Personalized Movie Recommendation System

Raj Kamal Yadav, 2016076, Rajat Bansal, 2016260, Manish Mahalwal, 2016054

I. ABSTRACT

In todays world of Netflix and Chill, recommendation systems have become an essential part of our life, especially due to the increasing of choices available to us. For a media commodity like movies, suggestions are made to users by finding user profiles of individuals with similar tastes. We shall be using both Content-Based approaches with features such as cast, crew etc. as well as Collaborative Filtering for which the input to our algorithm will be observed user ratings in the past to recommend items of interest to users. For baselines, we implement Matrix Factorisation and TF-IDF approaches. We also implement BPR for personalized ranking of movies for each user. This technique proves to be far better than the two baselines MF and TF-IDF.

II. LITERATURE REVIEW

IN today's world of Netflix and chill, recommender systems have become an essential part of our lives especially with the ever increasing number of choices to choose from. Platforms like Netflix, YouTube etc. that have millions of users as well as items available for the users to watch, helping the users discover relevant content is highly important for their businesses.

1) **Content-Based method:** TF-IDF similarity
   TF-IDF stands for Term-Frequency Inverse-Document-Frequency, is a measure of the importance of words. This method works by representing each movie in-terms of a TF-IDF vector and the calculating cosine similarities between those vectors.

2) **Collaborative-Filtering Based method:** User-Based CF
   In this approach, a utility matrix is created with the help of the explicit ratings provided by some of the users. The matrix is then factorized into 3 matrices, which explain the users and latent factors, items and latent factors as well as strength of the latent factors [1]

3) **Bayesian Personal Ranking:**
   In Movie recommendation, we predict a personalized ranking on a set of movies. There are many methods for this task like matrix factorization and or adaptive k-nearest-neighbour(kNN). Although these techniques are designed for movie recommendation task, we cant use them for ranking. We will implement a specialized optimization technique called BPR-OPT for personalized ranking. We use it maximize the posterior probability which is obtained from a Bayesian analysis of the problem. The learning part of the problem is based on bootstrap sampling and stochastic gradient descent. Bayesian Personalized Ranking is used to create a user-specific ranking for a set of movies.

   - **BPR-OPT:** implement the generic criterion for BPR-OPT using the maximum posterior estimator for optimum personalized ranking of items. This optimization is to related to maximizing the area under curve of an ROC.
   - **LearnBPR:** implement a learning algorithm based on stochastic gradient descent. This technique is superior than std grad descent techniques used for optimization of BPR-OPT.

[1]Latent factor refer to a property/concept a user or an item has. E.g. for music, the latent factor can refer to the genre that the music belongs to.

III. PROBLEMS FACED

We faced some problems due to these 2 major reasons:-

- **Sparsity**: Since users rarely give ratings to all the items they watch, and since one cannot watch everything, the utility matrices are often very sparse causing lots of space wastages during computation.

- **Scalability**: Even for about 700 users and 9000 movies only, the matrices took long to compute which means for platforms like YouTube which have millions of items as well as users, it takes up a lot of computation power to do the same.

IV. DATASET

Dataset used is available at Kaggle[1] however it contains 26 million ratings from 270,000 users and 45,000 movies. We use a subset of the dataset containing 9,000 movies and 100,000 ratings over 700 users. The data is provided to us in the form of:

- **Credits.csv**: About movie's cast and crew information
- **Keywords.csv**: Keywords associated to each movie
- **Links.csv**: IMDB and TMDB IDs' for all movies
- **Rating.csv**: Contains 100,000 ratings from 700 users to 9,000 movies

V. DATA PRE-PROCESSING

- **Duplicates Removal:** Remove the duplicate entries from the data set to make even all forms of data: Credits.csv, Keywords.csv and Movie_Metadata.csv.

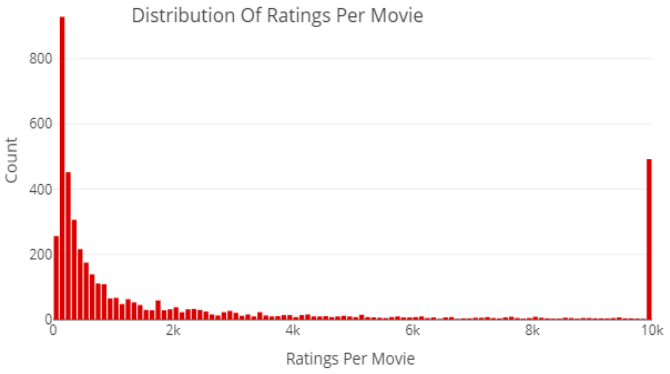

Fig. 1. Distribution of Ratings per movie

- **Reduction of Data set:** Reduce the Data set size from 45,000 initially to 9,000.

- **Fields Extraction:** Extract required fields from the chunk like Movie name, Movie ID, Movie Cast, Movie Keywords etc. for further feature vectorization, similarity calculation etc.

