Movie Recommendation System

Manisha Sharma
Master's in Data Science
Northeastern University
Boston, Massachusetts
sharma.manis@husky.neu.edu

Vanja Srivastava
Master's in Computer Science
Northeastern University
Boston, Massachusetts
srivastava.v@husky.neu.edu

1. INTRODUCTION

1.1 Topic

Recommendation system is in trend now-a-days. In daily life whenever we search something we try to get the most relevant results. The prevalence of internet has increased the amount of available information. However, the most recent problem is people have a hard time selecting the items they actually want to buy/watch. In a world where the number of choices can be overwhelming, recommendation systems help users find and evaluate items of interest. They connect consumers with items to "consume" (purchase, view, listen to, etc.) by associating the content of recommended items or the opinions of other individuals with the consuming user's actions or opinions.

1.2 Motivation

In the past, when internet was not such a hype, people used to go to physical stores to buy movies and select from a bundle of options and select their genre and decide the movie. It was noticed that people prefer to watch similar genres, the concept of grouping similar genres for shelves came into picture. Now, with internet where information is available in abundance, there is a need of a smart and efficient system to do the same work that was done manually.

Major tech companies such as Netflix, Amazon and Hulu have the recommendation system wherein a user gets a predicted preference set of items that the user might be interested in watching later. Out of curiosity and to learn how these systems work, we took up this task to learn more in this domain and make a recommendation system. As a result, it is in our best interest to investigate it and find potential solutions. In this project, we propose to use a MovieLens data set, which we explore in dataset section, to build recommendation systems using Content Based Filtering, Collabarative Filtering and the most popular award wining - the Latent Factor Model.

This report presents an overview of recommendation systems, the most widely used approaches in recommendation systems and techniques by building a movie recommendation system along with the analysis of the result obtained using these approaches.

1.3 Contribution

Both the authors have collaborated and equally contributed toward this report

2. RELATED WORK

2.1 Current Knowledge

The most common approach to recommendation system is based on neighborhood models. User-oriented methods estimate unknown ratings based on recorded ratings of like minded users. Later, an analogous item-oriented approach became popular. In those methods, a rating is estimated using known ratings made by the same user on similar items. Improved accuracy and better result prediction made the item oriented approach more favorable in many cases.Also they are explanatory regarding the reason behind predictions. This is because users are familiar with items previously preferred by them, but usually do not know those allegedly like minded users.

2.2 Gaps to Overcome

There are two kinds of feedback:

2.2.1 Implicit Feedback

This type of feedback reflect opinion through observing user behavior, such as purchase history, browsing history, search patterns, or even mouse movements.

For example, a user that purchased a movie released and cast a particular actor stating that the user might like the actor.

2.2.2 Explicit Feedback

The high quality explicit feedback includes input by users regarding their interest in products. Explicit feedback is not always available.This may reflect reluctance of users to rate products, or limitations of the system that is unable to collect explicit feedback. For example, giving reviews or rating on an e-commerce platform.

Our setup gives us data gathered from explicit feedback from users, so the system was solely based on explicit feedback – analyzing watching habits of unknown users.

2.2.3 Gaps in using Implicit Feedback

What about low ratings?

While our system can only point out at movies user might want to watch and is not capable of pointing out what the user dislikes, or is not aware of. Also, majority of our matrix is sparse. It could be that the user is yet to see the movie and may out of the box watch it some day and love it but implicit feedback considers such data as missing data and they are removed from our analysis.

What does numerical value indicate? The numerical value

in rating indicates 1 for dislike and 5 for like, therefore, such feedbacks will really help in getting good recommendation to the user. However, in implicit feedback such ratings are only indication of confidence.
Is implicit feedback noisy?
While we track the users behavior, we can only guess their preferences and true motives. For example, we may view purchase behavior for an individual, but this does not necessarily indicate a positive view of the product.
How to evaluate implicit feedback? While Explicit feedback can be evaluated using MAE, RMSE we would need appropriate measures to evaluate Implicit feedback.

