Sentiment Analysis for Movie Reviews

CS 584 Spring16 Final Project

Purushottam Sharma Ashutosh Kumar Singh

Abstract

This paper summarizes the work performed for the final project for class CS584 on sentiment Analysis methods for movie reviews. It addresses the problem of deriving the correct opinion or sentiment of the reviewer from what has been written in the review. The discussion includes the comparison of two strategies to mine the sentiment from the text. First approach uses Weka ,a popular data mining tool. Seccond approach uses the SentiwordNet, a tool created at Princeton University. Weka learns a model for the task while Sentiwordnet works by assigning each word certain score. The evaluation metrics are determined and the projects concludes with comparison of the approaches.

1. Introduction

"Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials" [1]. Sentiment analysis is widely applied to reviews and social media for a variety of applications, ranging from marketing to customer service.

"Generally, sentiment analysis 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 may be his or her judgment or evaluation, affective state (the emotional state of the author when writing), or the intended emotional communication (the emotional effect the author wishes to have on the reader)"[1]. In the project, we use the movie review dataset by Pang and Lee at Cornell (http://www.cs.cornell.edu/people/pabo/movie-review-data). Within the dataset, are the 2000 processed down-cased review text files.

The two approaches that we use to perform sentiment analysis on movie reviews are:

1. Running classifiers on a previously labelled text, using Weka
2. Counting and/or weighting sentiment words that have been evaluated and tagged by experts, in a lexical resource, using SentiWordNet.

Finally, we compare the performance of the classifiers based on two approaches. Conclusion includes the comparison and the applicability of the two techniques. Even though the dataset used for this project is a movie review dataset, the findings from this project can be used in other situations, like Twitter comments, public sentiments during elections, product reviews, etc.

2. Related Work

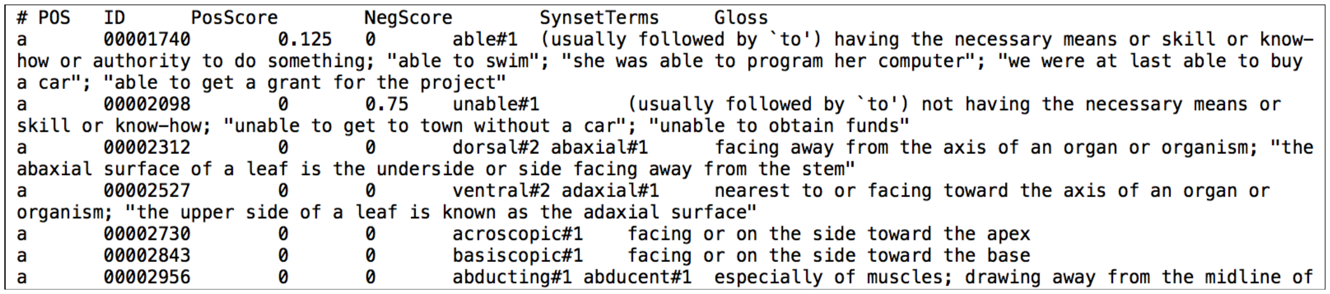
The paper published by Pang, Lee and Vaithyanathan (Pang, B., L. Lee, and S. Vaithyanathan. 2002, "Thumbs up ?: sentiment classification using machine learning techniques"), discussed the problem of classifying text based on the overall sentiment, classifying the reviews as positive or negative using three classifiers: Naive Bayes, maximum entropy classification, and support vector machines.

Another paper, "Sentiment Classification of Reviews Using SentiWordNet", by Bruno and Tierney applied SentiWordNet for the task of sentiment classification of film reviews. The approach involved counting positive and negative word scores to determine final sentiment of text. SentiWordNet was used for building the feature set and Classifiers were run on the data.

