2020 International Conference for Emerging Technology (INCET)
Belgaum, India. Jun 5-7, 2020

FLEX: A Content Based Movie Recommender

Rujhan Singla[1], Saamarth Gupta[2], Anirudh Gupta[3], Dinesh Kumar Vishwakarma[4]
[1,2,3] Department of Information Technology, Delhi Technological University, New Delhi, India
[4]Associate Professor, Department of Information Technology, Delhi Technological University, New Delhi, India
[1]rujhansingla@gmail.com, [2]saamarthgupta@gmail.com, [3]anirudhgupta@yahoo.com, [4]dinesh@dtu.ac.in

***Abstract*—Recommender systems are an efficient and powerful method for enabling users to filter through large information and product spaces. In this paper, we present a movie recommendation framework (FLEX) following a content based filtering approach. FLEX extends existing approaches like Doc2Vec and tf-idf by using a hybrid of the two methods. We use publicly available features such as movie plots, ratings, countries of production and release year to find similarity between movies and generate a recommendation list. By providing recommendations which are consistent with customer interests, the aim is to make a platform personalised and ensure customer engagement.**

Keywords—content-based, movies, recommender system, IMDb, doc2vec, tf-idf, Natural Language Processing

I. INTRODUCTION

A. Motivation

Digital content present on the web these days is proliferating. It is dynamic and diverse in nature. This complicates the process of deriving information which helps in providing users with individualised support in winnowing through the massive amounts of accessible data. Recommendation systems are algorithms used to suggest relevant items (like movies, songs, shows etc.) to users. They are helpful in tackling any kind of overload in product spaces. These are often integrated into e-commerce and online systems. A crucial reason for doing this is to increase sales or user base volume. A customer is more likely to buy/view (access) a product if it is suggested to them. One of the most prominent examples of the same is the online streaming website, Netflix.com. The amount of movies and shows available here is increasing exponentially. With so much content, Netflix has a recommendation engine to enhance user experience and keep the audience engaged. The engine recommends personalised content based on certain predefined parameters. These non-exhaustively include a user's watch history, search history, and the items (movies, TV shows) that are currently being viewed. With rapidly increasing content, recommendation systems turn out as one of the prominent methods to deliver 'actual value 'to a customer - by being a scalable method to personalise content for them.

B. Related Work

Broadly, recommender systems can be divided into two categories: Collaborative Filtering and Content-Based Filtering. Collaborative Filtering methods use user-related information, preferences and user-item interactions to identify similarity between users. It recommends movies that similar users like. These can be further divided into two categories - Model-based and Memory-based algorithms. Memory-based methods don't have a training phase: they calculate similarity of the 'test 'user to training users and perform a weighted average of the most similar ones to give their recommendations. While earlier methods simply used measures like Pearson correlation coefficient, Cosine similarity [1] to identify similar users, modern-day approaches also involve the analysis of co-rated items (as demonstrated in [2]) to remove irrelevant and dissimilar users, thereby reducing data sparsity. Model-based methods try to predict user-ratings of a movie using estimated models. Popular approaches in this category are the recommender systems used by big companies like Amazon, Netflix etc. While Amazon has been using collaborative filtering to recommend products to its customers for atleast a decade, Netflix still values improvements to their recommendation services via the much distinguished 'Netflix Prize '[3]. The latest ones to win used a collaborative filtering algorithm using a Restricted Boltzmann Machine model, as described in [4]. However, collaborative filtering methods in general incur heavy computational overhead and perform poorly in the case of sparse data. Also, they assume that users with similar tastes rate similarly, which might not be true. A user may give higher ratings to items in general. Content-Based methods (or cognitive filtering) on the other hand, use information and metadata about the content to find similarities among them, without incorporating user behaviour in any way. Items similar to those 'accessed 'or 'searched 'by the user are recommended here. Some approaches analyse the audio and visual features (video frames, audio clips, movie posters etc.), as in [5] using image and signal processing techniques while some analyse textual features (metadata like plots, subtitles, genre, cast etc.) via Natural Language Processing methods like tf-idf, as in [6], and word2vec, as in [7]. The major difference between collaborative filtering and content-based recommender systems is that the former only uses the user-item ratings to make recommendations, while content-based recommender systems rely on the features of users and items for predictions. In this paper, we experiment with the latter approach.

