Forecasting Popularity of Music Albums using Sentiments

Kavinya Rajendran
Ira. Fulton School
of Engineering
Arizona State University
Tempe Arizona 85281
Email: krajend2@asu.edu

Nitesh Thali
Ira. Fulton School
of Engineering
Arizona State University
Tempe Arizona 85281
Email: nthali@asu.edu

Thara Sridar
Ira. Fulton School
of Engineering
Arizona State University
Tempe Arizona 85281
Email: tsridha1@asu.edu

Manoj Rajandrakumar
Ira. Fulton School
of Engineering
Arizona State University
Tempe Arizona 85281
Email: mrajandr@asu.edu

Bhuvana Lalitha Namasivayam
Ira. Fulton School
of Engineering
Arizona State University
Tempe Arizona 85281
Email: blnamasi@asu.edu

Vishal Johri
Ira. Fulton School
of Engineering
Arizona State University
Tempe Arizona 85281
Email: vjohri@asu.edu

Prasanth Priya Nesan
Soundra Pandian
Ira. Fulton School
of Engineering
Arizona State University
Tempe Arizona 85281
Email: psoundra@asu.edu

***Abstract*—User reviews and sentiments expressed on the web are mined, analyzed and used to predict future outcomes in various domains such as product sales, restaurant popularity, identifying ideological bias, targeting advertising and so on. In our project, we intend to predict the popularity of music album based on user sentiments mined from YouTube[8] comments.**

***Index Terms*—Sentiment analysis, Popularity Prediction, Appraisal Words, Data Cleaning, Probabilistic Latent Semantic Analysis, classification, Mutinomial Logistic Regression**

I. Introduction

With the advent of web, people are able to express their opinion online about almost everything. Online reviews are available for every product or service, ranging from electronics and restaurants to artists, movies and albums. These opinions can be extracted from various web sources like E-commerce sites, social media and blogs. Sentiment analysis provides a way to classify these opinions into positive, negative, or neutral. There is wide range of applications in sentiment analysis. Ronen Feldman[7] lists "Reviews of consumer products and services", "monitoring the reputation of a specific brand on Twitter and/or Facebook", "Provide substantial value to candidates running for various positions", "Financial Markets" as few popular applications of sentiment analysis. In our project we have done popularity prediction of videos by scrapping comments from YouTube and analyzing them. In our paper we will first define the problem definition. Then we will focus on our approach for popularity prediction. Consequently, we define data sets and results evaluation.

II. Problem Definition

Of the limitless application of sentiment analysis, it is evident that analyzing user sentiments not only helps in making wise decision in product selection, but also assists the manufacturers to retrospect and improve their future releases. With that hint, we chose to analyze the popularity of music albums based on the user reviews on YouTube. The intention of the project is to compute a popularity factor for music videos based on user sentiments mined from their comments on YouTube. This popularity factor can be used for the betterment of the bands performance on their upcoming albums.

III. Approach

Our approach involves the steps of scraping YouTube Data, Data Cleaning, Sentiment Analysis using S-PLSA and Popularity Prediction. These steps are described in detail in this section. The block diagram of our implementation is depicted in the figure 1.

A. Scraping YouTube Data

User Comments for music videos are scrapped from YouTube in this step of our approach. YouTube comments scrapping is executed in two phases:

1) Phase 1: The first phase involves extracting 50 YouTube URLs for a given set of 30 search term (name of the albums) using YouTube API[9]. YouTube API provides facilities to create server, browser, IOS or android keys according to developers use. A unique browser key is generated to be used as a developer key for accessing YouTube data. The output of this phase would be a text file with list of URLs for YouTube video, whose comments are targeted for this project.

2) Phase 2: Phase two of scrapping involves scrapping the comments from the list of URLs scrapped in phase one. Although YouTube API V3[9] could be used to extract the

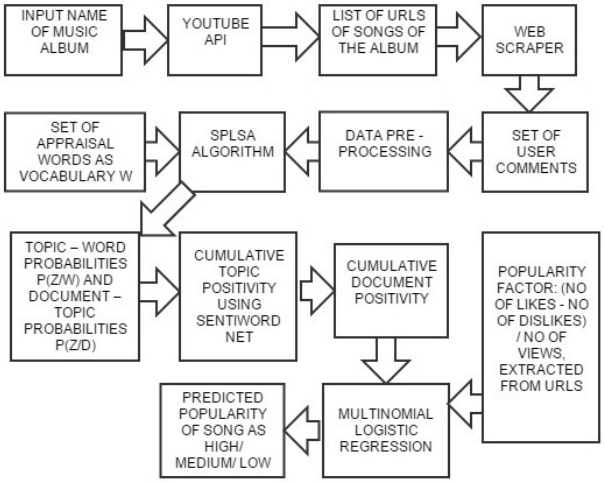

Fig. 1: Block Diagram of our Approach

URLs, it cant be used to scrap the comments as this method places a restriction on number comments that can be retrieved per call. To resolve this problem, we use Selenium to automate the "Show more" button click until there exists one. On doing this, we get to view the entire set of comments for the targeted videos, which can be scrapped and stored in a form of text documents using a python library called Beautiful Soup. These text documents resulting from phase two are fed as input to the data preprocessing module.