3. Solutions

3.1 Approach 1- Using Weka

We added on to the Pang, Lee approach by using the Filtered Chaining technique. With this technique the filters can be used progressively and output of previous filter can be used by the next. we chained the filters and classifiers to perform the experiment. We broke the text utterances into indexing terms and assigned them weights. Tokenizing the text into n-grams (e.g. "very good" vs. "good") can help improve the predictions of user sentiment. We used classifiers like Naïve Bayes, J48 and SVM on Weka for performing the classification task.



```
# POS   ID      PosScore        NegScore        SynsetTerms     Gloss
a       00001740        0.125   0       able#1  (usually followed by `to') having the necessary means or skill or know-
how or authority to do something; "able to swim"; "she was able to program her computer"; "we were at last able to buy
a car"; "able to get a grant for the project"
a       00002098        0       0.75    unable#1        (usually followed by `to') not having the necessary means or
skill or know-how; "unable to get to town without a car"; "unable to obtain funds"
a       00002312        0       0       dorsal#2 abaxial#1      facing away from the axis of an organ or organism; "the
abaxial surface of a leaf is the underside or side facing away from the stem"
a       00002527        0       0       ventral#2 adaxial#1     nearest to or facing toward the axis of an organ or
organism; "the upper side of a leaf is known as the adaxial surface"
a       00002730        0       0       acroscopic#1    facing or on the side toward the apex
a       00002843        0       0       basiscopic#1    facing or on the side toward the base
a       00002956        0       0       abducting#1 abducent#1  especially of muscles; drawing away from the midline of
```

Figure. 1: SentiWord's Lexical collection (SentiWordNet.txt)

3.2 Approach 2- Using SentiWordNet

As quoted from SentiWordNet's website, "SentiWordNet is a lexical resource for opinion mining. SentiWordNet assigns to each synset of WordNet three sentiment scores: positivity, negativity, objectivity" [1]. The SentiWordNet's website describes it as follows.This collection is currently available only for English language and covers the four most significant parts of speech (nouns, adjectives, adverbs and verbs). While the positivity and negativity scores are assigned by SentiWordNet in the resource file, the objectivity score can be calulated as:

$$\text{ObjScore} = 1 - (\text{PosScore} + \text{NegScore})$$

Thus there's a polarity score associated with each synset (or synonym set) which varies across different parts of speech. Each of the three scores ranges in the interval [0.0, 1.0], and their sum is 1.0 for each synset. The figure. 1 above shows how the SentiWord's Lexical collection looks like.The different Parts of Speech(POS) are represented as a(adjective), n(noun), r(adverb) and v(verb).

A naïve approach to sentiment analysis would be to use polarity scores of individual words(tokens) and sum them up. But in our presentation we showed how inaccurate that can be. The common words generally have a positive bias in them and how adjectives can be a better indicator of ones opinion. Also we presented the power of n-grams and part of speech. We developed an algorithm to harness the power of n-grams and off-the-shelf Part of Speech Tagger(Stanford Log-linear Part-Of-Speech Tagger) to improve on the naïve algorithm. We adjusted the polarity threshold for the final sentiment score to account for the positivity bias of the SentiWordNet and obtained better results. A code class('SentiWordNetDemoCode') by Petter Tornberg already present on the SentiWordNet's website was used in our algorithm. 'SentiWordNetDemoCode' considers all the occurrences of a token across various synsets in SentiWordNet and assigns a final polarity score to a token as a certain part of speech. Our algorithms uses this score to determine the final polarity score for the review. The language we used for coding is java.

There are two algorithms that we developed:

1. Tokenize text into n-grams and compute the sentiment score based on the token as:
   a. An adjective
   b. Average across different possible Parts of Speech(POS)
2. Tokenize and tag each word in a sentence with its POS using Stanford's POS Tagger to evaluate the score for exact POS

3.2.1 Algorithm 1- Tokenizing text into n-grams

In the naïve approach, we tokenize the text file into single words. if the word is an adjective as per SentiWordNet, then we retrieve the score for the word from the SentiWordNet dictionary and sum the score for all the adjectives and compute the final sentiment. If the sum is positive, the sentiment is positive, negative otherwise.

The rationale behind choosing adjective as the POS is that we believe that adjectives are a better indicator of a sentiment. For example, a word "entertaining" can appear as an adjective or as a verb in the text. As a verb it may not be very significant but as an adjective it can add to the positive sentiment of the text. In this approach however, we are not finding the exact POS from the context of the text. If the word can be an adjective in the SentiWordNet, we simply take its score.

