Sentiment Analysis of Rotten Tomatoes Reviews (Kaggle) for Analyzing Box-Office Revenues

Abstract

The purpose of this study was to determine the efficacy of various modified machine-learning algorithms in numerically classifying movie reviews on Rotten Tomatoes on a scale of 0 – 4, where 0 represents the most negative reviews and 4 represents the most positive. In addition, we looked for a correlation between the overall sentiment of a movie's reviews and its corresponding box-office revenue. Our team consisted of four members.

1. Motivation

Not only is our problem task an interesting case study in sentiment analysis and the efficacy of various machine-learning techniques in this domain, accurate sentiment analysis on movies has the potential for wealth of applications including but not limited to large-scale opinion mining operations and incorporation into automated summarization software.

Additionally, it is important for movie producers to understand the degree to which the reviews on Rotten Tomato, an immensely popular movie review site, can impact their box office revenues. The overall favorability of a movie's reviews can help movie producers recognize along with genres, actors, and other characteristics what tends to define what movies critics and consumers like as well as what features may not be so favorable.

2. Statement of Problem and Task

2.1 Problem

Currently, movie reviews are analyzed simply by looking at the individual words in the review and their associated connotations without context, e.g. if the word witty or funny appears in a review, then the review may generally be classified as positive since these words taken individually has a positive connotation. This method is clearly not accurate when the review is "The movie was neither funny, nor witty." Thus, one substantial challenge in our classification task is to determine a way to account for the meaning and positioning of other qualifying words within the review.

2.1.1 Key Questions

Is it possible for movies to be classified with a numerical value indicating its "freshness" as opposed to just binary classification of "rotten" or "fresh"?

What is the extent to which diction, syntax, and phrasal structure serve as predictors to general sentiment? If so, what ways can we exploit these predictors?

What is the degree of influence that favorable and unfavorable movie reviews have on box office performance?

2.2 Task

One of our ultimate goals is to submit our results to Kaggle's public competition on the sentiment analysis of Rotten Tomato reviews. Our goal to in turn determine the impact that Rotten Tomato reviews have on box office revenues is an extension beyond the Kaggle competition.

3. Raw Data Analysis

Before beginning our machine learning techniques, we first looked at the raw data provided by Kaggle to identify trends in the data to inform the direction which we should take. Kaggle provides 8544 classified reviews as well as the classification of many sub-phrases within them.

3.1 Histogram of Scores

We first determined the relative frequencies of each different sentiment score in the sentences and the sub-phrases provided in the training set. (Figure 1.)

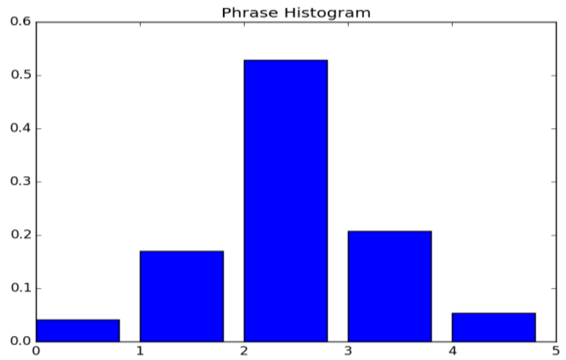

Figure 1. Histogram of sentiments of phrases to their frequencies.

The sentiment score of 2, or the neutral score, is extremely frequent in the given training phrases. The probable reason for this is that many sub-phrases of a sentence contain largely neutral words like “method”. In Figure 2, the frequencies of sentiments of the full sentences are shown in a histogram.

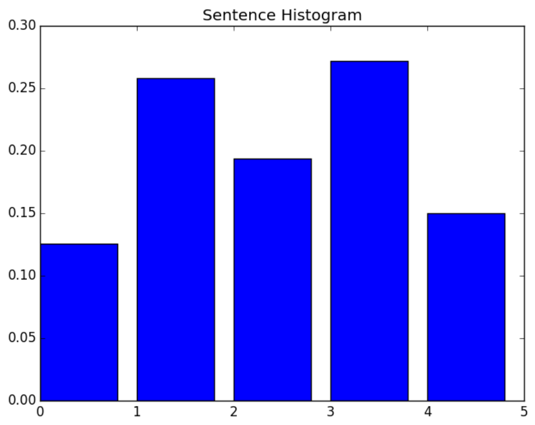

Figure 2. Histogram of sentiments of sentences to their frequencies.

In contrast to the sentiment frequencies of all phrases, the neutral sentiment score is less frequent. This leads to us believe that though a sentence can contain many sub-phrases of neutral sentiment, the review as a whole can demonstrate much greater variability.