II. METHODOLOGY

A. Dataset and Features

The dataset used for our task is the IMDb Dataset (Information courtesy of IMDb - http://www.imdb.com). This dataset contains details about all movies/ TV shows/ documentaries/ short films released from the year 1910 till date having approximately 6 million entries. However, we filter these and include only those movies with total votes greater than or equal to 2000 and rating greater than or equal to 5.0. This reduces our training data to a total of 13,469 movies. The data contains a variety of information about the movies such as title, ratings (includes votes), runtime, crew (directors, writers etc.), genre, principal crew (actors) and other production information . However, we fetch only a subset of this information and dump it onto a JSON file to complete data pre-processing. The procedure is illustrated in

978-1-7281-6221-8/20/$31.00 ©2020 IEEE

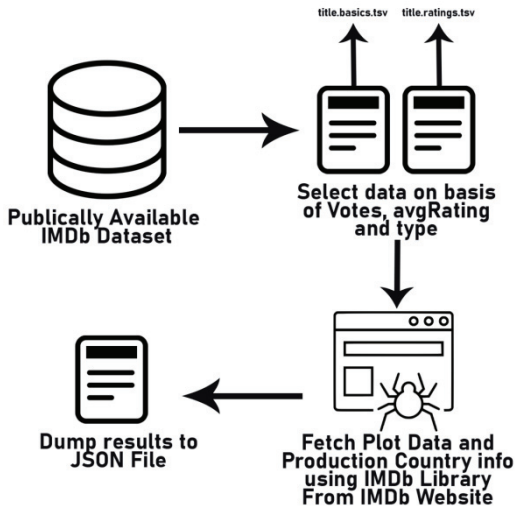

Fig. 1. The Data Cleaning Process: This includes data fetching using IMDb Library, cleaning on the basis on avgRating and Votes and combining this data with Movie Plots and Production Country Information to form a resultant JSON file.

Figure 1. The structure of our training JSON file is illustrated and explained in Table 1.

TABLE I. JSON FILE OBTAINED AFTER DATA CLEANING PROCESS

Field	Type	Usage
id	String	Alphanumeric unique identifier of the title
plot	String	A string explaining the plot of the given title
genre	String Array	Includes up to three genres associated with the titles
title	String	Original title of the movie
votes	Integer	Number of votes the titles has received
avgRating	Floating Point Integer	Weighted average of all the individual user ratings
year	YYYY	Represents the release year of a title
productionInfo	String Array	Includes the location of production/ country of origin of the given title

B. Algorithm

1) Distributed Bag of Words version of Paragraph Vector (PV-DBOW): We use movie plots/descriptions as our foundation to analyse, compare and evaluate inter-movie similarity scores. Our technique to generate 'plot embeddings' is inspired by 'Paragraph Vectors' doc2vec model approach introduced in [8]. We propose a 128 - dimensional paragraph vector distributed bag of words model (using doc2vec). The model is used to embed each movie plot (and the words found in them) into a multidimensional (128) vector space while also preserving semantic relationships among words. The plot similarity scores among these embeddings are evaluated using cosine similarity. This is used to generate a preliminary recommendation list.

However, this poses two problems. First, the approach fails to consider the quality of items. A highly-rated or universally liked movie and a badly received one are equally likely to be recommended if they share descriptions/plots with similar characteristics like identical phrases, common character names etc. Remake of cult movies which are not as well celebrated and received as their original counterpart are an example of this situation. Secondly, movie plots can often lack keywords. It is common for two movies to have similar storylines but very different genres. A Parody or spoof movie like 'Meet the Spartans '(spoof of the movie '300') is an example. '300 'is an action movie whereas its spoof is a comedy with the same storyline. To tackle the first problem, metadata such as rating of the movie, location/origin of release and year of release was considered. We use these to filter our list.