Also we are not harnessing the power of the n-grams. The word "entertaining" is positive in itself but "very entertaining" can enhance the positive sentiment that that author wants to convey. Hence we developed two algorithms to consider n-grams and POS.

Algorithm:

1. Tokenize text into n-grams(uni, bi and tri)
2. Using the dictionary created through the SentiWordNetDemoCode class
3.
   a. Approach 1: if the given token is an adjective, fetch its polarity score
   b. Approach 2: Consider all possible POS for the token and take the average across all of them.

```
        fetch their polarity score

  4. Add obtained scores for all such tokens

  5. If    score<=threshold
               Sentiment is negative
     Else
               Sentiment is positive
```

Step 2.a is already discussed above. In step 2.b, to reduce the error in determining the exact POS for a token (for example: if we take "entertaining" as an adjective when it is actually used as a verb) we consider the average score for the token across all the possible POS.

After a few runs, it was found that while the positive classes were being detected with fair precision, but this was not the case with the negative classes. The false positives were almost equal to the true negatives. Hence we inferred that the SentiWordNet is positively biased. In order to account for this bias, we adjusted the threshold for the score of the review to a positive value we obtained from these runs. This adjustment helped in improving classification results.

3.2.2 Algorithm 2- Tokenizing and tagging each word in a sentence with its POS

In algorithm 1, we relied on the sentiment score of a token as an adjective or averaged its score across all possible POS. Thus, we tried to improve the algorithm to decipher the exact POS for a certain token. We used the "MaxentTagger" class in the POS tagger provided by The Stanford Natural Language Processing Group to tag the words in the text[5]. The algorithm has following steps:

```
  1. Feed text into the POS tagger (Stanford Log-Linear
     POS Tagger)

  2. Tokenize the tagged text and get the tag for each
     token

  3. Retrieve the score of the token as per the assigned
     POS tag from the dictionary created through the
     "SentiWordNetDemoCode"

  4. Get sum of all such scores for all tokens to determine
     the final score

  5. If    score<=threshold,
               Sentiment is negative
     Else
               Sentiment is positive
```

The POS tagger tags each word in the text with its derived POS. SentiWordNet includes four POS (adjective, noun, adverb and verb). One thing to notice is that the algorithm is not able to harness the power of n-grams as the tagger tags individual word rather than phrases and thus we could not precisely determine the POS of a group of words (like "very good"). Also, the threshold value has been adjusted through multiple runs. Proper tagging of words leads to results comparable to those in algorithm 1 with n-grams.