B. Data Cleaning

The comments obtained from YouTube contains spelling mistakes, short forms of words, foreign languages, emoticons and irrelevant comments. The data obtained is noisy and hence requires cleaning before we can arrive at a meaningful conclusion from them. Cleaning is done so as to remove redundant and unwanted data by manipulating the available data and converting it into a meaningful form so that it can be analyzed to extract important information.

The comments after extraction are put into files and are processed to remove hyperlinks by comparing the comments with regular expressions. The special characters like emoticons, blank spaces, punctuations and new line characters are removed using the regular expression comparison.

The spelling correction is done by comparing them with the spelling corrector suggested by "Peter Norvig" [4]. Here the words are taken from the comments and are checked if they can be corrected using one or two data manipulations(edit distance) by comparing them with the standard literature. The distance of the misspelled word from the correct word is calculated. The words with one and two letter edit distance are generated and then compared to see if they exist in the standard literature. The word with the maximum frequency is taken as the replacement for the wrong word.

The spam removal from the comments is done using the "Nave Bayes Classification" algorithm from the NLTK library [5], where 1700 comments were used to train the model. The feature set to the model consisted of top 1000 words extracted from input training data. The model then classifies the comment as ham or spam based on the posterior probability of each class. Model assigns a class to a new comment based on the maximum probability between class based on the features present in the comment. The model was evaluated by using 800 labeled comments. The results of the classification are compared with the manually labeled test data and the percentage of the correctly labeled data is used to account for the precision, recall and accuracy of the Nave Bayes Classifier.

C. Sentiment Analysis using S-PLSA

1) Goal: The goal of this module is to generate topic and document positivity from a clean set of YouTube comments. We use the hidden sentiments of the YouTube comments, to determine the overall sentiment for the video.

2) Input: A set of Appraisal Words is taken as feature set. There are totally 6800 appraisal words in this set, containing both positive and negative words[3]. Cleaned comments from YouTube video are denoted as documents and given input to S-PLSA algorithm.

3) Algorithm: S-PLSA executes by constructing a Document-Term matrix, where each clean YouTube comment is represented as an individual document[11] and the term is set of appraisal words. This matrix determines the number of times a term occurs in each document. For each YouTube video, S-PLSA is given a set of N comments $D = d_1, d_2, ..., d_N$ and set of M appraisal words $W = w_1, w_2, ..., w_M$. It starts executing by constructing a Document-Term matrix, where each clean YouTube comment is represented as an individual document and term is set of appraisal words. This matrix determines the number of times a term occurs in each document, represented by $c(d_i, w_i)$. The hidden sentiment is determined by a set of K latent sentiment factor $Z = z_1, z_2, ..., zk$, where K is the number of topics desired (11 for this project). The algorithm computes three set of probabilities term in topic as $P(w|z)$, topic in document as $P(z|d)$ and topic probability as P(z)[1]. Starting with random probability, their likelihood is maximized by iterating four times using Expectation Maximization (EM) algorithm as given in the following two steps:

i) Expectation Step: In this step posterior probabilities for the latent variable(z) are calculated based on the current estimates of the parameter[1].

$$P(z_k|d,w) = \frac{P(z_k)P(d|zk)P(w|z_k)}{\sum_{z'_k \in Z} P(z'_k)P(d|z'_k)P(w|z'_k)}$$

ii) Maximization Step: In this step estimates for the parameters are updated to maximize the complete data likelihood[1].

$$P(w|z_k) = \frac{\sum_{d \in D} c(d,w)P(z_k|d,w)}{\sum_{d \in D}\sum_{w' \in W} c(d,w')P(z_k|d,w')}$$

$$P(d|z_k) = \frac{\sum_{w \in W} c(d,w)P(z_k|d,w)}{\sum_{d' \in D}\sum_{w \in W} c(d',w)P(z_k|d',w)}$$

$$P(z_k) = \frac{\sum_{d \in D} \sum_{w \in W} c(d,w) P(z_k|d,w)}{\sum_{d \in D} \sum_{w \in W} c(d,w)}$$

After the model is constructed, we can calculate $P(z_k|d)$ using Bayes rule:

$$P(z_k|d) = \frac{P(d|z_k)P(z_k)}{\sum_{z'_k \in Z} P(d|z_k)P(z_k)}$$

In the figure 2, d denotes document, z denotes topic, w denotes appraisal word, N denotes number of words and M denotes number of documents.

