Sentiment Analysis And Tag Recommendation For Online MultiMedia

Afshin Moin (afshinm), Abhishek Bharani (abharani), and Peng Seng Kuok (pkuok)

1 Introduction

Content sharing has been the key ingredient of the biggest success stories in the last decade. Social networks like Facebook and LinkedIn, video sharing services like YouTube, Ecommerce websites like Amazon and online travel agencies like TripAdviser and Expedia are all examples of online platforms that provide a convenient way for their users to exchange content, goods or services. It is obvious that the success of such online platforms depends on how effectively they link the end users to relevant content. This latter cannot be achieved without proper content classification and user feedback analysis. This motivated us to apply machine learning techniques to automate the process of user feedback analysis and content classification. We work with a real sample dataset taken from YouTube [3]. Nevertheless, same techniques can be applied to any other database with similar structure like Yelp and Netflix.

2 Sentiment Analysis

We apply classification techniques on a sample dataset taken from YouTube. The dataset can be downloaded from [3]. It includes 691407 comments for 200 most trending YouTube videos in US in a two weeks period. The same source also offers a similar dataset for the most trending videos in GB. We will use a fraction of this dataset as development set. The data contains video title, channel title, category, tags, number of views, likes and dislikes and user reviews. In this section, we first explain our methodology to classify comments into three categories, positive, neutral and negative. Then, we apply a number of classification methods on the data and compare their performance with each other.

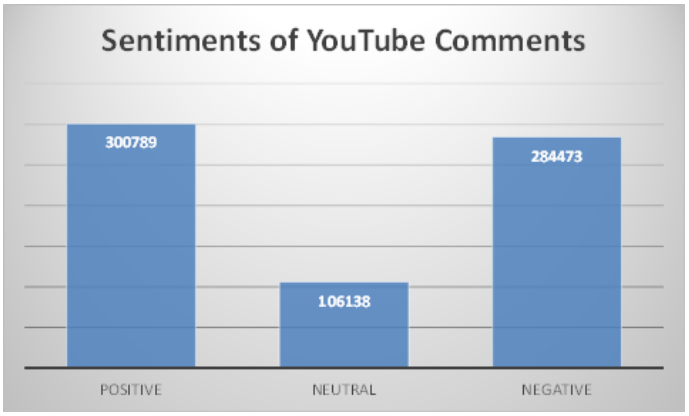

Fig. 1. YouTube Dataset Statistics

2.1 Label Generation

Our goal is to apply classification techniques on the YouTube data in order to understand whether a new comment bears positive or negative sentiment. However, there are no labels classifying the comments based on their sentiment. Considering the large number of comments, it was not possible for us to manually label a reasonable fraction of this dataset. Consequently, we used TextBlob [2] library for Python to generate the labels. TextBlob applies Natural language Processing techniques to estimate the **polarity** of each comment independently from the rest of the comments, and only based on its content. For us, it replaces the burdensome task of manual labeling. TextBlob generates polarity scores in the range of $(-1, 1)$. We convert them into categorical data using Equation 1. Statistics on the size of each class is shown in Figure 1.

$$\text{sentiment} = \begin{cases} -1 & \text{polarity} < 0 \\ 0 & \text{polarity} = 0 \\ 1 & \text{polarity} > 0 \end{cases} \tag{1}$$

Words that happen more frequently in positive and negative comments are shown as word clouds in Figures 2 and 3.

Fig. 2. Word cloud for positive comments.

Fig. 3. Word cloud for negative comments.

2.2 Feature Extraction

We use TF-IDF to convert comments into feature vectors. In this technique, stop words are first removed and the remaining words are stemmed. Stop words are commonly used words such as *the* which are filtered out before the analysis. Stemming is the process of reducing derived words to their common root known as stem. For instance, *going* and *goes* both come from the stem *go*. TF (Term Frequency) counts the number of times a word has occurred in each document. However, the number of occurrences of words is in general higher for longer documents. Then, it is helpful to divide the number of word occurrences by the number of words in the document. The output of TF is a sparse matrix mapping documents to a normalized vector representing how many times each word has occurred in them. IDF (Inverse Document Frequency) accounts for the number of occurrences of a word in the entire corpus. Namely, some words are more likely than others to happen in a given corpus of documents. Then, we scale TF terms by a decreasing function of the number of occurrences of each word in the whole corpus which is indeed the IDF term.

2.3 Experiments and Evaluation

In order to measure the performance of different classification techniques on the YouTube data, we split the data from US comments into train and test subsets. To generate the train set, we randomly choose 80% of the comments from each of the three categories. The remaining comments form the test set. We use 20% of the GB comments as development set. We ran three different classification techniques on the data, including Ridge Classifier, Bernoulli Naive Bayes and SVM. Table 1 shows the accuracy measured on test and dev sets. Naive Bayes classifier has the worst performance among all methods. This result is predictable as the conditional independence condition required by the Naive Bayes classifier is not satisfied in this dataset. In particular, the probability of two words happening in a comment given the category of the comment are not independent from one another. On this dataset, SVM classifier has the best performance.

Model	Test Accuracy	Dev Accuracy
Bernoulli NB	0.709	0.715
Ridge Classifier	0.914	0.905
LinearSVC	0.956	0.951

Table 1. Accuracy of different classification techniques on YouTube comments.

3 Future Work

For future work, we are planning on applying SVM with different Kernels and compare the performance of them. Also, since the dataset seems easy to predict with 80% of the comments as train set, we will study the accuracy of clasification methods on different proportions of train to test. We have also other similar datasets like Yelp on which same classification techniques can be applied.

Another possible direction for future work is tag recommendation for different video categories. Tags are an important type of data that can be used by search algorithms to enhance the quality of results. If time permits, we will predict the category of a video given its tags. Using a generative method, the distribution function of tags given video category is computed. This distribution function may be used for automatic tag recommendations. Same techniques can be used to predict the distribution of tags based on review features. A related work about tag recommendation and category discovery can be found in [1].

4 Contributions

All of us contributed in the discussions about what problem to target, what dataset to use and what techniques to apply. All of use helped with writing and reviewing the report. Peng searched different potential dataset candidates and summarized their advantages and shortcomings. He generated statistics about the YouTube dataset. Abhishek did sentiment polarity analysis and generated word clouds using TextBlob. Afshin implemented feature extraction and conducted classification experiments using scikit.

References

1. G. Toderici, H. Aradhye, M. Pasca, L. Sbaiz, and J. Yagnik. Finding meaning on youtube: Tag recommendation and category discovery. In *Computer Vision and Pattern Recognition*, 2010.
2. YouTube. TextBlob Library For Python. https://textblob.readthedocs.io/en/dev.
3. YouTube. YouTube Database For Kaggle Competitions. https://www.kaggle.com/datasnaek/youtube.