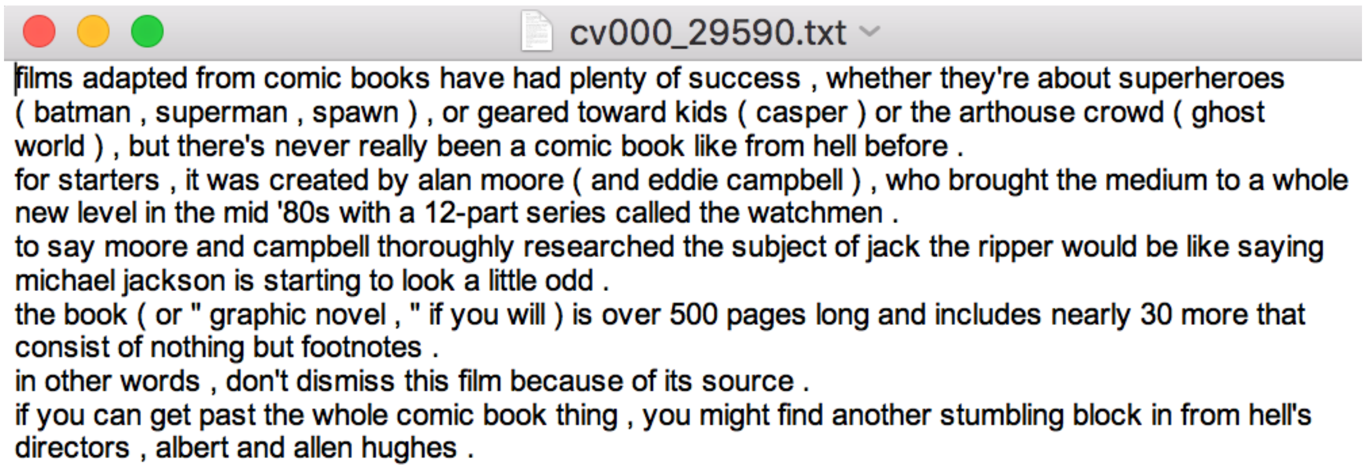
4. Experiments

4.1 Data

Movie Review Dataset Pang/Lee ACL 2004.

We used the movie review dataset labelled with respect to their overall sentiment polarity (positive or negative). As stated in the description of the dataset, it consists of unprocessed, unlabeled html files from the IMDb archive of the rec.arts.movies.reviews newsgroup, http://reviews.imdb.com/Reviews. The dataset is balanced so random chance would be 50%. The link to the datsset is given below:

http://www.cs.cornell.edu/people/pabo/moviereview-data/
Dataset Name: polarity dataset v2.0 (3.0Mb)
Total Instance: 1000 positive and 1000 negative processed reviews.



films adapted from comic books have had plenty of success , whether they're about superheroes (batman , superman , spawn) , or geared toward kids (casper) or the arthouse crowd (ghost world) , but there's never really been a comic book like from hell before .
for starters , it was created by alan moore (and eddie campbell) , who brought the medium to a whole new level in the mid '80s with a 12-part series called the watchmen .
to say moore and campbell thoroughly researched the subject of jack the ripper would be like saying michael jackson is starting to look a little odd .
the book (or " graphic novel , " if you will) is over 500 pages long and includes nearly 30 more that consist of nothing but footnotes .
in other words , don't dismiss this film because of its source .
if you can get past the whole comic book thing , you might find another stumbling block in from hell's directors , albert and allen hughes .

Figure. 2: a part of one of the 2000 review files

Introduced in Pang/Lee ACL 2004. Released June 2004. We have 2000 processed down-cased text files used in Pang/Lee ACL 2004. These data are kept in folders "pos" and "neg"; indicating the true classification (sentiment) of the function describing the measurement process and n_k is the measurement noise.

4.2 Experimental Setup

4.2.1 Approach 1 - Using Weka

For use with weka, the files in the dataset were formatted, converting them to .arff format. To convert the text files to .arff files, we used the weka.core.-converters.TextDirectoryLoader class, a new version of a previously existing TextDirectoryToArff.java class available in WEKA Documentation at wikispaces. The figure below shows the converted arff file of the entire dataset.

A cross-validation experiment was performed on the data. The number of folds for the cross-validation was set to be 5. we chained the Filters: StringToWordVector and Attribute-Selection, and the Classifiers :Naive Bayes, J48 and SVM. With chaining, the filters are executed progressively, and the output of the previous filter get fed to the next filter.

The step by step procedure of chaining Filters and Classifiers in Weka is as follows:

1. In Preprocess Tab of Weka, load the .arff file of the dataset created through .TextDirectoryLoader

2. Open Classify Tab and set the cross-validation fold value to 5. All the experiments were performed with the same =cross-validation value of 5

