Proceedings of the Fifth International Conference on Communication and Electronics Systems (ICCES 2020)
IEEE Conference Record # 48766; IEEE Xplore ISBN: 978-1-7281-5371-1

Movie Recommendation System Using NLP Tools

Nimish Kapoor
Department of Computer Science and Engineering ,
Amrita School of Engineering,
Amritapuri, Amrita Vishwa Vidyapeetham, Amrita University, India
Email:nmshkpr@gmail.com

Saurav Vishal
Department of Computer Science and Engineering,
Amrita School of Engineering,
Amritapuri, Amrita Vishwa Vidyapeetham, Amrita University, India
Email:sauravvishal8797@gmail.com

Krishnaveni K S
Department of Computer Science and Engineering,
Amrita School of Engineering,
Amritapuri, Amrita Vishwa Vidyapeetham, Amrita University, India
Email:krishnaveniks@am.amrita.edu

***Abstract*—Movie industry has been booming ever since early days.But not all movies are great and worth users' time. Hence people depend a lot on movie reviews before watching a movie. Classically movies are rated on the basis of rating score.And in addition users also provide comments for review. But the reviews aren't made full use of to add to its rating score and recommendations. And people are very short of time these days so comments are hardly read. So, sentiment analysis are applied on reviews/comments so that they can also add to the ratings of a movie and hence result in better recommendation. In this project, a Sentiment analysis system is built based on the TMDB dataset and used SVM to predict positive negative sentiment from the user's movie review and then movies are rated based on the sentiment analysis and scoring of the reviews.**

Keywords—SVM, KNN, Text Mining, Polarity Scoring, Android App

I. INTRODUCTION

An android application is built which has movie data of many movies. It will display varied categories of movies to the user.

Users can input new ratings, reviews. Users can maintain a favourite list of movies. The application will be interactive. Users can also watch trailers of the movie. The main aim of the application is to give rating to movies based on sentiment analysis of user review. The user will add the review and the review will be sent to a server which will process the review based on the SVM model and classify it into positive or negative. And then the rating of the movie will be updated in the database.

Sentiment analysis is the prediction of emotions in a word which is part of a sentence. It aims to serve as an application to understand attitudes, opinions and emotions. The intention is to gain an overview of the wider public opinion behind certain topics. Precisely, it is a model of categorising conversations into positive, negative or neutral labels. Many people use social media sites for socialising with other people and to stay up-to-date with news and current events [2]. These sites (Twitter, Facebook, Instagram, google+) offer a platform to people to voice their opinions. For example, after watching any movie people quickly post their reviews online and then start a series of comments to discuss the acting skills depicted in the movie. This information forms a basis for people to evaluate, rate about the performance of not only any movie but about other products and to know about whether it will be a success or not. This type of deep information on these sites can be used for marketing and social studies. Therefore, sentiment analysis has wide applications and includes emotion mining, polarity, classification and influence analysis.

A. Existing Solution Approaches

Currently the movies are rated based on the rating score provided by the user and based on those ratings the movies are classified and recommended to the users [1]. This is a pretty direct approach and in a way a decent approach to ask users to rate movies using rating score. But nowadays the users also give comment reviews to the movies and mostly that comments are made to be read by the user to know how the movie is. But is not used to add rating to the movies and in personalized recommendations.

B. Our Approach

Text reviews are used to rate movies by doing sentiment analysis of the reviews and assigning polarity scores to them. In addition to numeric rating, the sentiment analysis of user reviews are also a very important parameter to rate and classify the movies. Polarity scores are dynamically assigned to the positive words, based on that assigning scores to individual positive reviews, and then summing up all the positive review scores to provide a score to the movie.

II. DETAILED APPROACH

A. Text Mining

For text mining the NLTK (Natural Language Toolkit) module is used which comprises the Splitter class [3]. Firstly, the review from the user is stored in the database. Then our script runs periodically as a cron job and sees if a new review comes. Each sentence is then split and is tokenized using the Splitter class. Each tokenized sentence is stored in a list. After tokenizing all the tokens are fed to our SVM model.

For example let's consider this review given by a user, "This is a very bad movie". This sentence is tokenized and stored in a list as shown below.

['This','is','a','very','bad','movie']

Once the tokenization of the sentence is done the next step is to check for emotions [9].

Now it is fed to the SVM model which categorizes positive and negative words and outputs the overall sentiment of the text as positive or negative. Here in this case output will be negative.