3.2 Most Frequent Terms

We also determined the most frequent terms that appear when a review is given a certain sentiment score. In this analysis, we ignored a few extremely common tokens such as “movie” and punctuation components.

Tabel 1. Most frequent terms for each category and its count.

0	1	2	3	4
N’T - 107	N’T - 252	N’T - 147	N’T - 146	ONE - 117
ONE - 75	LIKE - 176	ONE - 109	ONE - 125	BEST - 69
LIKE - 69	ONE - 127	LIKE - 90	STORY - 115	FUNNY - 60
BAD - 66	MUCH - 97	STORY - 62	LIKE - 109	COMEDY - 54
EVEN - 52	STORY - 84	MUCH - 55	GOOD - 89	STORY - 47
STORY - 39	EVEN - 72	GOOD - 50	FUNNY - 69	LIKE - 47
COMEDY - 36	CHARACTERS - 69	TIME - 43	WAY - 69	LOVE - 45
CHARACTERS - 34	TIME - 67	MAY - 43	EVEN - 69	PERFORMANCES - 44
TIME - 34	LITTLE - 67	WOULD - 41	LOVE - 64	WORK - 41
WOULD - 34	BAD - 65	NEVER - 40	LITTLE - 63	GOOD - 40

Some words that appeared were as expected, such as “bad” in 0 and “best” in 4. It is surprising that “n’t” is the most frequent term for group 3 though when taken alone it has a negative connotation. It is also clear that most of the frequent words for a category are not unique. Because of this, we also looked at the most frequent terms for a category that are unique for that category.

Tabel 2. Most frequent terms for each category and its count, ignoring words in common.

0	1	2	3	4
DISGUSTING - 4	FRONTAL - 5	CYPHER - 4	FRANCE - 6	PACINO - 5
STINKER - 4	MANUFACTURED - 5	H.G - 4	J. - 6	SCOPE - 4
DOZE - 4	OBSURE - 4	WELLS - 4	DIVERSION - 6	KINNEAR - 4
LAUGHABLY - 4	BARNEY - 4	MARS - 4	EXPERIMENT - 5	FERRARA - 4
WHATSOEVER - 3	CHELSEA - 4	EXHAUSTING - 4	OPENNESS - 5	BREATHTAKINGLY - 4
INSULT - 3	DROWNED - 4	PARALLELS - 3	DESTINED - 4	IMMENSELY - 4
DEPRAVED - 3	DESIRED - 4	INNOCUOUS - 3	FESSENDEN - 4	SUPERBLY - 4
PILEUP - 3	STRETCH - 3	WORN - 3	CHORD - 4	FIRST-CLASS - 3
REJECTED - 3	HAWAIIAN - 3	DEPEND - 3	TOSCA - 4	INTELLIGENTLY - 3
SOURCES - 3	PAIRING - 3	SHOCKED - 3	BOURNE - 4	COMPELLINGLY - 3

Because we eliminated all the words that were in multiple categories, the actual word count dropped significantly. We also had terms specific to certain movies appear such as “bourne’, “h.g”, and “wells”. All the terms appeared in the category that we expected. This analysis helped to elucidate the distinction in the diction employed by reviews that carry varying sentiments.

4. General Approach

For a given review abstract, we perform a tokenization of the text using the natural language toolkit (NLTK) tokenizer. One way to model a review is to treat it simply as a bag of words – the only characteristics which we consider at are what words appear and how often which they appear. One significant drawback of this approach is that order of words is completely ignored.

One way to improve accuracy beyond a pure textual bag of words model is to include information about the syntax and relations among words. Models that take into account these more subtle features of a sentence may utilize all the various n-grams of a sentence or the parse tree of a review constructed from a probabilistic context-free grammar. As just one example, such a step makes it possible to only use the portion of a review containing adjectives and other significant carriers of sentiment or directly adjacent modifiers like negations such as "not good" as inputs into our algorithms.

To perform testing, we use 5-fold cross validation. Once we train a machine learning algorithm using a subset of the pre-sentiment classified dataset, we classify the remaining subset on a scale of 0 – 4. These results will yield insight into which learner has the best generalization error and consequently perform best on Kaggle's test data.

Finally, we will see how well our predicted review sentiments relate to the corresponding movie's actual box office performance. To do this, we will further apply machine learning methods such as regression to see if we can find a good correlation between the average review sentiment of a movie and its box office performance.

5. Experimental Evaluation of Sentiment Analysis

5.1 Histogram of Scores

Our first attempt at classification was a histogram of the sentiment scores. Training for this method is trivial, as we simply had to create a dictionary of phrases and their sentiment values. We have seven variations on this method. Our accuracies were determined using 5-fold cross validation. This uses all the sentences and phrases for training.