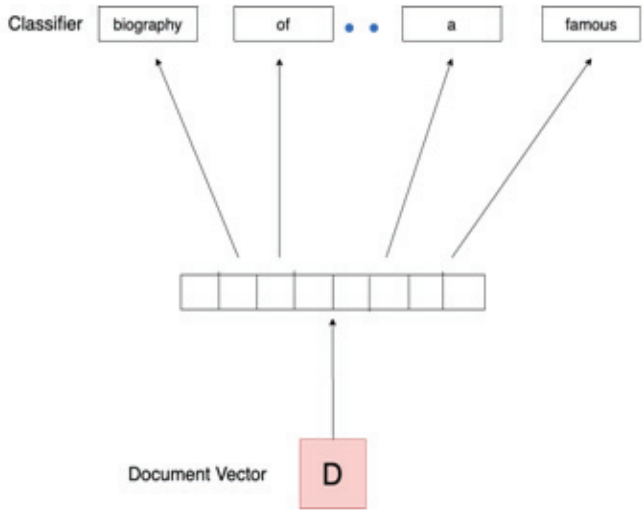

Fig. 2. Paragraph vector - Distributed Bag of Words

Term Frequency - Inverse Document Frequency: For the second problem, genres of a movie were taken separately as keywords. Each movie in the training dataset has upto 3 genres. However, some genres (keywords) are frequent, whereas others are rare and almost enough to identify relevant recommendations. For example - 'Drama 'or 'Love ' is common and present in most of the movies in some way or the other. 'Biography' or 'Musical 'on the other hand is rare and a crucial factor in finding similar movies. Therefore, we use the term frequency-inverse document frequency (tf-idf) weighting technique introduced in [9] and [10]. Based on it's usage in [11] for finding word relevance in documents, we use this method to assign appropriate weights to genres.

In [9], term frequency is defined as the frequency of a term in a given document. In this scenario, document stands for movie and term denotes genre. This value is appropriately normalized using log normalization. Inverse Document Frequency of a document in a corpus is the number of documents (movies) in which that term (genre) appears. This term is used to give lower weight to the commonly occurring words, as explained in [10]. These terms are divided and manipulated to embed each document (movie) into a unit vector, as in [12]. For each pair of movies, cosine similarity between their corresponding unit vectors is used to calculate the genre similarity between them.

3) Refining the preliminary recommendations: In this step, we try to incorporate other metadata such as release year and country of production. These are in addition to the existing plot similarity scores calculated from the doc2vec model as well as the genre similarity values calculated from the tf-idf vectorization weighting scheme. First of all, for a searched movie 'M', our initial recommendation list consists of top-k (k=40) similar movies (m_i) given by the doc2vec model. The year similarity between a searched movie 'M' and the corresponding recommended movie 'm' is calculated with the help of the following function:

$$yearSim(m,M) = \begin{cases} 1 - \frac{|m_y - M_y|}{50}, if\,|m_y - M_y| < 40 \\ 0.2, if\,|m_y - M_y| \geq 40 \end{cases} \quad (1)$$

Here, subscript 'y' denotes the release year. Similarly, for obtaining the Production Country Similarity, we use the production country metadata to compare the production country sets of two movies. If they match completely, the similarity is set to 1. In case of a partial match, the similarity is set to 0.5. In case there is no match, or if the countrySet doesn't contain any of the countries in the default set of ['India', 'United States', 'United Kingdom', 'Australia'], we set the similarity to 0.

In the end, a weighted sum of all the similarity scores calculated above is used to calculate the final similarity scores. The final function to calculate similarity between two movies is shown below (Equation 2). We recommend the top k_1 (k_1=5) to our user. The overall flow of the algorithm is illustrated in Fig. 3

$$Similarity(m,M) = 0.55 * plotSim(m,M) + 0.45 * genreSim(m,M) + 0.15 * yearSim(m,M) + 0.4 * productionCountrySim(m,M) \quad (1)$$

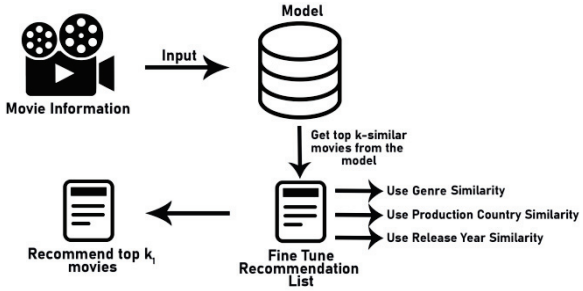

Fig. 3. Recommendation Engine: The recommendation engine is primarily based on Doc2Vec model and is further fine-tuned using Genre similarity, Production Country similarity and Release Year similarity to provide the top recommendations.