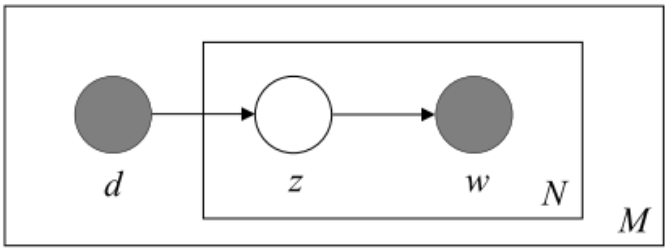

Fig. 2: S-PLSA

If there exists a term (appraisal word) that is not present in any of the document, the probability of that term in any topic would be 0. To resolve this issue, S-PLSA term set would be the intersection of appraisal words and words present in all the document. Also, if there is a document that doesnt have any appraisal word, the probability of any topic related to that document would be 0. To ensure this does not happen, any comment that doesnt have even one appraisal word is not added to the document list.

4) Implementation: The implementation of PLSA is in python language[2]. The feature set of this implementation is Bag of Words (words present in all the documents), which is modified to take appraisal words as its feature set. The code is also modified to ensure that only comments with at least one appraisal word is present in the document list. We place a restriction in removing stop words from the documents so that appraisal words are not removed as a part of stop words. Hence, like this we modified this PLSA algorithm to work as S-PLSA. We are running the algorithm for 11 topics and 4 iterations of EM steps.

All the probabilities for term in topic $P(w|z)$ and topic in document $P(z|d)$ are arranged in descending order and written in separate files. After this following two steps are used to calculate positivity of individual document:

i) SentiwordNet corpus provided by nltk[5] is used to calculate positiveness of each Topic, denoted by TP

$$TP = \sum_{k=1}^{K} Pr(Word|Topic)(pos\_score - neg\_score)]$$

where pos_score and neg_score for each topic is taken from SentiwordNet.

ii) Document positiveness is calculated by aggregating the positiveness of each of the contributing topics, denoted by DP

$$DP = \sum_{k=1}^{K} Pr(Topic|Document)(Topic\ Positiveness)]$$

5) Output: The output of S-PLSA algorithm is the positiveness of each document given as input.

6) Validation: The efficiency of the S-PLSA algorithm relies on taking the appraisal words as the feature set instead of taking Bag of Words. Bing Liu and Minqing Hu have proved that having these 6800 appraisal words as feature set for sentiment, improves the accuracy of the algorithm[3].

D. Popularity Prediction

For predicting the popularity of a music album with better accuracy, we use both the sentiment value of the songs which is the output of SPLSA algorithm and the ratio of the difference between number of likes and number of dislikes of the song to the total number of views of the song. Based on these features, we classify the popularity of the song as $High$, Low or $Medium$.

1) Training, Testing and Prediction: User comments are scraped for approximately 1500 songs from YouTube and hidden sentiment value is calculated for each song by S-PLSA. For each of these songs, the corresponding number of likes, number of views and number of dislikes are scraped from YouTube. We used 1123 instances to train and test our data. We are using the following formula to calculate the popularity factor:

$$PF = \frac{Number\ of\ likes - Number\ of\ dislikes}{Number\ of\ views}$$

Popularity Factor is calculated for each song and we assign the class labels $High$, Low or $Medium$ based on this value for training our classifier. Popularity factor value calculated for each song is multiplied with the corresponding sentiment factor output from SPLSA to obtain the feature vector for classifier. Target Value of Classifier: $High$, Low or $Medium$.

Assigning Class labels: We observed the popularity factor values for multiple songs and divided it into three ranges (less than 0.005 , between 0.005 and 0.01 and greater than 0.01) so that the three class labels have approximately equally distributed instances under them.

70% of our instances are used for training and 30% for testing. Finally, we predict the target value, that is, popularity as $High$, Low or $Medium$ for a new song.

2) Choosing the Right Classifier: Classification is the problem of identifying to which of a set of categories (subpopulations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known[10].

Accuracy is one of the prime factor which helps in determining the performance of the classifier. We trained our data using different classifiers: Decision Trees, Random Forest and Multinomial Logistic Regression and compared the accuracy.

Based on the performance of from each classifier, we found that multinomial logistic regression performs better than the others. Also as our data is linear, it makes Logistic Regression a good fit for our data and we use it to predict the song popularity. We have used python sklearn package and weka for implementation and testing of our classifier.