3. BACKGROUND INFORMATION

3.1 Content - Based Filtering

At the most basic level, content-based filtering is about assigning attributes to items, so that the algorithm knows something about the content of each item in the database. Content-based filtering recommends items based on a comparison between the content of the items and a user profile. The content of each item is represented as a set of descriptors or terms, typically the words that occur in a document. The user profile is represented with the same terms and built up by analyzing the content of items which have been seen by the user.
We will look at these important concepts here which go into building this content based recommendation systems based on TF-IDF and Cosine Similarity

3.2 Collaborative - Based Filtering

Collaborative Filtering is a mean of recommendation based on users' past behavior. The key idea behind CF is that similar users share the same interest and that similar items are liked by a user.
We will look at two important concepts here which go into building this Collaborative based recommendation systems based on User-based: measure the similarity between target users and other users Item-based: measure the similarity between the items that target users rates/ interacts with and other items

3.3 Latent Factor Model

Latent factor model is based on the similarity between users and items. Initially, the methods used were neighborhood based. Instead we have used SVD to bring our data to a lower dimension since we know the [user X movies] matrix is very sparse we need to fill in the missing values. Given, R=Q.P$\hat{T}$ we know, A=U..V$\hat{T}$ Therefore A = R; Q = U; P$\hat{T}$ = .V$\hat{T}$ Select a set of factor vectors Pu for each user u and Qi for each product i that minimize the errors on the training set. Make new predictions using the above matrix.

4. PROPOSED APPROACH

4.1 Content-based filtering

This is also referred to as cognitive filtering, recommends items based on a comparison between the content of the items and a user profile. Imagine you are reading a book on data visualization and want to look for other books on the same topic. In this scenario, content based recommender system would be apt. Content-based filtering technique does not need the profile of other users since they do not influence recommendation. Item description and a profile of the users orientation play an important role in Content-based filtering. Content-based filtering algorithms try to recommend items based on similarity count. The best-matching items are recommended by comparing various candidate items with items previously rated by the user. A content based recommender works with data that the user provides. Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.
Content based filtering uses various models to find similarity between documents in order to generate meaningful recommendations. Following are the few which we used in this project:

4.1.1 TF-IDF:

The tf-idf is a statistical measure that reflects the importance of a particular word with respect to a document in a corpus. The importance of the word increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Term frequency is the number of times a term occurs in a document. Inverse document frequency is the inverse function of the number of documents in which it occurs.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Hence a term like "the" that is common across a collection will have lesser tf-idf values, as its weight is dampened by the idf component. Hence the weight computed by tf-idf represents the importance of a term inside a document. The tokenized data was used to generate a sparse matrix of tf-idf features for representation. This represented our feature vector and was used in subsequent prediction algorithms.

4.1.2 Cosine Similarity:

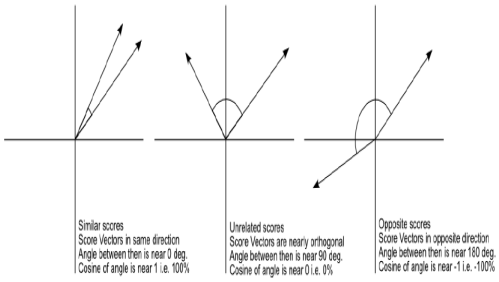
The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them. This metric is a measurement of orientation and not magnitude. The smaller the angle, the more similar the two vectors are. What we have to do to build the cosine similarity equation is to solve the equation of the dot product for the $\cos\theta$

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

Using cosine similarity, we get that two vectors achieve maximum similarity when the angle between them is 0 (they are oriented in the same direction), they have 0 similarity when the angle between them is 90 (they are orthogonal to one another), and they have -1 similarity when the angle between

them is 180 (they are oriented in diametrically opposing directions).



4.2 Collaborative - Based Filtering

Collaborative recommender systems (or collaborative filtering systems) for each user, recommender systems recommend items based on how similar users liked the item. Collaborative Filtering uses either a User-Based approach or an item-based approach.

4.2.1 User- User Collaborative Filtering

In the user-based approach, the users perform the main role. If certain majority of the customers has the same taste then they join into one group Recommendations are given to user based on evaluation of items by other users form the same group, with whom he/she shares common preferences.

Let's say Alice and Bob have similar interests in video games. Alice recently played and enjoyed the game Red Dead Redemption 2. Bob has not played this game, but because the system has learned that Alice and Bob have similar tastes as they both liked playing GTA(Grand Theft Auto), it recommends this game to Bob.