A word based polarity scoring system is implemented which assigns polarity scores dynamically as the data grows [4]. Therefore the weight of the emotions is recalculated periodically with time and the more data gathered the emotions weight becomes more refined.

B. Assumptions

There is no sarcasm in the review text. Human intelligence can detect sarcasm according to situation and context but this is still one of shortcomings of sentiment analysis to correctly classify sarcasm. And this can be subject for further research for this paper.

C. Application of SVM Classifier

The mined text data is then fed to the SVM model which classifies the text word into positive and negatives. A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for binary classification problems [5].

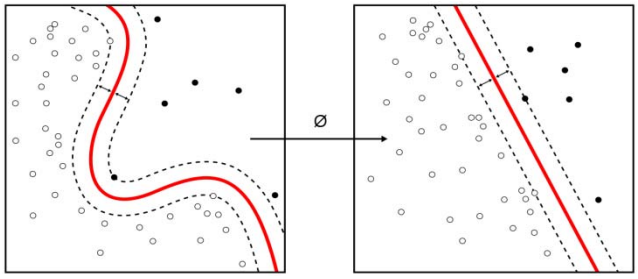

Fig. 1: Data separation in SVM.

SVM performs classification by calculating the hyper-plane that separates the classes plotted in n-dimensional space with the help of mathematical functions called “Kernels”. Various types of Kernels are linear, sigmoid, RBF, non-linear, polynomial, etc., “RBF” is for non-linear problems and it is also a general purpose kernel used when there is no prior knowledge about the data. Kernel —” linear” is for linear separable problems. Since our problem is linear(just positive and negative) here, will go for “linear SVM[11].

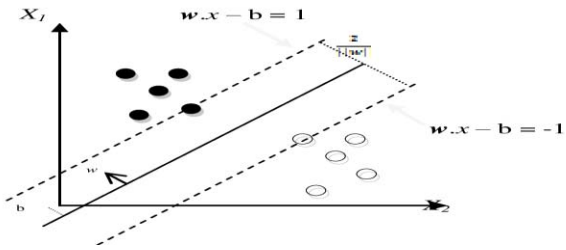

Fig. 2: Optimal separating hyperplane between two classes.

D. Application of KNN Classifier

k nearest neighbor (kNN) method is a one of best classification methods in data mining and statistics because of its simple implementation and satisfactory classification performance [6].

In the KNN algorithm an object is classified based on the majority vote of k nearest neighbours. It is used for both classification and regression. *k*-NN is an example of a lazy learning model because the functions are only approximated locally and all computation is deferred until function evaluation [8].

For our purpose, applied the KNN algorithm to find the top 3 closest words for a particular word. Vectorial representation of the words have been used to compute the distance between two words which would determine how close are the words in terms of meaning and sentimental significance. This step is a part of our movie ranking algorithm(step no 10) explained in the later section of the paper.

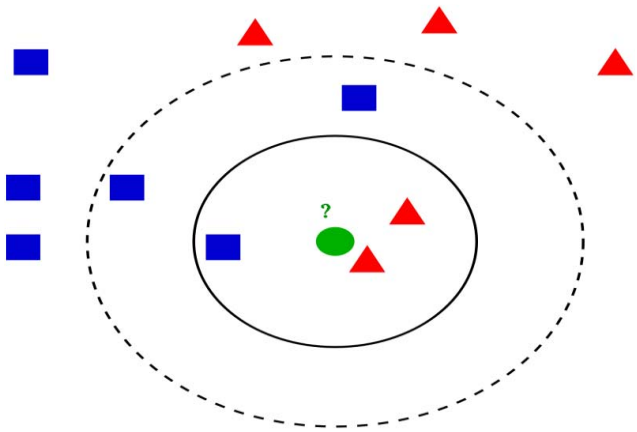

Fig. 3: Example of *k*-NN classification.

E. Android Application Model

As mentioned before, have an android application as the user software. The app will be responsible for providing users all the data about various kinds of movies with some of the categories being Highest Rated movies, Popular movies, Upcoming movies etc [12]. The app makes use of the TMDB movie API that provides endpoints to retrieve lots of information about different categories of movies, such as synopsis, cast details, all the reviews given to the movie being a few. The app displays all the information to the user in a very intuitive way. The app also follows Google’s Model-

View-ViewModel system architecture which is the industry standard for architectural patterns in Android nowadays [10].