VI. Proposed Methods

1) **TF-IDF Based Similarity**
The method exploits the common assumption that similar movies share similar keywords and cast. A Feature Matrix M is created of dimensions m*n where m is # of movies and n is the # feature(Movie Keywords and Cast separately) length.
Pairwise Cosine Similarity between all movies is calculated and stored in the the Similarity Matrix of dimensions m*m where m is # of movies per feature.
The weighted combination of all such Similarity Matrices(per feature) forms the Weighted Similarity Matrix that stores the final similarity scores between all movies.

2) **User Based Collaborative-Filtering**
The method is based on the simple premise that similar users share similar interests. A utility matrix M is created of dimensions m*n where m is # of users and n is the # of items. $M_{i,j}$ represents the rating given by i[th] user to the j[th] item. Doing the Singular Value Decomposition of M gives:-

$$\underset{m \times n}{\hat{X}} \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix} \approx \underset{m \times r}{U} \begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix} \underset{r \times r}{S} \begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix} \underset{r \times n}{V^{\mathsf{T}}} \begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}$$

where matrix U shall represent the relationship between users and latent factors, matrix S shall represent the strengths of the latent factors and matrix V shall represent the relationship between items and latent factors.

We can predict $M_{a,b}$, which was previously unknown, using the below equation:-

$$\hat{r}_{ui} = \mu_\mu + \frac{\sum_{v \epsilon N_i^k(u)} sim(u,v)(r_{vi} - \mu_v)}{\sum_{v \epsilon N_i^k(u)} sim(u,v)}$$

where:
r_{ui} = rating given by user u to item i
r_{vi} = rating given by user v to item i
μ_u = rating given by user u to all the items
μ_v = rating given by user v to all the items
$sim(u,v)$ = similarity between user u and user v

From there on, scores for the given user were calculated for each movies and the top-N movies with the highest scores were reported as predictions by the model.

3) **Bayesian personalized ranking** This algorithm assumes that the user has interacted with the content in some manner such as – number of clicks, purchases, number of views. We have used the ratings from the users as explicit feedback to the algorithm.
It uses a pair-wise item feedback matrix to reconstruct a personalized total ranking for each user by transforming a positive only feedback into positive and negative feedback in terms of pairs of items (i,j), where the user prefers i over j (positive) and correspondingly rephrased dislikes j over i(negative).

VII. Results

1) **TF-IDF**
We used movie similarity matrices from TF-IDF for content based filtering. The weighted combination chosen is as following;
0.7 * Keyword Similarity + 0.3 * Cast Similarity
Based on the the Weighted Similarity Matrices obtained,for any given movie we can find the top-K similar movies. For representation, we have shown only five similar movies for two movies ”Toy Story” and ”Golden Eye”. We have included the similarity score of the movie to itself for comparison and reference. As we can see for ”Toy Story”, its in-series parts ”Toy Story 2” and ”Toy Story 3” turned out to be the most similar movies. Similarly ”Golden Eye” is more similar with ”The World is not Enough” and ”Tomorrow Never Dies” which is similar to Google’s recommendation(People also search for) for the mentioned movie.

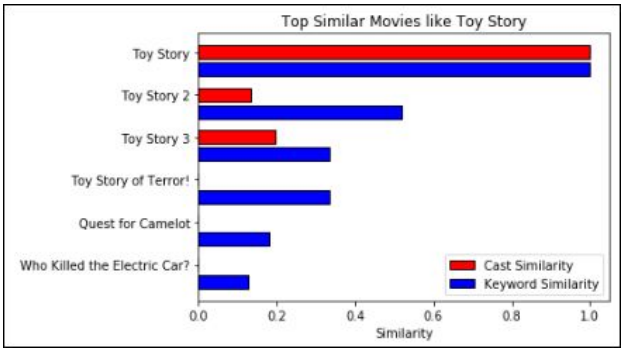

Fig. 2. Histogram of predictions by TF-IDF similarity matrix for Toy Story

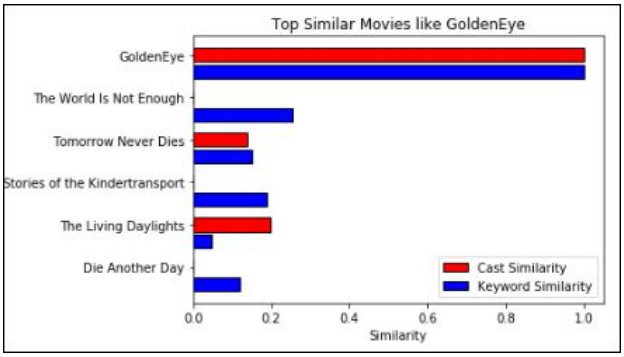

Fig. 3. Histogram of predictions by TF-IDF similarity matrix for GoldenEye

2) **Collaborative filtering**
For collaborative filtering, we use SVD, a matrix factorisation technique. We iterated through the k(no. of latent factors) i.e. k = 1 to 25 (Fig 3) and found out that the RMSE decreases as k is increased. With k = 20 we achieve an RMSE of 0.359 and with k = 25 an RMSE of 0.347 for the user-user based CF. RMSE value stagnates for values beyond k = 25.