In addition to user similarity, recommender systems can also perform collaborative filtering using item similarity.

4.2.2 Item - Item Collaborative Filtering

Item-based collaborative filtering is a Model-based algorithm for making recommendations. In the algorithm, the similarities between different items in the dataset are calculated by using one of a number of similarity measures, and then these similarity values are used to predict ratings for user-item pairs not present in the dataset.

Instead of just relying on the most similar person, a prediction is normally based on the weighted average of the recommendations of several people. The weight given to a persons ratings is determined by the correlation between that person and the person for whom to make a prediction. As a measure of correlation the Pearson correlation coefficient can be used.

4.2.3 Pearson (Correlation based) Similarity:

Similarity based on Pearson correlation measures the extent to which there is a linear dependence between two variables. This similarity measure is based on how much the rating by common users for a pair of items deviate from average ratings for those items. The weight $w_{a,u}$ is a measure of similarity between the user u and the active user a. Pearson correlation coefficient between the ratings of the two users is defined below:

$$w_{a,u} = \frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2 \sum_{i \in I} (r_{u,i} - \bar{r}_u)^2}}$$

where I is the set of items rated by both users, $r_{u,i}$ is the rating given to item i by user u, and $\bar{r}u$ is the mean rating given by user u. Predictions are generally computed as the weighted average of deviations from the neighbor's mean, as in:

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u \in K} (r_{u,i} - \bar{r}_u) \times w_{a,u}}{\sum_{u \in K} w_{a,u}}$$

where $p_{a,i}$ is the prediction for the active user a for item i, $w_{a,u}$ is the similarity between users a and u, and K is the neighborhood or set of most similar users.
Similarity based on Pearson correlation measures the extent to which there is a linear dependence between two variables. Alternatively, one can treat the ratings of two users as a vector in an m-dimensional space, and compute similarity based on the cosine of the angle between them, given by:

$$w_{a,u} = \cos(\vec{r}_a, \vec{r}_u) = \frac{\vec{r}_a \cdot \vec{r}_u}{\|\vec{r}_a\|_2 \times \|\vec{r}_u\|_2} = \frac{\sum_{i=1}^{m} r_{a,i} r_{u,i}}{\sqrt{\sum_{i=1}^{m} r_{a,i}^2} \sqrt{\sum_{i=1}^{m} r_{u,i}^2}}$$

When computing cosine similarity, one cannot have negative ratings, and unrated items are treated as having a rating of zero. Empirical studies have found that Pearson correlation generally performs better.

4.3 Latent Factor Model

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. Accordingly, each item i is associated with a vector Qi and each user u is associated with a vector Pu We have to find P, Q such that:
The major challenge is computing the mapping of each item and user to factor vectors it can easily estimate the rating a user will give to any item by using below equation. Such a model is closely related to singular value decomposition (SVD), a well-established technique for identifying latent semantic factors in information retrieval. Applying SVD in the collaborative Filtering domain requires factoring the user-item rating matrix. This often raises difficulties due to the high portion of missing values caused by sparseness in the user-item ratings matrix. Conventional SVD

$$\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - q_i \cdot p_x)^2$$

- q_i = the i-th row of Q
- p_x = the x-th column of P^T

is undefined when knowledge about the matrix is incomplete. Moreover, carelessly addressing only the relatively few known entries is highly prone to overfitting. Earlier systems relied on imputation to fill in missing ratings and make the rating matrix dense. However, imputation can be very expensive as it significantly increases the amount of data. In addition, inaccurate imputation might distort the data considerably. Hence, more recent works suggested modeling directly the observed ratings only, while avoiding overfitting through a regularized model.
The more involved part is parameter estimation. Many of the recent works, applied to explicit feedback datasets, suggested modeling directly only the observed ratings, while avoiding overfitting through an adequate regularized model, such as: Here, is used for regularizing the model. Parameters are often learnt by stochastic gradient descent. One strength of matrix factorization is that it allows incorporation of additional information by a densely
lled matrix.