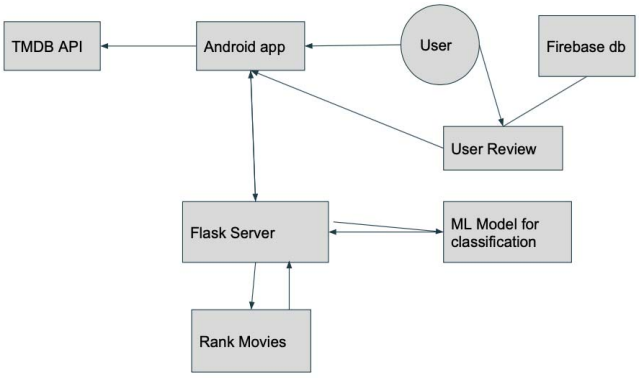

Fig. 4: Application Design.

Main Features:

1. User SignIn/SignUp - As our main aim is to provide personalized movie recommendations based on reviews, it becomes utmost necessary that every person using the app has a registered account to keep track of his/her choices.
2. Display information about various categories of movies - In order for the user to pick and choose movies to go with, provide feedback for movies it is necessary to first present the user with a world of movie data to opt from.This data is obtained from the TMDB API, to display to the user.
3. Maintain a favourite collection of movies - This functionality allows users to maintain a list of their most favourite movies which along with the review data proves to be really useful in determining the preference of a particular user in order to provide recommendations.
4. Add reviews for movies - This functionality allows users to add text reviews for any movie. It is the main feature of the app as sentiment analysis is done on the text review and determines whether or not the user liked the movie and the recommendations are generated real-time based on this functionality.

The app in real-time communicates with the TMDB API to fetch and display data to the user and thus has to make network requests at frequent intervals. As network requests can sometimes prove to be difficult and also have incorporated the functionality for offline syncing of movie info in order to avoid any difficulty due to network coverage at any instant. When the net-work coverage is supposed down, the app displays an offline synced list of movie data and when the network connectivity is good it directly communicates with the TMDB server for data.
For storing the user review data, the app uses Firebase's Realtime database support, where the data is stored in a No-SQL form i.e JSON structure. For storing the user's private info received at the time of sign up, the app also uses Firebase Realtime database.

When a user adds a review, the movie-review pair and user-review pair is updated in the Firebase Realtime database and the event is listened by the app by the application of event listeners attached to the Realtime database and actions are performed accordingly in the application real-time

F. Movie Ranking Approach

One of the most critical parts of our system is ranking movies from a particular category based on sentiments from all the text reviews received. To assign dynamic polarity scores to positive words extracted from the review and have used tf-idf weighting methodology where each review corresponds to an individual document. The corpse would consist of all the reviews from all the movies. All the positive words would be assigned a tf-idf score which would be recalculated every time new reviews are added. For the sake of computation and scalability challenges would be using reviews of only a decent sized set of movies for now rather than all the movies which would number close to millions. Our algorithm for this ranking system is provided below:

1. Store a list of most common and standard positive words. Some of the common positive words being ['good', 'brilliant', 'love', 'joyful', 'fun', 'enjoy', 'amazing'] etc.
2. Extract each word's vector representation using Word2Vec library.
3. Assign an initial static score/weight to each word based on how frequently that word is used for expressing positive sentiment(assigning static scores based on research). Also assign a polarity score(initially 0).
4. Parse a review and extract out the positive words from it using SVM.
5. Check for each word if it's present in the list of stored words.
6. Store the corresponding vector representation of each word (using Word2Vec).
7. If the word is already in the list, update the tf-idf weight of the word, calculate a new polarity score by multiplying the updated tf-idf score with the static score and take the average of the previous polarity score and the new score.
8. If the word is not present in the original list, fetch synonyms of words from the WordsAPI service and check if the synonymous words are present in the list. If present, follow step no 7 to calculate and update/store the score.
9. If none of the synonymous words are present in the list. go to step 10.
10. Calculate the three closest words to it present in the list by using KNN algorithm (take the vectorial representation of the words to calculate Euclidean distance).

Vectorial representation are extracted for every word using the Word2Vec library. So the word is taken at hand, take it's vectorial representation, calculate it's Euclidean distance from each word in the original list and select the 3 nearest neighbours based on the distance calculated. Thereafter, took the average of the static

scores(score which was initially assigned) of the 3 nearest neighbours, calculate the tf-idf score of the word and multiply it with the static score calculated(from the 3 nearest neighbours), and store this as the new polarity score of the word. Also add the word to the original list with its static score(KNN one), polarity score and its updated tf-idf score [7].