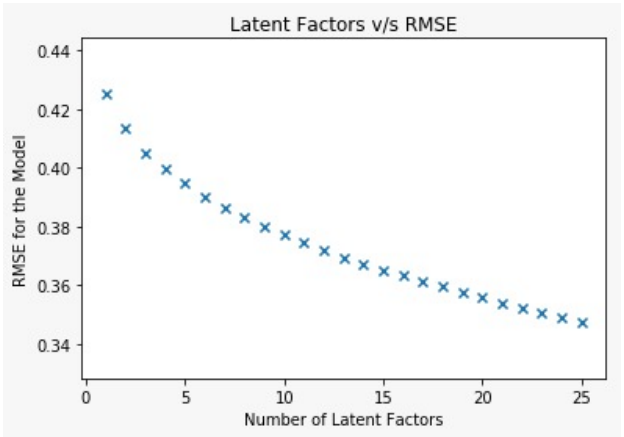

Fig. 4. RMSE for different number of latent factors.

3) **Combining results from the two baselines:** We propose the following architecture for evaluating TF-IDF approach as discussed:

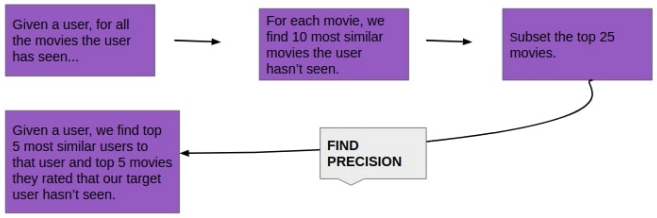

Fig. 5. Evaluation Architecture

The intuition behind the above architecture is that the movies similar to those the user have seen should also be seen by similar users and liked by them.
We obtained a **precision of 36.84%** on User3 while testing the above architecture.

4) **BPR vs TF-IDF:** In figure 6, predicting movies similar to Total Recall BPR outputs a more personalized choice to the user as compared to TF-IDF approach where TF-IDF recommends movies similar to the movies content not to the liking of the user. Thus BPR is far better than TF-IDF.

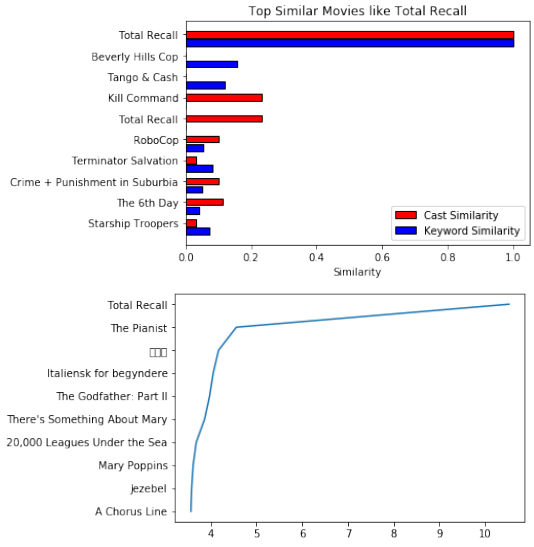

Fig. 6. TF-IDF v/s BPR for a given movie

VIII. Discussion and Summary

We implemented and explored baseline models of both content-based and collaborative filtering for personalized recommendation system.
For content-based filtering TF-IDF was implemented. For a given movie, we are able to predict top k related movies to it using their genres.
For collaborative filtering, matrix factorization technique was implemented using SVD. Our model recommends movies based on users past watched movies and by weighing the rating given by other users to these movies, we can recommend top k to a user.

We combined the two baselines MF and TF-IDF which produced improved results than the two working independently. We observed that the Bayesian personalized Ranking is far superior than the two baseline and even the combination of the two baselines and produces much more personalized results than the other methods.

IX. Individual Contribution

Manish worked on dataset preprocessing, exploration and implemented SVD technique. Rajat worked on TF-IDF and combining the two baselines. Raj worked on implementing Bayesian personalized Ranking.

References

[1] Rendle, Steffen Freudenthaler, Christoph Gantner, Zeno Schmidt-Thieme, Lars. (2012). BPR: Bayesian Personalized Ranking from Implicit Feedback. Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI 2009.

[2] Recommender System using Bayesian Personalized Ranking: https://towardsdatascience.com/recommender-system-using-bayesian-personalized-ranking-d30e98bba0b9

[3] Introduction to Recommender System: https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75

[4] Qaiser, Shahzad Ali, Ramsha. (2018). Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications. 181. 10.5120/ijca2018917395.

[5] Ma, Chih-Chao. (2019). A Guide to Singular Value Decomposition for Collaborative Filtering.

[6] The Movies Dataset. Kaggle. Rounak Banik: https://www.kaggle.com/rounakbanik/the-movies-dataset