3. For Classifier, Select weka >classifiers >meta > FilteredClassifier

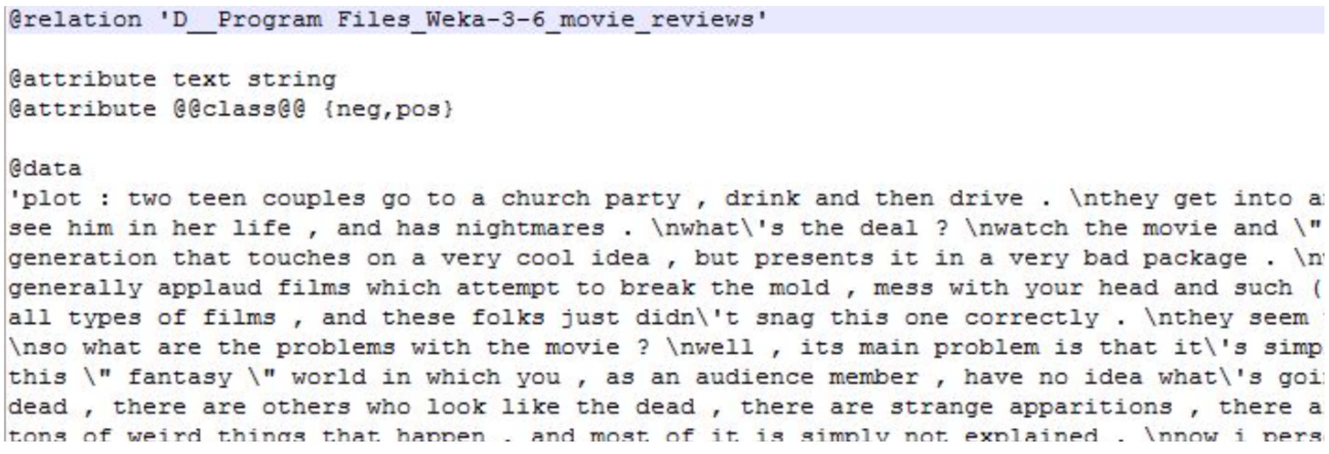
4. From the available Classifier, select Filtered-Classifier. Experiments are performed using NaiveBayes, J48 and SVM with their default configurations

5. From Filtered-Classifiers, select Multi Filter, we now chain the StringToWordVector (weka.filters. unsupervised.attribute.StringToWordVector) filter with Attribute Selection (weka.filters.supervised. attribute.Attribute Selection) filter. For StringTo Vector following parameters are of interest:
   tokenizer : NGram Tokenizer (weka.core.tokenizers.NGramTokenizer) with min and max between 1 to3, such that uni-grams, bi-grams, and trigrams can be set and delimiter is set to "\W"
   StopWord: it make the filter ignore the unimportant connecting words like a, the , are, etc

   The AttributeSelection has following parameters:
   evaluator: InfoGainAttributeEval
   search: Ranker (weka.attributeSelection.Ranker)

