crossValidationFolds - if positive, the training data is divided in to this many folds and cross-validation is done on the training data
serializeTo - path to serialize classifier to

Then, 2 specifies the second column (column numbering starts from 0) of the training data set, which is the sentence containing the movie review.
useClassFeature - include a feature for the class (as a class marginal)

splitWordsRegexp - if defined, this is used as a regular expression on which to split the whole string
useSplitWords - make features from the "words" that are returned by dividing the string on splitWordsRegexp or splitWordsTokenizerRegexp
useSplitPrefixSuffixNGrams - make features from prefixes and suffixes of each token, after splitting string with splitWordsRegexp
maxNGramLeng - n-grams above this size will not be used in the model
minNGramLeng - n-grams below this size will not be used in the model
useNB - use a Naive Bayes generative classifier (over all features) rather than a discriminative logistic regression classifier
goldAnswerColumn - the column which is treated as the class label column

Then this file was saved as a properties with the name, rottonmr.prop, and model building was initialized with the following command.
java -cp "*" edu.stanford.nlp.classify.ColumnDataClassifier -prop rottonmr.prop

Making Predictions on the Test set

After building the classification model it was saved into a model file, rottonmr-model.ser.gz, as specified within the properties file. Then this saved model was loaded into to a linear classifier which was contained within a Column Data Classifier and then the given test.tsv was classified using that classifier. Then from this classifications attributes required for the submission were extracted and save in the comma separated values format.

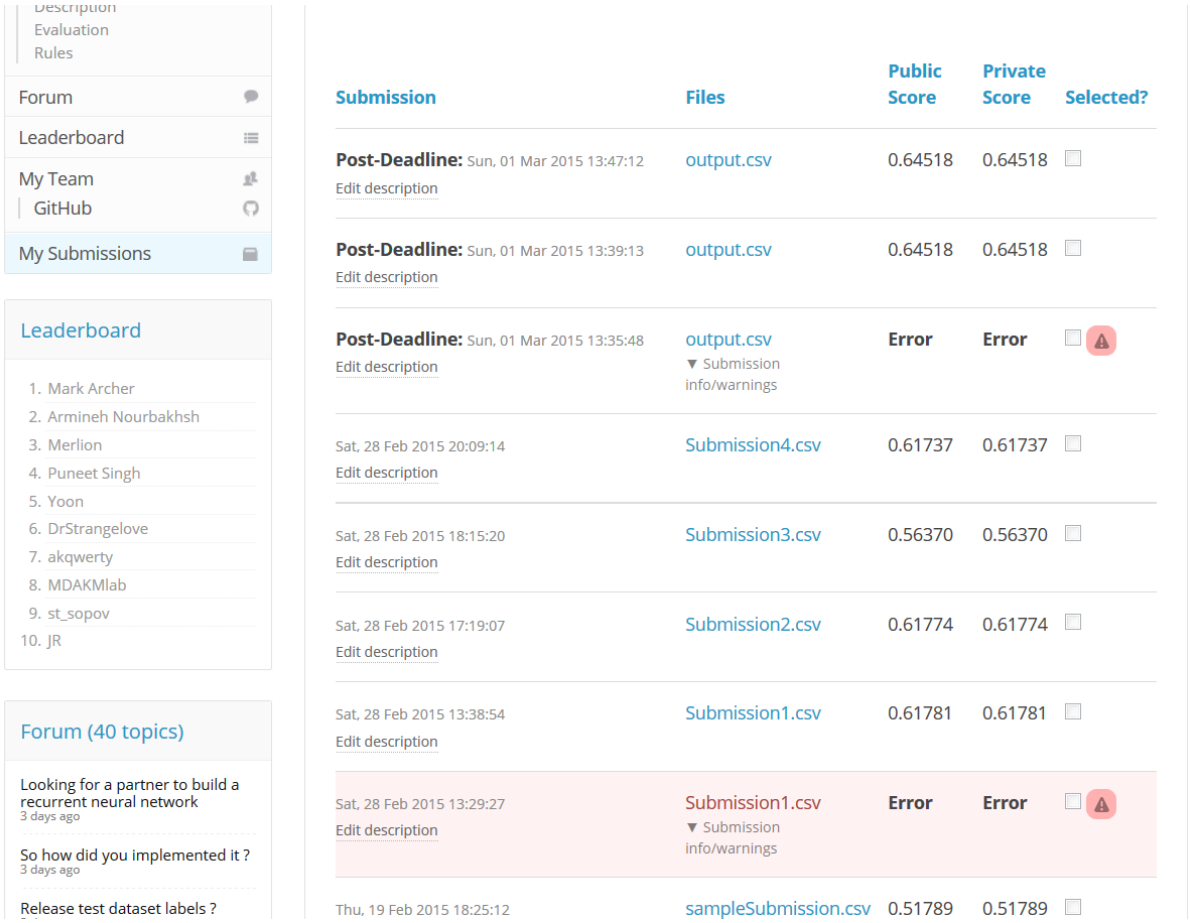
Submission and quality of results

Results of the cross-validation on the classification model build using the Stanford Core NLP based approach is given in the following figure.

```
Cls 1: TP=975 FN=1652 FP=1124 TN=11855; Acc 0.822 P 0.465 R 0.371 F1 0.413
Cls 2: TP=6488 FN=1500 FP=2945 TN=4673; Acc 0.715 P 0.688 R 0.812 F1 0.745
Cls 3: TP=1440 FN=1878 FP=1520 TN=10768; Acc 0.782 P 0.486 R 0.434 F1 0.459
Cls 4: TP=348 FN=635 FP=417 TN=14206; Acc 0.933 P 0.455 R 0.354 F1 0.398
Cls 0: TP=112 FN=578 FP=237 TN=14679; Acc 0.948 P 0.321 R 0.162 F1 0.216
Accuracy/micro-averaged F1: 0.59996
Macro-averaged F1: 0.44599
```

We have submitted results to the Kaggle competition of Sentiment Analysis on Movie Reviews. We submitted results obtained using both of our approaches. First we used Vowpal Wabbit and managed to improve the ROC score up to 0.61737 which gave us the rank of 333 at the end of the competition. However we were able to obtain a ROC score of 0.64518 about 14 hours after the completion of the competition which would have given us a rank of around 160 if we submitted it before competition ended.

Submission history:



Discussion and Conclusion

According to above results it has become clear that the model implemented with the support of the Stanford Core NLP has been much more promising than the Vowpal Wabbit based approach.

Basically with the Stanford Core NLP based method other than just carrying out normal classification process some important data preprocessing steps on the movie review data were carried out using the available tools within the Stanford Core NLP library, such as Stanford Tokenizer and Stanford Lemmatizer, in order to obtain further improvement of the performance. Moreover the Stanford Sentiment Analyzer module significantly supported with the model building process of this approach which considerably increased the accuracy of the model which was built.

Therefore with the discussion which has been carried out it has been clear that proper preprocessing of data based on natural language processing approaches as well as incorporating already existing models in the domain of sentiment analysis altogether with appropriate classification process can improve the performance of the model for multiclass classification of movie reviews.