5. EXPERIMENTS

5.1 Experimental Setup

To analyse a recommender system we would use the Movie Lens Dataset. The description of the dataset is below. We have used three approaches to evaluate our recommender system namely, Content Based Filtering, Collaborative Filtering and Latent Factor Model. In Content Based Filtering the content of the movie is used to find its similarity with other movies. With this content we will make a TF-IDF Matrix, where each column represents a word in the movie content (all the words that appear in at least one document) and each column represents a movie, as before. After this matrix has been created, we will compute the cosine similarity scores as a numerical quantity that denotes the similarity between two movies. And then finally recommend the top 10 movies.

In User-User collaborative filtering: We will analyse similarity of a user with another user. We have userid and ratings to each movies. Therefore we make a matrix where each row represents a user, while the columns correspond to different movies and fill in the values with user ratings, we will find a Pearson correlation matrix using the above matrix to get similar users. We will then recommend top 5 movies to our user.
In item Based Collaborative Filtering - Instead of measuring the similarity between users, the item-based recommends items based on their similarity with the items that our user rated. Therefore We constructed item-item similarity matrix based on Pearson correlation. We will then recommend top 5 movies to our user.

In Latent Factor Model - We used SVD to turn the recommendation problem into an optimization problem. We compute the top recommended movies for user 85.

5.2 DataSet

For this project, dataset was taken from Kaggle which can be found here with this URL:
https://grouplens.org/datasets/movielens/ This dataset (ml-latest-small) describes 5-star rating from MovieLens, a movie recommendation service. The entire dataset amounted to over 9742 different movies and has 100836 ratings and 3683 tag applications across movies from 610 users. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided. The data are contained in the files - links.csv, movies.csv, ratings.csv and tags.csv.
File **User ids** contains anonymized ids of Movielens users who were selected at random for inclusion. User ids are consistent between ratings.csv and tags.csv
All ratings are contained in the file **ratings.csv**. Each line of this file after the header row represents one rating of one movie by one user, and has the following format: **userId,movieId,rating,timestamp**
All tags are contained in the file **tags.csv**. Each line of this file after the header row represents one tag applied to one movie by one user, and has the following format: **userId,movieId,tag,timestamp**
Movie information is contained in the file **movies.csv**. Each line of this file after the header row represents one movie, and has the following format: **movieId,title,genres**
Identifiers that can be used to link to other sources of movie data are contained in the file **links.csv**. Each line of this file after the header row represents one movie, and has the following format: **movieId,imdbId,tmdbId**

5.3 Results

Content-based Filtering gave relevant results when searched for 'Iron Man 3'. Although, it only suggests movies which are close to a certain movie.

Recommendations for 'Iron Man 3' are:

15324	Iron Man 2
22889	Behind Enemy Lines
12696	Iron Man
18008	The Avengers
24092	Teenage Mutant Ninja Turtles
26773	Avengers: Age of Ultron
26782	Captain America: Civil War
19503	Bells of Innocence
26149	The Hire: Ticker
18223	Radioactive Dreams

Fig 1. Results for Content Based

Collaborative Based Filtering: As we can see the results for 'User 85' are comparable in both kinds of recommendation however, it takes a lot of time to compute these values and

also as our data grows so does our n * n matrix. Plus, these are sparse Matrix.

```
Recommendation for User 85 using User-User Collaborative Filtering

0                       GoldenEye (1995)
1         American President, The (1995)
2                     Get Shorty (1995)
3                            Babe (1995)
4             Usual Suspects, The (1995)
5                   Birdcage, The (1996)
6                       Apollo 13 (1995)

Recommendation for User 85 using Item-Item Collaborative
Jerky Boys, The (1995)
Pulp Fiction (1994)
Leap of Faith (1992)
Star Trek: Generations (1994)
15 Minutes (2001)
Alive (1993)
```

Fig 2. Results for Collaborative Based Filtering

Hence, Latent Factor Model – Since we tackle sparsity we also save a lot of computational time. Latent Factor gives a good result. The RMSE 0.8808 and MAE: 0.6751

```
User 85 loved these movies:
Jumanji (1995)
GoldenEye (1995)
Braveheart (1995)
Jerky Boys, The (1995)
LÃ©on: The Professional (a.k.a. The Professional) (LÃ©on) (1994)
Pulp Fiction (1994)
Stargate (1994)
Shawshank Redemption, The (1994)
Star Trek: Generations (1994)
Clear and Present Danger (1994)
Speed (1994)
True Lies (1994)
Fugitive, The (1993)
Jurassic Park (1993)
Terminator 2: Judgment Day (1991)
Mission: Impossible (1996)
Rock, The (1996)
```

: Latent factor Model results for User 85

5.4 Evaluation of Results

We have qualitatively and quantitatively evaluated our results.