11. Similarly calculate a score for all the extracted positive words from the review, their sum would give the weight/score of that particular review.
12. Keep updating the polarity score and tf-idf score in successive iterations according to the steps mentioned above.
13. Similarly calculate scores of all the positive reviews for the movie, the summation of all would give the score of the movie.
14. Now enlist the top 'n' movies of that genre based on our score calculated for the movies in that genre in the recommendation section.
15. Extract out top 'n' movies for the other liked genres too based on our scoring system.
16. Display top 'n' movies from the liked genres to the users as recommendation.

G. Flask Server

This implementation corresponds to the Machine Learning part of the system. The main aim, as stated earlier, of the system is to make use of the text reviews provided by users to effectively recommend movies to the users based on personalized choices. So the app stores user reviews and in order to get the most about user's preference from the reviews provided the app has to communicate with the Machine learning SVM model to do sentiment analysis on the user reviews.

As the reviews are generated by users in real-time and providing recommendation would only fair when provided real-time without the user required to perform any specific action for getting recommendations, it necessary that the machine learning model is available real time to take input reviews and classify it either positive or negative using sentiment analysis and return the result to the application. Therefore to provide real-time classification of text reviews the ML model is deployed using a Python Flask server as Heroku application.

Some of the main features of the Flask server are stated below:

1. Run the ML model - The Flask server contains the ML model for sentiment analysis of text reviews, and therefore is responsible for categorizing reviews into positive or negative and determining user's preference for some particular movie.
2. Provide API interface for the mobile app to connect to the flask server- The Flask server exposes the ML model to the Android application through API architecture i.e process to communicate with the app using HTTP GET, SET, POST verbs etc. And the android application can only access the ML model functionalities by connecting to the web server via the exposed API interface.
3. Return category along with the review - The server is responsible for returning proper data for the request made by the android app. Depending on whether or not the request made was proper with the correct form of data,the server returns the result to the requesting application. If the data sent is improper or the model fails to classify then a proper error message is sent back to the application stating the error, and if the request was correctly made with the correct parameter, category classified is returned.

Overall, the application follows a client-server architecture which is common one for distributed applications these days. The android application acts as the client for the part of requesting movie data and the TMDB API acts as the server application accepting requests from the client side for data corresponding to movies, validates the requests and returns the data if the request was valid one with proper parameters or returns a clear error message stating what was wrong in the request.

For the part of interacting with the ML model, the Android app again acts as the client application and the Flask app acts as the server application which serves some data as results of the query, and after getting the user text reviews the application sends the text reviews to the Flask server, requesting the server to deliver the sentiment of the text review sent. Once the category for the text review is obtained by the app, if the class is negative it is discarded but if the review is positive, corresponding information from the review like the movie ID, genre of the movie, cast of the movie etc is extracted and movies belonging to similar features are searched and obtained from the TMDB database in order to recommend to the users.

H. TMDB API for dataset

The Movie Database(TMDB) is a very popular database for gathering various kinds of information about movies, TV Series, actors etc. It provides a developer API through which developers can retrieve information about various kinds/genres of movies. For our purpose mostly concerned with movie data. TMDB API provides various endpoints to fetch/collect data. For our purpose, data is required such as movies from various categories including "most-popular", "all-time-highest-rated", "upcoming", "top-rated" etc. For every category TMDB provides a separate endpoint. The data that fetched from the api consists of a huge list of movies with all kinds of information. Some of the attributes are:

1. Movie-name
2. Movie-genre
3. Movie-rating(/10)
4. Reviews(all the text reviews till date)
5. Actors
6. Earnings

Since a full-fledged application is built to implement in our research work, all of the above mentioned attributes along with a few more are required. But most importantly required review, name, genre data. The review data is what

fed to the ML model for sentiment analysis and to the movie ranking algorithm to rank movies in a particular genre/category.

I. Testing

In order for the system to work properly, the model has to be properly trained with sufficient amounts of data and then testing can be first done with another dataset and then after deployment, real-time review sentiment can be generated for the user. The main necessity for a machine learning model to predict correctly is data. If there is no proper data available, the model does not learn correctly and most of its predictions will be wrong. The TMDB movie review dataset is used for this purpose. It correctly classifies the entire review data to two portions:1. The positive review denoting user liking for that genre.2. The negative review denoting user dis-liking for that genre. The trained Machine learning model is Linear SVM model, the built-in one using the library scikit-learn. The accuracy achieved using the SVM model is close to 85.

