Classifying Sentiment in Survey Results

Zachariah Dunkerton
Department of Computer and Information Sciences
Towson University
Towson, MD
zdunke1@students.towson.edu

Abstract – In this document I describe my use of sentiment analysis to process data in text based surveys. I describe my web application which collects surveys, and my model which classifies them based on sentiment. I used traditional and new methods to classify the data, both yielding similar results. I conclude with an analysis of my classification models and application, and describe the benefits of using sentiment analysis in surveys.

I. Introduction

In this paper I present my research into sentiment analysis of survey results. I was working on developing a survey for hospice patients to assess their quality of care. The care providers used these surveys to make any needed adjustments to the care of the patient. The problem with this model is the potential number of surveys. For example, a supervisor may be responsible for 100 patients. If each patient fills out two surveys a day, and the supervisor checks these surveys every other day, they will have four hundred surveys to review. For text based surveys, this is not practical.

I attempted to solve this problem with sentiment analysis. If the surveys can be labeled as positive or negative, the administrator of the survey can then prioritize which surveys to address. In the example of patient quality of care, positive responses don't need to be addressed. Very negative responses however, need to be immediately addressed. Sentiment analysis aids in filtering these results. I created a web application which allows users to take a survey, and the administrator to see the responses as well as the sentiment of each response. This gives the administrator of the survey a useful resource to help track large amounts of data.

There has been much research in the field of natural language processing and sentiment analysis. My technique combines several previous methodologies and resources to form a new model. This new model is used to create a "Senti-Survey" which helps to solve the problem of large amounts of responses. This survey was not built specifically for patient quality of care. Rather it was designed as a general framework to showcase the usefulness of sentiment analysis in surveys, and to test the new methodology. In this paper I will describe the problem of sentiment analysis, give an overview of the application architecture, introduce my dataset and model, give the results of my model, and conclude with an analysis of my results.

A. Sentiment Analysis

Sentiment analysis, or opinion mining, takes a text document, and classifies it based on the tone of the document. For example, sentiment analysis would classify "That movie was terrible, what a waste of time" as negative, and "My word, that movie was fantastic, can't wait to see it again!" as positive. The challenges involved with sentiment analysis are subtle. An initial analysis of this problem may lead to an approache similar to other text based classifications: if the document has positive words in it, then its positive and visa versa.

While classifying text documents into categories by this method can achieve well over 90% accuracy, it performs significantly worse when classifying sentiment [1]. Achieving such a result in sentiment analysis is extremely difficult if not impossible. This can generally be attributed to the subtlety of language. A sentence can have many positive words and still be

negative. For example, “I love pizza, I enjoy a good salad, and I always crave milkshakes, but they were not up to scratch at this place.” In this sentence, the last phrase negates all the positive words. It is not enough to simply count the positive and negative words. Sarcasm is another instance where it is difficult for models to classify sentiment. Opinion depends very much on context, which the model does not always have access to. In summary, sentiment analysis is difficult because language is extremely complex and multifaceted, especially when opinions are being expressed. The problem is explored in more detail by Pang and Lee [1].

B. Literature Review

An overview of the field of sentiment analysis was given by Pang and Lee in 2008 [1]. They give a very good description of the problem of sentiment analysis and some of the solutions. Pang, Lee and Vaithyanathan also produced the original dataset used in my experiments, including their original 2002 experiments which were some of the first in sentiment analysis [2].

Two categories of research can be seen within sentiment analysis. The first involves the bag of words approach with classification algorithms, mostly Support Vector Machine and Naïve Bayes, to classify the data. This originated with Pang et al [2]. For example, Poursepanj et al. take data from Twitter, add additional information, and classify the data with SVM and Naïve Bayes [3]. Twitter data is commonly used in sentiment analysis because it is readily accessible, and because classifying social media data has many uses and applications. This is the approach taken in my experiments.

The second field of research is examining different ways to capture the complexity of language, rather than just looking at word counts. One of the best examples in the Stanford Sentiment Tree Bank, which iterates deep learning models over a sentiment tree [4]. This method departs completely from original methods of Pang and Lee, but better captures some of the finer points of language by using more complex approaches and data structures. In summary, many researchers are still using bag of words, but new techniques have been emerging which can breakthrough the threshold of these traditional approaches.

II. METHODOLOGY

A. Web application architecture

My approach to building the application was to first create a front-end website that handled both entry of surveys by users, and analysis of the results by administrators. A back-end server was then created to parse and classify the data from the users, and publish that data to the administrators. Any user can take a survey, but only users with credentials can view the results. The server portion of the application stores both the survey results, and the model for classifying those results.

The survey consisted of five questions with text fields allocated for the user’s answer. Once the survey was submitted, each response was passed through the model and classified, either 1 for positive, or -1 for negative. These response scores were then added to produce the overall survey sentiment score. Scores greater than or equal to +3 were classified as positive, scores less than or equal to -3 were classified as negative, and scores of -1 or +1 were classified as neutral. For example, a survey with four negative responses and one positive response would yield a sentiment score of -3, and the survey is classified as negative.