6. Start the classification task and the results will be shown.

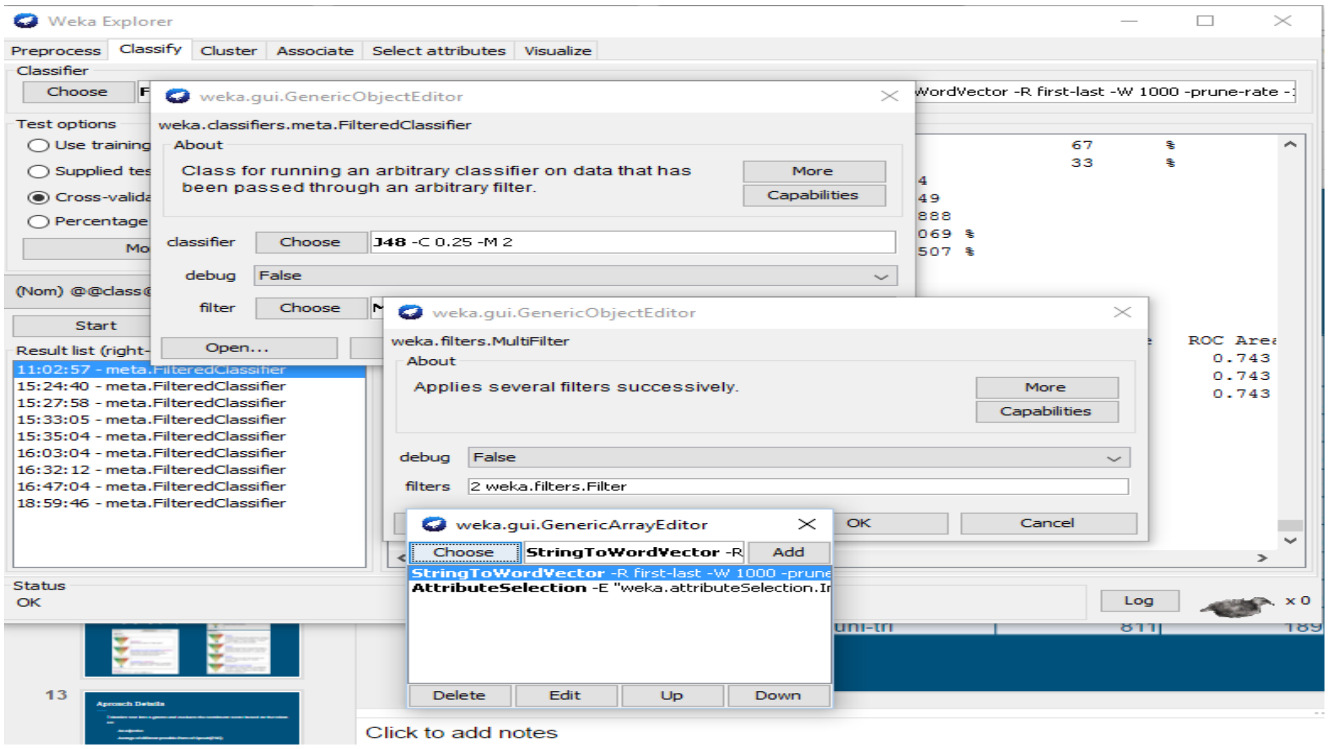
The steps are depicted in the figure. 4, on the next page of the Weka GUI


```
@relation 'D__Program Files_Weka-3-6_movie_reviews'

@attribute text string
@attribute @@class@@ {neg,pos}

@data
'plot : two teen couples go to a church party , drink and then drive . \nthey get into a
see him in her life , and has nightmares . \nwhat\'s the deal ? \nwatch the movie and \"
generation that touches on a very cool idea , but presents it in a very bad package . \n
generally applaud films which attempt to break the mold , mess with your head and such (
all types of films , and these folks just didn\'t snag this one correctly . \nthey seem
\nso what are the problems with the movie ? \nwell , its main problem is that it\'s simp
this \" fantasy \" world in which you , as an audience member , have no idea what\'s goi
dead , there are others who look like the dead , there are strange apparitions , there a
tons of weird things that happen , and most of it is simply not explained . \nnow i pers
```

Figure. 3: Review file converted to .arff file



WordVector -R first-last -W 1000 -prune-rate -

```
11:02:57 - meta.FilteredClassifier
15:24:40 - meta.FilteredClassifier
15:27:58 - meta.FilteredClassifier
15:33:05 - meta.FilteredClassifier
15:35:04 - meta.FilteredClassifier
16:03:04 - meta.FilteredClassifier
16:32:12 - meta.FilteredClassifier
16:47:04 - meta.FilteredClassifier
18:59:46 - meta.FilteredClassifier
```

```
67      %
33      %
4
49
888
069 %
507 %
ROC Area
0.743
0.743
0.743
```

weka.gui.GenericObjectEditor

weka.classifiers.meta.FilteredClassifier

About

Class for running an arbitrary classifier on data that has been passed through an arbitrary filter.

More | Capabilities

classifier: Choose | J48 -C 0.25 -M 2

debug: False

filter: Choose

Open...

weka.gui.GenericObjectEditor

weka.filters.MultiFilter

About

Applies several filters successively.