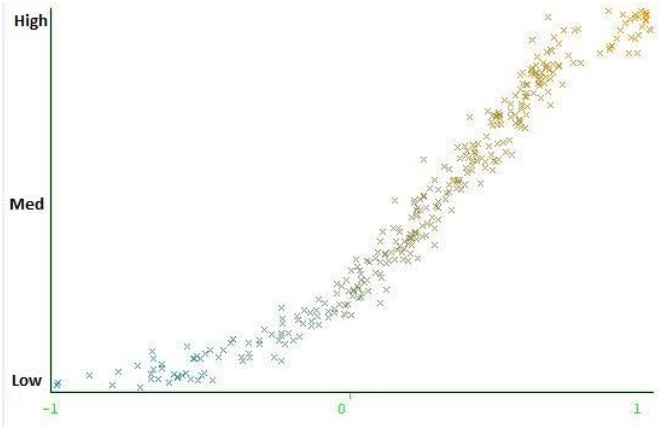

Fig. 3: Plot of Training Data for Multinomial Linear Regression

In the figure 3, we have shown the plot of training data for Multinomial Linear Regression. The graph represents the model constructed using the training data. The training data is classified into three categories: $High$, $Medium$ and Low. For any song for which the prediction has to be made, it's sentiment value multiplied by popularity factor (feature vector), which is plotted in the x-axis, is extended to the curve and the corresponding class label is given as the predicted popularity.

IV. Data Sets

For data cleaning, YouTube labelled comments with 2,500 comments are used as training and test dataset, out of which 800 comments are used as test dataset. A set of Appraisal Words is taken as feature set. There are totally 6800 appraisal words in this set, containing both positive and negative words[3]. SentiwordNet corpus provided by nltk[5] is used to calculate positiveness of each Topic. To test the scheme we developed, the large movie review dataset provided by Stanford [6] is used. We have scrapped comments for approximately 1500 url videos from YouTube. After data cleaning, we had 1123 urls for sentiment analysis. Out of these 1123 urls, we are using 70% for training and remaining 30% for testing. Each of the url in this list consists of 2500 user comments.

V. Results Evaluation

Below are the precision, recall and accuracy values for each classifier:

Table 1: Performance Analysis of Classifiers

Classifier	Precision	Recall	Accuracy
Multinomial Logistic Regression	77	75	75
Random Forest	63	64	64
Decision Tree	75	73	73

The precision, recall and accuracy values obtained for the Decision Tree and Multinomial Logistic Regression classifiers are quite similar but Multinomial Logistic regression is slightly more accurate in predicting the popularity of a new song.

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.649     0          1         0.649     0.787      0.873    High
                 0.489     0.078      0.698     0.489     0.575      0.78     Medium
                 0.946     0.377      0.716     0.946     0.815      0.844    Low
Weighted Avg.    0.755     0.21       0.777     0.755     0.744      0.834

=== Confusion Matrix ===

   a   b   c   <-- classified as
  50  10  17 |   a = High
   0  44  46 |   b = Medium
   0   9 159 |   c = Low
```

Fig. 4: Classification and Confusion Matrix for Multinomial Logistic Regression

As depicted in the figure 4, we obtain precision and recall values for each of our class labels when predicting popularity for test data.

VI. Conclusion and Future work

In this project, we implemented S-PLSA algorithm to extract the sentiments from YouTube comments and use it as input to Multinomial Logistic Regression, Random Forest and Decision Tree for predicting the popularity of a new video. Although the results obtained from Mutlinomial Logistic Regression and Decision Tree are quite close, Multinomial Regression model is slightly better. The project also included scrapping comments from YouTube and perform data preprocessing for providing clean data to the S-PLSA algorithm. The experimental results prove the validity of the chosen model. For future work, the sentiment analysis could be made adaptive using Latent Dirichlet Allocation (bayesian algorithm)[1]

References

[1] Yang Liu, Xiaohui Yu, Aijun An, Xiangji Huang : Riding the tide of sentiment change: sentiment analysis with evolving online reviews. In: WWW '13, pp.477-496 (2013)
[2] https://github.com/hitalex/PLSA
[3] https://www.cs.uic.edu/ liub/FBS/sentiment-analysis.html
[4] http://norvig.com/spell-correct.html
[5] http://www.nltk.org/
[6] http://ai.stanford.edu/ amaas/data/sentiment/
[7] Lipika Dey Sk. Mirajul Haque Opinion mining from noisy text data AND '08 Proceedings of the second workshop on Analytics for noisy unstructured text data.
[8] https://www.youtube.com/
[9] https://developers.google.com/youtube/v3/?hl=en
[10] https://en.wikipedia.org/wiki/Statistical_classification
[11] Hofmann, T.: Probabilistic latent semantic analysis. In: UAI99, pp. 289296 (1999)