5.1.1 Counting scores of sub-phrases

For a new phrase, we looked up the sentiment values in the training set provided by Kaggle of all the possible sub-phrases. Once we found the sentiment value, we incremented the bucket associated with that score by 1. Lastly, we normalized the vector so its components sum to 1. The actual result we output is the index of the bucket with the largest value. Creating this feature vector is very fast as our data is stored in a dictionary. However, if words appear that are not in our training set, this method cannot assign a score. This has an accuracy of 52.3%.

5.1.2 Weighted count of scores of sub-phrases

This variation is very similar to 5.1.1, except that instead of adding 1 to the bucket, we increased the bucket associated with that score by the number of words in that sub-phrase. We used this weighting because longer sub-phrases are more likely to be closer to the overall sentiment score of the phrase. That way, phrases like "far from good" have more of a weight than "good". This has an accuracy of 57.7%. This was an improvement because the sentiment of longer phrases is closer to the overall sentiment of the sentence.

5.1.3 Ignoring bucket two

We followed the method of 5.1.2, except that we also ignored bucket 2 as it is a neutral sentiment and only assigned this value if the other buckets were empty. This has an accuracy of 55.8%. Since we had already weighted the sentiment in 5.1.2, removing bucket 2 lowered the accuracy, as some reviews are neutral although they contain a few phrases that are not.

5.1.4 Only counting adjectives

Instead of looking at all possible sub-phrases, we only looked at the adjectives. To do this, we used nltk's parts-of-speech tagger. Once we had the adjectives, we followed the method of 5.1.1. This method is slower than the previous variations as it takes time to do POS tagging. This has an accuracy of 52.7%. It brought a very small improvement in accuracy for the additional time it required.

5.1.5 Sub-prhases with adjectives

Instead of looking at all possible sub-phrases, we only looked at ones that contained an adjective. Once we had the adjectives, we got the sub-phrases that contained any of these adjectives and followed the method of 5.1.2. This has an accuracy of 58.2%. Adjectives tend to be terms with sentiment values closer to the overall sentiment of the sentence.

5.1.6 SVM with weighted count vector

Using the vector created in 5.1.2, we trained an SVM. This has an accuracy of 51.8%. It is interesting to note that a simple majority vote out performed the SVM here.

5.1.7 SVM with tagged sub-phrases vector

Using the vector created in 5.1.5, we trained an SVM. This has an accuracy of 58.5%. The improvement from using the SVM is basically negligible.

5.1.8 Discussion

As the sentiment score 2 is 52% of our training phrases, many of these methods just barely did better than assigning the score of 2 to everything, even when using machine learning methods such as SVMs. Thus, we looked for better methods.

5.2 Bag-of-Words

Our second attempt at classification was bag-of-words (BOW). We used a basic BOW model that had a feature vector where each component was associated with a pre-defined vocabulary token. Our vocabulary size in variations that did not perform stopping and stemming was 19,479, meaning our feature vector dimensionality was close to 20,000. Naturally, a review will only contain a fraction of the words present in the entire vocabulary. Thus, a sparse encoding was necessary in this scenario. In fact, it is almost impossible to train any classifier if a dense vector representation is used due to the sheer memory requirements. All the given sentences and phrases are used for training. Other possible feature vector encodings which we tried were binary weights (instead of the frequency of the word in each component, simply whether it occurs or not is listed) and feature vectors using tf*idf weights. However, these variations actually performed slightly worse so they were not used.

5.2.1 Stopping and Stemming

Using this vector, we trained an SVM with 4 variations: normal, stopping only, stemming only, and both.

Tabel 3. Accuracy using 5-fold cross validation with an SVM

Normal	Stopping Only	Stemming Only	Stopping & Stemming
64.2%	68.2%	64.6%	67.0%

Incorporating stopping increased our accuracy, as stop words did not add meaning to the sentence. Thus, removing them reduced noise. However, adding stemming could only marginally increase the accuracy, as in this domain, some meaning can be lost through stemming.

5.2.2 Other classifiers

Although the SVM performed well, we wanted to see how other classifiers did on the same data using the same bag of words model to find the best one. Using the same vector, we trained several other classifiers: ridge classifier, perceptron, multinomial Naïve Bayes, stochastic gradient descent, and nearest centroid. We also trained them using the variations of stopping and stemming. In Table 4, there are some of the results.

