Sentiment Prediction on Movie Reviews

Patrick Lehane
Eric Wang

Introduction

Sentiment prediction is the process where analysis of words and phrases leads to a conclusion on whether a comment has a positive or negative bias, and is a growing area of research in machine learning. The purpose of this project was to predict whether a movie review had a positive or negative sentiment, given a set of training data to work with. Our method was using a logistic regression inside of a bag of features in order to predict the outcome of the review. The features could range from things like words to the punctuation that was used on the words or capitalization that appear in the reviews.

The problem is important for many reasons, but it is particularly useful to businesses and for social media analysis. Sentiment analysis is particularly important because it allows businesses to trawl social media and analyze how their products are faring. Social media, with its quick growth, has become an increasingly important part of everyday life. Twitter alone has an average of 500 million tweets per day, which is an amount that is unreasonable to analyze manually, so there is great demand for automated analysis of all of the tweets. This would lead to methods for companies to determine public sentiment as well as see the influences that social media has on its consumers. This means that sentiment analysis is a large part of data mining, since groups are trying to amass large amounts of data to see where larger-scale trends are going. Sentiment also impacts natural language processing, another field where machine learning is making growths, since the two are so closely related.

The method of solving the problem is also important. Since reviews only have one input parameter (words), you need a way to increase the number of features in order to make predictions on the topic. One way of doing this is to simply parse out all the words and feed them into a logistic regression, which would be the first way tried outside of simply using probability. Logistic regression is the base of most machine learning models, so any attempts using it naturally lead into more advanced techniques. The method to solve sentiment analysis could also revolve around many other parameters, like using the length of a post itself or various capitalization/emoticons that could affect the sentiment, all leading to models that could be built on many different features.

Background

Sentiment analysis also has many approaches that have been already tried. One method is using LDA to determine the value of words probabilistically, without needing to be assigned values (Maas, 142). In addition, they used it to assign vectors based on topics, rather than the words themselves. Following the creation of the vectors, the sentiment itself would be assigned through a supervised method in order to test it. This was done as models that included sentiment added in were much better at predicting the polarity of a review as compared to a model that had no sentiment input. In addition, this led to greater amounts of related words as well as increasing accuracy among those words.

Other methods of looking into sentiment prediction involved bigrams integrated with both the Naive Bayes method and Support Vector Machines. The inclusion of bigrams inside these models was shown to have improvements over many other models including many other state of the art models. Normally, “word bigram features are not that commonly used in text classification tasks”, because a “bag of words” was more than enough to determine it, but for sentiment, word pairs became much more indicative of actual meaning due to the capture of modified verbs and nouns (Wang, 93). It was also found that Naive Bayes worked better on shorter inputs while SVMs were more accurate when there was more input data.

Our Approach

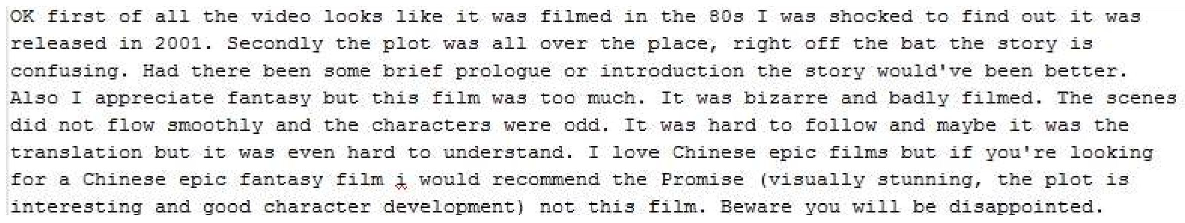
Our approach was based on the Naive Bayes method of vector creation. By taking incidences of words, the most polarizing words would be assigned either a positive or negative value and fed into the learning algorithm. Thus, we used a version of the maximum a postieri in order to create the equation.

$$\hat{y} = \underset{k \in \{1,\dots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i | C_k).$$

Here, C would have been the data set in general while X would be the input vector of words that would be used to assign value and given training data. Then, the idea is that after the vector of words was created, and run, we would take gradient descent steps in order to find a way to optimize the value of each feature inside of the bag of words. We used the sigmoid function in order to normalize all of the outputs, which was then translated into a predictor for the given input review.

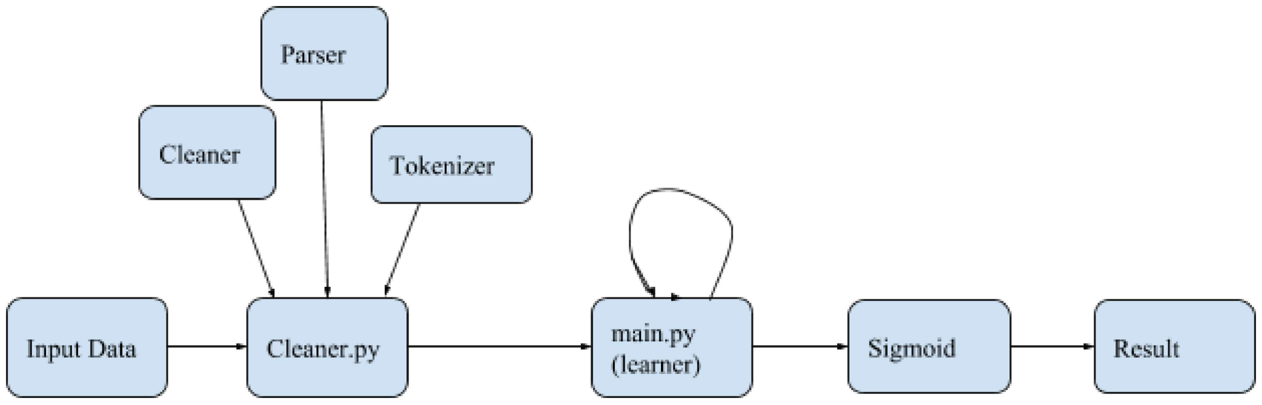
Data

The data set used was 25,000 IMDB reviews that were given a review rating from 1-4 and 6-10. In addition, there was a test set of 11,000 reviews that were left unlabeled and used for testing purposes. This data would be extracted by the program and then parsed in various ways in order to create the bag. Included is an example of a negative review that was given 1 star.