J. Result

The result for our work can be classified into 2 parts, first the result produced by the ML model trained for sentiment analysis of reviews, and second performance of our movie ranking algorithm to rank and recommend movies. The accuracy achieved in classifying a review as positive or negative, from our SVM model is close to 85 and have a dynamic dataset that grows over time(as people won't stop writing reviews right), and with more and more data aimed to further improve our model for better results. For our movie ranking algorithm, it was a bit tricky to determine the success rate since what it does is rank movies based on the polarity scores determined for their reviews and finally recommend the top ranked movies from the user's liked genres to them. So it was important to test it out with real users, who would use our application to browse movies from various categories, add reviews to movies and based on their positive reviews, extracted the genres they like and recommend to them the top ranked movies from those genres. So, did test it out with real users and about 65-70% of them had positive feedback regarding our recommendation system.

Component	Accuracy	Basis
SVM sentiment Analysis model	85% 30,000 text reviews(train+test)	Categories assigned to the classified reviews(pos, neg)
Bernoulli's Naive Bayes sentiment analysis model	76% 30,000 text reviews(train+test)	Categories assigned to the classified reviews(pos, neg)
Movie Ranking Algorithm	66.7% 5,000 positive reviews to construct a dynamic bag of positive words and assign dynamic polarity scoring.	As of now, IMDB ratings and overall success of the movie in terms of earnings and aggregation of popularity. End-user opinion has also been taken into account here.
Real-time corpus construction, parsing and adding to dynamically growing bag of words	66% Reviews of movies displayed to the user, this is real-time. Also in fetching new movies from the TMDB API on a scheduled job.	Latency and accuracy
Android App	70%	End users reviews for the entire system, considering factors such as display of info, option to add reviews, time taken for real-time recommendations etc.

III. Conclusion And Future Scope

After testing our application and having satisfactory results aim to extend our rating system into a movie recommender system that users can also get recommendations based on his watch history in accordance with ratings of movies from sentiment analysis of the reviews. This text review analysis based system can be implemented with other recommendation use-cases as well such as product review etc. Scalability and infrastructure can be addressed to store and parse more and more reviews and make the system more efficient.

Acknowledgment

We thank Amma for the inspiration for our work. We thank faculties at the Department of computer Science for help and inspiration.

References

[1] Prof.Manoj Kumar, DK Yadav(MNNIT Lucknow), 2015, A Movie Rec- ommender System.

[2] Leung, C.W., 2009. Sentiment analysis of product reviews. In *Encyclopedia of Data Warehousing and Mining, Second Edition* (pp. 1794-1799). IGI Global.

[3] K. S. Krishnaveni, R. R. Pai and V. Iyer, "Faculty rating system based on student feedback using sentimental analysis," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, 2017, pp. 1648-1653.

[4] Thangaraj, M. and Sivakami, M., 2018. TEXT CLASSIFICATION TECHNIQUES: A LITERATURE REVIEW. *Interdisciplinary Journal of Information, Knowledge & Management*, *13*.

[5] Raj, J. S., & Ananthi, J. V. (2019). RECURRENT NEURAL NETWORKS AND NONLINEAR PREDICTION IN SUPPORT VECTOR MACHINES. Journal of Soft Computing Paradigm (JSCP), 1(01), 33-40.

[6] S. Zhang, X. Li, M. Zong, X. Zhu and R. Wang, "Efficient kNN Classification With Different Numbers of Nearest Neighbors," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 5, pp. 1774-1785, May 2018.

[7] A. Moldagulova and R. B. Sulaiman, "Using KNN algorithm for classification of textual documents," 2017 8th International Conference on Information Technology (ICIT), Amman, 2017, pp. 665-671.

[8] B. Sun, J. Du and T. Gao, "Study on the Improvement of K-Nearest-Neighbor Algorithm," 2009 International Conference on Artificial Intelligence and Computational Intelligence, Shanghai, 2009, pp. 390-393.

[9] M. Kanakaraj and R. M. R. Guddeti, "Performance analysis of Ensemble methods on Twitter sentiment analysis using NLP techniques," Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015), Anaheim, CA, 2015, pp. 169-170.

[10] Holla, Suhas and Mahima M Katti. "ANDROID BASED MOBILE APPLICATION DEVELOPMENT and its SECURITY." (2012).

[11] Yujun Yang, Jianping Li and Yimei Yang, "The research of the fast SVM classifier method," *2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, Chengdu, 2015, pp. 121-124.

[12] J. Liu and J. Yu, "Research on Development of Android Applications," 2011 4th International Conference on Intelligent Networks and Intelligent Systems, Kunming, 2011, pp. 69-72.