Tabel 4. Accuracy using 5-fold cross validation with other classifiers

Classifier	Accuracy
Ridge Classifier - Normal	61.5%
Ridge Classifier - Stopping	64.1%
Ridge Classifier - Stemming	61.4%
Perceptron - Normal	55.8%
Perceptron - Stopping	61.8%
Perceptron - Stemming	54.9%
Multinomial Naïve Bayes - Stopping	60.2%
Stochastic Gradient Descent - Stopping	56.7%
Nearest Centroid - Stopping	47.9%
Nearest Centroid - Stemming	38.7%

These classifiers all performed worse than the BOW soft-margin SVM with C = 1.0. As the SVM is an optimized version of a Perceptron these results were as expected. The interesting case was Nearest Centroid. It performed worse than simply assigning the sentiment score of 2 to every instance. One possible reason for these results could be because of the size of our vector. By Law of Large numbers, since our vector space was so large, the distance measure for Nearest Centroid was less effective.

5.2.1 Multi-level classifier

Using the same vector, we trained a multilevel classifier using SVMs. The first level was a coarse classifier that assigns the labels: “low”, “2”, “high”. The next level splits “low” into “0” and “1” and “high” into “3” and “4”. The idea behind this is that levels close to each other have small nuances that differentiate them. Thus, looking at just those levels will help identify those differences.

Unfortunately, because of the sparse encoding we were unable to establish a reliable testing method.

5.3 Bag-of-Words Variations

Branching from a pure BOW model, a variation would be not only including unigrams in the feature vector, but also at bigrams, and possibly even trigrams. Even with knowing our vocabulary, the problem with this approach is that the vector becomes unmanageably long and thus impossible to train on. A way to overcome this problem involves hashing. We hash the unigrams and bigrams of a phrase four times. Two of these are used to identify the buckets and the other two indicate the signs. Modifying two buckets mimics a Bloom Filter and adds variation. The purpose of using a hash for the sign is to increase variation and reduce bias.

When run with a one vs. one multiclass SVM, this method never finished training. We were unable to figure out why this was the case. We think our data may not have been linearly separable. Also, as we have 5 classes, each time we needed to train 10 SVMs. This also uses all the sentences and phrases for training. Using smaller sets of data, however, this variation does not perform better.

5.4 Training Sentences and Phrases Separately

Because the distribution of sentiments for sentences and phrases differ and the length of sentences is longer than the length of phrases, we decided to train two classifiers: one for sentences and one for phrases. However, this performed poorly. One possible reason for this could be because the training data few sentences as it was mainly the sub-phrases of those sentences.

6. Analysis of Box Office Revenues

6.1 Web Crawling

For analyzing the box office revenue, we needed to know what movies were associated with each movie review and its final box office revenue. Since the data provided on Kaggle did not contain this information, we found our own data. We created a web crawler to go through Rotten Tomatoes to collect the reviews written by experts and the box office revenue for a given movie. We decided to only look at the reviews by experts, as they are more likely to be grammatically correct and fit the format of our training data.

6.2 Discussion

We were able to format the reviews and test it using a bag-of-words SVM from 5.2.1. After creating our own sentiment, we took the average of the scores for each movie, plotted it against the log of the revenue, and drew a linear regression line. Although we were hoping to see some correlation, the results did not show this. Even after we added more data, we did not end up seeing any stronger of a correlation. A graph of our results is shown below.

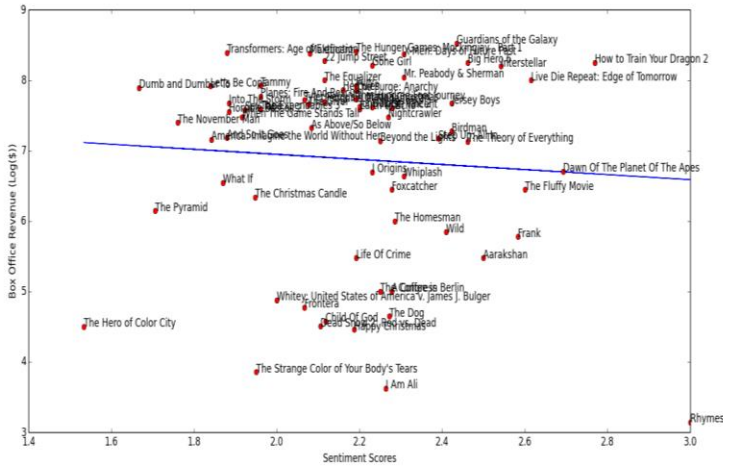

Figure 3. plot of average sentiment scores vs log(box office revenues).