More | Capabilities

debug: False

filters: 2 weka.filters.Filter

OK | Cancel

weka.gui.GenericArrayEditor

Choose | StringToWordVector -R | Add

StringToWordVector -R first-last -W 1000 -prune

AttributeSelection -E "weka.attributeSelection.Ir

Delete | Edit | Up | Down

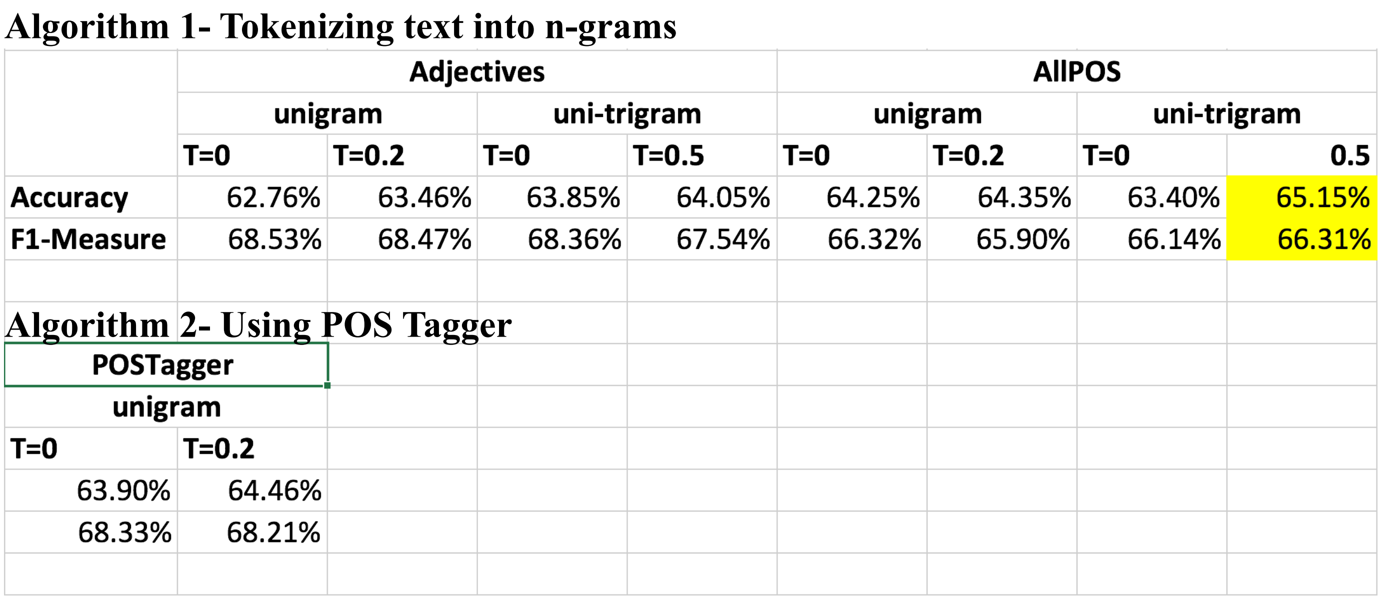
Figure. 4: Chaining Filters and Classifiers in Weka GUI

Experimental Results

Approach-1: Using Weka

	n-grams	TP	FP	FN	TN	Avg F-1 (%)	Accuracy(%)
NaïveBayes	uni	765	235	144	856	81	81.05
	tri	655	345	315	685	67	67
	uni-tri	826	174	222	778	80	80.2
J48	uni	706	294	300	700	70.3	70.3
	tri	664	336	419	581	62.2	62.25
	uni-tri	675	325	316	684	68	68
SVM	uni	806	194	195	805	80.5	80.5
	tri	721	279	300	700	71	71.05
	uni-tri	817	183	197	803	81	81

Figure. 5: Approach 1 Result



Approach-2: Using SentiNetWord

Algorithm 1- Tokenizing text into n-grams

	Adjectives				AllPOS			
	unigram		uni-trigram		unigram		uni-trigram	
	T=0	T=0.2	T=0	T=0.5	T=0	T=0.2	T=0	0.5
Accuracy	62.76%	63.46%	63.85%	64.05%	64.25%	64.35%	63.40%	65.15%
F1-Measure	68.53%	68.47%	68.36%	67.54%	66.32%	65.90%	66.14%	66.31%

Algorithm 2- Using POS Tagger

POSTagger	
unigram	
T=0	T=0.2
63.90%	64.46%
68.33%	68.21%

Figure. 6: Approach 2 Result

4.2.2 Approach 2 – Using SentiNetWord

For this approach the data from dataset does not need to be preprocessed. Both the 'pos' and 'neg' review folders are placed in the 'data' folder. We create a runnable jar SentimentClassifier.jar for our classifier and place it along with the SentiWordNet file and the 'data' folder.