III. EXPERIMENT & RESULTS

Contrary to a collaborative filtering system, where the predicted ratings can always be compared with actual ones, evaluating a content-based recommender system on the basis of statistical measures can be a daunting problem. This is mainly due to the absence of any 'ground truth 'regarding its predictions. In order to evaluate the performance of our proposed algorithm, an experiment is conducted on the dataset of IMDb, wherein a subset of the total data, i.e. data of 13469 movie titles, selected on the basis of average number of votes ≥ 2000 and average rating ≥ 5.0, is used.. We run the algorithm over this dataset to build a content-based recommender system. The dataset along with the predictions made by our system are then used to obtain the subsequently mentioned result metrics through a variety of methods:

A. Coverage

The coverage of a recommender system deals with the representation of the percentage of the training data that the recommender system was able to recommend. For a dataset containing 'N 'items, out of which a total of 'n 'items are recommended at some point, the coverage is given as:

$$coverage_{system} = \frac{n}{N} * 100 \quad (3)$$

For a value of 'N 'as 13469, which is the size of the training data, a coverage of **77.058%** was achieved using the proposed methodology. The movies not being recommended could have conflicting plots or score low in similarity score to an extent where the top k=5 recommendations over-shadow them.

B. Precision@k

We use Precision@k as one of the result metrics for evaluating the performance of our algorithm. We define Precision@k for top-k recommended movies as:

$$Precision@k = \frac{Number\ of\ Movies\ that\ were\ Relevant}{Total\ Number\ of\ Movies\ that\ were\ Recommended} \quad (4)$$

Here, we treat the recommendation task as a classic binary classification problem. The recommendations made by our system are predictions and a relevant movie counts as a 'positive'. A 'true positive' here denotes movies which were relevant and were recommended by our system. We propose the following two approaches at obtaining the number of relevant movies .

1) Using Similarity Threshold: We use min-max normalization to normalize the similarity scores of our recommendation engine. We then use observation to arrive at a similarity threshold of 0.85 to check for relevant recommendations.

2) Using Average Rating of a Title: After surveying 200 people about what they would consider relevant, an IMDb movie with a rating score ≥ 7.0 or a movie with an inferior score but more popularity, we got an 87% positive response to the criteria of average rating ≥ 7.0. So, we assume that movies matching this criteria are relevant.

Each system is run on 10, 50, 100, 200, 400 and 500 movies (chosen randomly), giving top-k recommendations, for 3 dummy users.

The Precision@k metric for k=5 using both the techniques is depicted in the graphs below (Fig. 4 and 5). As we increase the sample size, the value saturates to 0.66 (for similarity threshold) and 0.47 (for average rating) respectively.

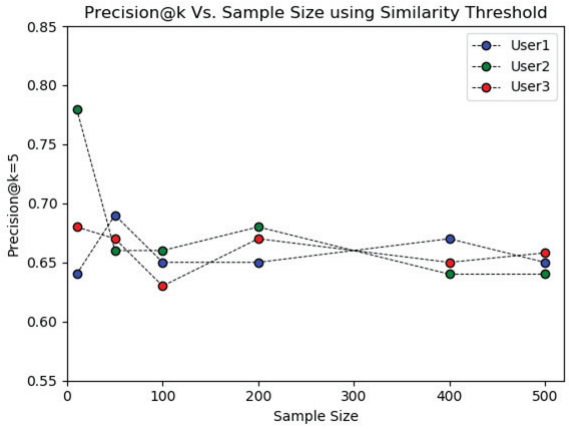

Fig 4. *Precision@5 vs. Sample size* for 3 dummy users using similarity threshold of 0.85 as criteria for relevance (III.B.1)

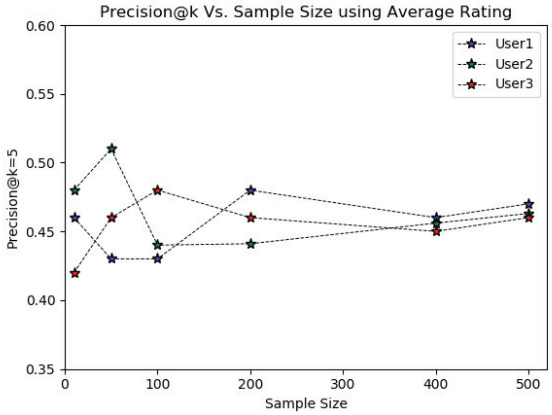

Fig. 5. *Precision@5 vs. Sample size* for 3 dummy users using movie rating >= 7.0 as criteria for relevance (III.B.2).