These results are published from the server to the administrator in several ways. First, in a table which displays all surveys that have been taken. The administrator can filter and search through these results. Second, they are given a dashboard that shows the sentiment of the most recent surveys, a count of positive, negative and neutral surveys, and a graph that shows the number of surveys by sentiment over the past week. This dashboard is meant to allow the administrator to quickly gauge the general

sentiment of incoming surveys. The web application was implemented using Meteor.js [5].

B. Model approach

My approach in developing the classification model builds upon an area of research within data science: keywords. Pang and Lee how shown that a good level of accuracy can be achieved by simply looking at keywords within documents [1]. This idea was expanded upon by Poursepanj et al. in their research into classifying Twitter data [3]. They identified a keyword which related to sentiment in each individual document, and appended that word's definition to the document. This method increases the number of words the model associates with a particular sentiment. This significantly improved the accuracy of their models. This is an effective approach, however it requires the additional preprocessing step of identifying keywords within the document.

My approach identifies the keyword in the document automatically using SentiWordNet and a list of positive and negative words [6]. I had three methods of identifying keywords and appending their definitions and synonyms. Both however took the key word(s) and added their definition and synonyms to the document.

C. Dataset and preprocessing

My models were based on a set of movie reviews, which were labeled by sentiment. The original set of reviews came from Pang and Lee [2]. Socher et al took this set and parsed it into phrases, and then labeled each phrase by hand [4]. The ratings were very negative (1), somewhat negative (2), neutral (3), somewhat positive *(4)*, and very positive (5). I chose to use this dataset because it rated the severity of the sentiment, rather than just having positive and negative results. For my experiments, I only used the full documents, and not their parsed phrases.

While the survey I created was not exclusively for movie reviews, the only way to get data for a specific survey is to collect it. I did not have the time or resources to collect data for one specific type of survey, so I had to choose preexisting data. This will be discussed in more depth in later sections.

The first step in preprocessing was to add negative documents to the dataset so the same number of positive and negative documents were present in the training data. These additional responses were randomly chosen from the original dataset. In total 330 negative responses were duplicated in the final dataset. In total there were 3602 positive documents, and 3602 negative documents.

The second step was to make the additions to each individual document. This was done in one of three ways. The first two methods started by parsing each document into individual words. These words were then compared against a list of positive or negative words, depending on the sentiment of the document. These lists were compiled by Hu and Liu [7]. If the document was positive and if the word was contained in the positive list, then it was eligible to have its definition appended to the document.

These eligible words were looked up in SenitWordNet. SentiWordNet has multiple definitions for some words, based on context and part of speech. In the first method, all forms of each word were considered when making these additions, so one word can contribute multiple definitions to a document. Definitions and synonyms were added if their sentiment score met certain criteria. For positive documents, a word's positive score needed to be greater than 0.5, negative score equal to 0.0. For negative documents, a word's negative score needed to be greater than 0.5, positive score equal to 0.0. In the second method, only the highest scored definition was added.

In other words, for both methods positive/negative words have their definitions and synonyms added to the document. In the first method, they are added if they meet a criterion, and in the second only the best is

added. In the final method, words were not compared against a list of positive and negative words. Rather, all forms of each word in SentiWordNet were considered. Only the word with the highest score had its definition and synonyms added.

After the additions were made, the documents were stemmed. This process attempts to reduce the number words in the corpus by removing the ending of some words so that different forms of the same word become the same word again. For example, take the words "stemmed", "stemmer" and "stemming". The stemmer coverts these words back to "stem" so that the model is not clogged up by different words with the same meaning.

D. Model Implementation

This model uses a "bag of words" approach, meaning, grammar, word order, punctuation etc. are discarded, and documents are looked at as a collection of words. To represent these words in a way a model can understand, Term Frequency Inverse Document Frequency (Tf-idf) was used. Each word is given a Tf-idf score, normalized between 0 and 1, which is the term's frequency in a document, times idf, which is given in Eq. 1 below, where n_d is the total number of documents and df(d,t) is the number of documents that contain the term t. This is then normalized, and each term is assigned this value.

$$\text{idf}(t) = \log\left(\frac{1 + n_d}{1 + \text{df}(d, t)}\right) + 1 \qquad (1)$$

This transformation was performed by the TfidfVectorizer module from sklearn [8]. This module also removed stop words, which are terms that appear frequently and don't affect the meaning of a document, such as "the" "a" etc. It also adds "n-grams," which count n number of words as a term. This helps capture phrases that only have meaning if the words are together rather than separate. For example, if the document consisted of "I like n-grams," the bi-grams method would have "I like" and "like n-grams" as two additional terms rather than just "I", "like", and "n-grams" as three separate terms.