The SentiWordNet.txt contains the SentiWord's Lexical collection of words, their POS and polarity scores. The taggers folder contains the taggers model for English. the tagger model in the taggers folder, the "English-Bidirectional-distsim.tagger" model is used for tagging the POS which claims to provide higher accuracy.[4]

At the end of the run, a file Output.txt is created in the same folder as the jar. Output.txt contains the experimental results for individual run of the experiment.

4.3 Evaluation Metrics

For the evaluation of approaches, we use the Accuracy and the F- measure as the metrics for the evaluation.

4.4 Experimental Results

4.4.1 Results: Approach 1 - Using Weka

The evaluation table for the algorithm have been shown on the next page.

The results show that:

1. The results show that the chaining filters and Classifiers gives good accuracy and F-1 measure.
2. Tokenizing data into uni-grams and uni-tri-grams improves on the accuracy ab F-1 measure
3. Using only tri-grams for tokenizing the data gives low accuracy and gives high false-positives (FPs)
4. Naïve Bayes and SVM both perform comparitively well compared to the J48.

4.4.2 Results: Approach 2 - Using SentiWordNet

For Approach-2, we had developed two algorithms. We discuss the results for both, one after the other.

For Algorithm 1, the algorithm gives best results when text is tokenized into uni-tri grams and average across all possible POS is taken. Although, the F1-score is similar to the one obtained with the tokens as unigrams and adjective as the only POS, there is 2% improvement in the accuracy. The overall result improves for a threshold value greater than 0, conforming with the notion of positivity bias mentioned earlier. Thus the performance of the algorithm overall is similar for when using uni and uni-tri grams. But it gets better when the average across different possible POS for a token is taken.

Algorithm 2 does not consider the n-grams capability of the text words. It works only on the uni-grams and tags each word with its POS. So, there are not many parameters in the algorithm to play with and we just derive an optimum value for the threshold by observing repeated runs. With an optimum threshold value of 0.2. The results were:
ACCURACY: 64.46%
F1-MEASURE: 68.21%

These results are comparable with the best performance of the Algorithm 1. The results have been achieved without harnessing the power of n-grams, so there is scope for further improvement in the performance.

5. Conclusions

On the analysis of the results for the different methods , it can be concluded that combining results from the different approaches, we can conclude that sentiment analysis using a lexical resource like SentiWordNet can give results that are comparable to the performance of J48 Classifier.

The use of on-the-go approaches using SentiWordNet can be advantageous in the manner that there is no cost involved in learning a model for the input data, which can be a huge factor when faced with massive datasets. Instead it can give results right away. Since the approach is based on keywords and their polarity, the need for a training data is not there. However the accuracy of this approach, as of now, is not very high. For high accuracy, the model learning classifiers like SVM and naive bayes' are better options. But still this approach can be used for online classification. Hence, it can be said that there is tradeoff between the accuracy and the cost and time, when we consider the choice of the learning based approach (using Weka classifiers) and the keyword based approach (SentiWordNet).

6.Our Contributions

We Debated over the scope of sentiment analysis to be considered for our 'small project' in order to come up with some meaningful analysis. We kept each other well informed about the progress of the project and kept each other unto date about new findings. We constantly researched to discover new ways to improve the results and upgrade the project overall.

7.References

[1] Wikipedia, **"Sentiment Analysis"** , https://en.wikipedia.org/wiki/Sentiment_analysis

[2] **"Movie Review Data"**, http://www.cs.cornell.edu/people/pabo/movie-review-data/

[3] "**Stefano Scerra's blog**",

http://www.stefanoscerra.it/movie-reviews-classification-weka-data-mining/

[4] **"Opinion Mining Using SentiWordNet"** http://stp.lingfil.uu.se/~santinim/sais/Ass1_Essays/Neele_Julia_SentiWordNet_V01.pdf

[5] The Stanford Natural Language Processing Group http://nlp.stanford.edu/software/tagger.shtml