V. CONCLUSION

In this paper, we proposed FLEX - a content-based movie recommendation system. The proposed algorithm uses textual metadata of the movies like plot, cast, genre, release year and other production information to analyse them and recommend the most similar ones. Our system only needs a movie which the user is interested in to come up with suitable recommendations. For evaluation, we ran our algorithm on a subset of all the movies present on the IMDb server. Through thorough experimentation, we were able to deduce that the method proposed in this paper would help in serving as an effective solution to one of the most thriving industries in today's time.

Future work includes incorporating audio and visual features along with the metadata we used to perform a multi-modal analysis. These include video frames and audio clips from the movie and involve image and signal processing. Moreover, keeping a track of movies searched by users in nearby location to recommend trending movies is also a favourable option. We can try to combine the watch history of the user with the watch history of geographically contextual users (those living nearby) to give more 'location relevant' recommendations. Furthermore, using user ratings of movies on websites like Rotten tomatoes, Metacritic, IMDb etc. opens up the possibility of combining collaborative filtering techniques with our method into a hybrid model to get the best out of both approaches.

REFERENCES

[1] J. S. Breese, D. E. Heckerman and C. M. Kadie "Empirical analysis of predictive algorithms for collaborative filtering," *UAI'98: Proc. of the 14th Conf. on Uncertainty in artificial intelligence*, Madison, 1998, pp. 43–52.

[2] R. Lumauag, A. Sison, and R. P. Medina, "An enhanced memory-based collaborative filtering algorithm based on user similarity for recommender systems," *Int. Journal of Recent Technology and Engineering,* (vol. 7), 2019, pp. 778-782.

[3] J. Bennett, S. Lanning and Netflix, "The Netflix Prize," 2019.

[4] R. M. Bell, Y. Koren, Yahoo Research and I. C. Volinsky, "The BellKor 2008 solution to the Netflix Prize," *AT&T Research,* 2008.

[5] T. Lehinevych, N. Kokkinis-Ntrenis, G. Siantikos, A. S. Dogruöz, T. Giannakopoulos and S. Konstantopoulos, "Discovering Similarities for Content-Based Recommendation and Browsing in Multimedia Collections," *2014 10th Int. Conf. on Signal-Image Technology and Internet-Based Systems*, Marrakech, 2014, pp. 237-243.

[6] Z. Cataltepe, M. Uluyağmur and E. Tayfur, "Feature selection for movie recommendation," *Turkish Journal of Electrical Engineering and Computer Sciences,* (vol. 24), 2016, pp. 833-848.

[7] H. Chen, Y. Wu, M. Hor and C. Tang, "Fully content-based movie recommender system with feature extraction using neural network," *2017 Int. Conf. on Machine Learning and Cybernetics (ICMLC)*, Ningbo, 2017, pp. 504-509.

[8] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," *31st Int. Conf. on Machine Learning (ICML)*, Beijing, 2014, (vol. 4).

[9] K. S. Jones, "A Statistical Interpretation of term specificity and its application in retrieval", *Journal of Documentation,* (vol. 60 no. 5), 2004, pp. 493-502

[10] S. E. Robertson, "Understanding inverse document frequency: on theoretical arguments for IDF", *Journal of Documentation*, (vol. 60 no. 5), 2004, pp 503–520.

[11] S. Qaiser and R. Ali, "Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents," *Int. Journal of Computer Applications,* (vol. 181), 2018.

[12] O. Shahmirzadi, A. Lugowski and K. Younge, "Text Similarity in Vector Space Models: A Comparative Study", arXiv:1810.00664, 2018, pp. 659-666.