The points are wide spread and do not fit any normal regression. This near horizontal regression line shows us that there is very little correlation to the movie reviews and the box office revenue. Although our classifier is not perfect, taking the average of the review scores for each movie reduces the noise caused by this imperfection, making it unlikely for the classifications to be significantly wrong and cause of this lack of correlation. We feel that the reviews itself may not be well correlated on how the movie performs in the box office. There are probably more important factors (like genre, actors present, how well advertised the movie is, etc.) that also affect revenue. Therefore, we conclude that sentiment analysis of movie reviews alone is not the right way to predict box office revenue.

7. Applications and Related Work

7.1 Opinion Mining

Film production companies can employ these automated sentiment analysis techniques to automatically mine large corpora of reviews in order to direct future marketing decisions. These sentiment analysis techniques can be used for opinion mining of various other types of articles as well, not just movie reviews. Related classification tasks such as relevancy testing can also stand to benefit from these techniques.

7.2 Text summarization

These techniques can also be used for text summarization. This requires advanced natural language processing to get the meaning out of the text. Instead of having someone manually read article after article, natural language processing can be used to summarize articles. Though classification may not always be the objective of summarizing, sentiment analysis can help accomplish this goal.

8. Future Work

8.1 Bag-Of-Words Augmented with Syntactical Information

Another variation on bag-of-words is to not only include the relevant bigrams and trigrams in the feature vector for a given review, but also numerical aspects of the parse tree for the review. Thus, we also encode in the feature vector attributes such as the frequency of each part of speech (“NN”, “JJ”, “ADV”, etc), the maximum depth of the parse tree, and the total number of subtrees. We do this because the very sentence structure of a phrase can yield insight into its sentiment.

8.2 Non-Vectorial Parse Tree

Along the same lines of a parse tree, another variation of classification involves feeding in the several forms of the parse trees of reviews into an SVM classifier as a piece of non-vectorial data. Of course, this necessitates we define a kernel function on trees that defines similarity. This function could be the number of common subtree structures between two reviews. We were unable to implement this algorithm due to the fact that our software, sci-kit SVM, does not natively support non-vectorial string feature vectors. The outline of the sub-structure comparing Kernel is written however – we may later incorporate this into liblinear.

8.3 Hyper-Parameter Optimization

Currently, the parameters of our best classifiers have yet to be tuned. Optimally, future work would involve performing a grid search on the classifier parameters, particularly the value of C and tol in the soft-margin SVM classifier.

9. Conclusion

Current movie analysis looks only at individual words to decide its sentiment. In this paper, we compared various machine learning algorithms that would improve the classification of movie reviews. We also looked at the correlation between the sentiment of reviews and the associated box office to see how much influence reviews have on moviegoers.

The algorithms we used are histograms of scores, bag-of-words, SVM and other classifiers including ridge, perceptron, multinomial Naïve Bayes, stochastic gradient descent, and nearest centroid. The variations of histograms of scores had accuracies a little above 50% on average, which is not very different from classifying all sentences as neutral. The BOW showed a greater improvement and the best result we got was the BOW SVM with stopping which had an accuracy of 68.2%.

To determine whether our SVM classifier with stopping did a good job, we did a significance test. We constructed a 99% confidence interval and found that the bounds of these were [54.1231%, 60.913%]. Since this classifier is at 68.1% accuracy, it is outside of the bounds and is actually statistically significant.

Our accuracy on Kaggle is 60.32% (As a benchmark the individual currently in 5th place has around 65% accuracy and the individual in 1st has around 75%). We are able to conclude that machine learning techniques such as SVMs can be used, to a fairly accurate degree, to perform sentiment analysis on movie reviews, although the scale of 0 – 4 may be too refined to achieve very high accuracies.

Unfortunately, we were unable to find any significant correlation between the sentiment of movie reviews from our classifier and the box office revenues. There are many other factors that influences how much people want to watch a certain movie like genre, actors and advertisements. We conclude that the sentiments alone of critique reviews do not work well to predict box office revenue.

References

Barthelemy, P. Thomas, Devin Guillory, and Chip Mandal. "Using Twitter Data to Predict Box Office Revenues." *Stanford* (2012): 1-3. 2012. Web. 18 Oct. 2014.

Socher, Richard, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank." *Stanford* (2013): 1-12. 2013. Web. 18 Oct. 2014.

Weinberger, Kilian, Anirban Dasgupta, Alex Smola, and Josh Attenberg. "Feature Hashing for Large Scale Multitask Learning." (n.d.): n. pag. Yahoo! Research. Web. 9 Nov. 2014.

Sean Massung, Chengxiang Zhai, and Julia Hockenmaier. “Structural parse tree features for text representation.” In ICSC, pages 9–16. Web. 11 Nov. 2014.