These terms were then fed into a Support Vector Machine, and Multinomial Naïve Bayes, both of which were implemented in sklearn [8]. The SVM was sklearn's Linear SVC, which is a SVM with a linear kernel. As the results section will show in more detail, the Naïve Bayes implementation outperformed the Linear SVC algorithm, and was the algorithm implemented in the survey.

The algorithm and vectorized count of words were saved and passed to the server. This only happens once. The application then passes any new survey values to the server. The values are parsed and passed through the Tf-idf vectorizer so that the words are given a numeric value as described above. They are then passed through the already fit model and classified. This value is then added to the data base and published to the user.

III. RESULTS

For each algorithm, the data was randomly split into 70% training data and 30% testing data. I used accuracy, precision, recall and f-1 score as my evaluation metrics. The full set of results for the Multinomial Naïve Bayes are shown in Fig 1, and results for Linear SVC in Fig. 2. In the linear SVC tests, the additions decreased the accuracy of the model significantly. For the Naïve Bayes model, all the results were relatively consistent. For both models, the baseline yielded the best results. The Naïve Bayes model outperform the Linear SVC, and was the model implemented in the web application. Method three appears to give the best performance of the three.

Naïve Bayes	Accuracy	Precision	Recall	F-1 Score
Baseline	0.79	0.79	0.79	0.79
Method 1	0.77	0.77	0.77	0.77
Method 2	0.778	0.78	0.78	0.78
Method 3	0.786	0.79	0.79	0.79

Fig. 1: Results Multinomial Naïve Bayes

Linear SVC	Accuracy	Precision	Recall	F-1 Score
Baseline	0.78	0.78	0.78	0.78
Method 1	0.72	0.72	0.72	0.72
Method 2	0.708	0.73	0.71	0.7
Method 3	0.72	0.73	0.72	0.72

Fig. 2: Results Linear SVC

IV. Conclusion

A. *Classification Model*

My methodology for adding additional definitions and synonyms to opinionated words was not successful. In one model it decreased performance significantly, and in the other it had no significant impact. A potential reason for this is that the baseline model is overfit. Adding more terms generalizes the model and increases the bias but decreases the variance leading to the decrease in accuracy.

Another possibility for no increase in accuracy is the complexity of language as described above. The movie reviews that I used to test my model are full of subtle references, sarcasms etc. that determine the sentiment. It does not matter how many positive or negative words, synonyms, or definitions the model is trained for; some documents are not classified by the sentiment of the individual words. Even by implementing bi-grams and tri-grams these methods were not affective.

B. *Further research*

The web application was successful in taking survey input and classifying to for the administrator. As a concept, the "Senti-Survey" was successfully implemented. A real-world test is needed to determine how useful it is for an actual user.

The main area where this concept can be improved is making specific, rather than general classification models. Ideally, the administrator would label surveys, and this data would be used to create the model. This would create a model that best fits the specific nuances of that survey. Once the model is created, the administrator could then correct misclassified data to further improve the model.

Further research into the idea of additions to documents is needed. Much success has come from adding features to Twitter data (see Agarwal et al. [9]). My method was to add definitions and synonyms of opinionated words as the features, but this did not improve performance. More research is needed into what additional features do improve accuracy for plain text documents.

In conclusion this project served as a proof on concept for a sentiment analysis within surveys. This application shows useful and pertinent information about incoming surveys which the administrator otherwise would not have. Once integrated into a traditional survey, it serves as a viable metric in helping administrators see trends in their data, as well as address individual cases. It gives them a powerful tool to analyze the sentiment of their users, and will extract useful knowledge from overwhelming amounts of data.

REFERENCES

[1] B. Pang, and L. Lee, "Opinion Mining and Sentiment Analysis," Retrieved from http://www.cs.cornell.edu/home/llee/omsa/omsa-published.pdf

[2] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment Classification using Machine Learning Techniques," Retrieved from https://www.cs.cornell.edu/home/llee/papers/sentiment.pdf

[3] H. Poursepanj, J. Weissbock, and D. Inkpen, "uOttawa: System description for SemEval 2013 Task 2 Senitment Analysis in Twitter." Proceedings of the Seventh International Workshop on Semantic Evaluation, Vol. 2, pp. 380-383, June 14-15, 2013, Atlanta, Georgia, USA

[4] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive Deep Models for Semantic Compositionality Over a Sentiement Tree Bank," Retrieved from https://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf

[5] Meteor.js https://www.meteor.com/

[6] A. Esuli, F. Sebastaini, "SentiWordNet: A Publically Availible Lexical Resource for Opinoin Mining," Retrieved from http://nmis.isti.cnr.it/sebastiani/Publications/LREC06.pdf

[7] M. Hu, and B. Liu, "Opinion Lexicon," Retrieved from https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[8] Sklearn http://scikit-learn.org/stable/

[9] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, R. Passon, "Sentiment Analysis of Twitter Data," Workshop on Language in Social Media, pp. 30-38 , 23 June, 2011 Portland, Oregon, USA