OK first of all the video looks like it was filmed in the 80s I was shocked to find out it was released in 2001. Secondly the plot was all over the place, right off the bat the story is confusing. Had there been some brief prologue or introduction the story would've been better. Also I appreciate fantasy but this film was too much. It was bizarre and badly filmed. The scenes did not flow smoothly and the characters were odd. It was hard to follow and maybe it was the translation but it was even hard to understand. I love Chinese epic films but if you're looking for a Chinese epic fantasy film i would recommend the Promise (visually stunning, the plot is interesting and good character development) not this film. Beware you will be disappointed.

Evaluation



This is the general overview of our regression mechanism. The raw data was first run through a cleaner which would remove things like various html tags as well as converting all of the words into lowercase to normalize the input data. In addition, it replaced common punctuation like commas with a space so that it could act as a delimiter for the tokenizer. Using the BeautifulSoup python library, all of the html tags were removed, as they were needless noise in the inputs.

Then, the tokenizer would be run, which would extract each word and use it to build a counter of how many times each word was used as well as the values of the words. One issue with this method is that words like “isn’t” would be parsed into “isn t”, but was largely a non issue as isn is not an actual word and would be treated the same as isn't, should it have been input into the word vector in that way. Empty strings were also filtered out as those provided no value for the learner.

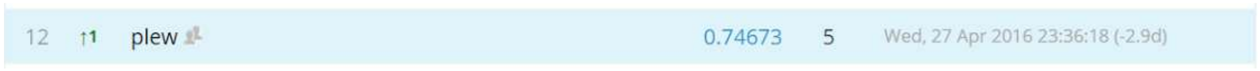
Common stop words were also included, since they are impactful of the final result of an analysis. This was important as words like "not" as well as all contradictions of that sort can be used to determine negation and changes the meanings of sentences and reviews entirely. In addition, research showed that stop words, while not effective for text classification, their inclusion was very good for sentiment prediction.

Finally, as a further option, we used regexes to filter out all non-alphanumerics. This was done because punctuation was discovered to be a largely ineffective training method. Our method would have treated "great" and "great!" as two different tokens, reducing the incidences and values of both of them. Removing punctuation also killed the inclusion of emojis that were used in some reviews, but the number of appearances of those were also too few to appear inside of the bag of features unless the bag itself was increased to contain a large number of features. The bag of features was decided to hold 5000 as this allowed it to contain the top 5000 words and removing the extremely small incidence words that could have skewed results. It was found that as feature size increased, it led to decreasing accuracy in the test set and increasing accuracy in the training set, which strongly implied that the data was overfitting the training set.

The bag of features was initially set up as a vector of the most popular features. The size was a hyperparameter that could be adjusted at any time. The initial weights were their incidences inside of all the reviews combined, and this value would be iterated upon as the data was being tested. When used for prediction, the cleaner function would clean the incoming review and create a vector of the number of times each word appeared. This vector was then multiplied with the bag of features and used as a predictor for sentiment.

Our method also included cross-validation of the data itself, with groups of reviews being given to the algorithm to build up the vector values. Afterwords, the results were always tested against the full training set in order to alway keep improving the value of the learner.

Furthermore, we opted to use the Broyden–Fletcher–Goldfarb–Shanno algorithm as it was shown to work very well on sets of data with a large number of features while remaining relatively quick to run and also being relatively light computationally. In addition, through research, it was determined that this algorithm was a fairly popular optimization algorithm when used for sentiment analysis.



12 ↑1 plew 0.74673 5 Wed, 27 Apr 2016 23:36:18 (-2.9d)

The above image shows the results that we achieved with a bag size of 5000, included stop words, no punctuation, and cross validation. The training set accuracy for this particular run was approximately 88%. We had varied the inclusion of punctuation as well as the size of the bag of words, but this was the one that had returned the best result.

Conclusion

In conclusion, sentiment prediction can definitely be done using a Naive bayes classifier algorithm that is based off of logistic regression and gradient descent. With an accuracy of approximately 74.5% on the test set, it is fairly close results that others using the same models have tried. There were also many possible improvements that could have been made. Looking into the inclusion of n-grams could have increased accuracy. Increasing the bag size in conjunction with a regularization parameter to prevent overfitting could also lead to possible improvements of the learner without sacrificing test set accuracy at larger bag sizes. Finally, there is also the case of the optimization algorithm. Although the BFGS algorithm is supposed to be good at large feature values and also light on computation power, this algorithm also ground processes to a halt on lower-end computers. As a result, this project was a good way to show the accuracy that a logistic regression model could receive.

Team Roles

Patrick Lehane - Responsible for testing different models, feature extraction, logistic regression model, optimizing features.

Eric Wang - Responsible for parsing training set, normalize all words (remove duplicates, lower case, etc.), and returning the data in a data structure.

References

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). *Learning Word Vectors for Sentiment Analysis*. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011) : 142-150. https://www.aclweb.org/anthology/P/P11/P11-1015.pdf

Wang, Sida, and Christopher D. Manning. (2012). *Baselines and Bigrams: Simple, Good Sentiment and Topic Classification.* 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012) : 90-94. http://www.aclweb.org/anthology/P/P12/P12-2.pdf#page=118