5.4.1 Qualitative Evaluation:

Content based gave results for 'Iron Man 3' as 'Iron Man', 'Iron Man 2', 'The Avengers' and so on. This means that it recommended items that are similar in content to items the user has liked in the past. Latent Factor Captured most of the movies that were actually liked by our 'user 85'. Item-Based also captured a few however User based didnt give very good results in terms of quality

5.4.2 Quantitative Evaluation:

For evaluation of results we used Mean average precision at K. For K we used the value as 100. We implemented our model with different number of factors (f), ranging from 30 to 100. We found out the precision for Collaborative Filtering - Item-Item for user 85 was 0.2294 and the precision for Collaborative Filtering - User Based for user 85 was 0.1674 The predicted movie precision for user 85 using Latent Factor model was 0.3514 In all, the latent factor worked the best of all.

6. CONCLUSION

6.1 Summarize

Content-based recommending approach recommends items that are similar in content to items the user has liked in the past, or matched to attributes of the user. For example, If a user commonly reads articles about iPhones or is likely to leave comments on blogs about apple gadgets, content-based filtering can use this history to identify and recommend similar content (articles on iPhones/apple products or other blogs about mobile devices). This content can be manually defined or automatically extracted based on other similarity methods.
In Collaborative Filtering systems a user is recommended items based on the past ratings of all users collectively. Collaborative filtering arrives at a recommendation that's based on a model of prior user behavior. The model can be constructed solely from a single user's behavior or — more effectively — also from the behavior of other users who have similar traits. When it takes other users' behavior into account, collaborative filtering uses group knowledge to form a recommendation based on like users. In essence, recommendations are based on an automatic collaboration of multiple users and filtered on those who exhibit similar preferences or behaviors.
Collaborative Filtering has some key advantages over others. Like, it can perform in domains where there is not much content associated with items, or where the content is difficult for a computer to analyze, such as ideas, opinions, etc. Latent Factor Model on the other hand gave commendable results and has improved the precision of recommendation.

6.2 State limitations of the study

Some of the common limitations in deploying Recommender Systems are as follows:

- **Scalability:** Algorithms such as the one we used in this project, work well with smaller amounts of data, but when the data sets grow, these algorithms can not behave robustly.
- **Sparsity:** In the dataset available, we find that most users have not rated most of the items and hence the user ratings matrix is typically very sparse. This is a problem for systems implementing Collaborative Filtering as missing ratings would make it difficult to find or group similar users/items.
- **Cold - Start Problem:** Recommender systems always give unsatisfactory results for the new user and items because there is little to no learning about the new user/item. Collaborative recommender systems tend to recommend only popular items and hence any new item will not be in the recommendations unless majority of users have rated it. Similary, without learning about the previous likes or dislikes of the new user the system can not predict recommendations or build a profile for new user.

6.3 State importance of findings

In conclusion we would like to suggest that implementing a hybrid approach, incorporating both content-based filtering and collaborative based filtering, creates the potential for a more accurate recommendation. This hybrid system will tremendously increase the customer satisfaction and thus, inturn increase the overall revenue of the concerned industry. Such approaches that leverage the strengths of content-

based and collaborative filtering are also increasing the efficiency (and complexity) of recommender systems. Thus a Hybrid system would prove to be more rewarding.

6.4 Announce directions for further research

Recommendation systems have become a crucial and popular component of social and e-commerce websites. They provide huge value to the consumers as well as to the owner of these companies. While it was fun to learn how to build one recommendation system, we also became familiar with the downsides of the simple traditional algorithms that exists. In future we would like to research more on robust and more-specialized approaches to build much needed hybrid algorithms that takes less processing time for the fast and real-time scenarios.

7. REFERENCES

[1] https://www.sciencedirect.com/science/article/pii/S1110866516300470

[2] http://eliassi.org/fa19dm.html

[3] https://grouplens.org/datasets/movielens/

[4] Z. Wang, X. Yu, N. Feng, Z. Wang An improved collaborative movie recommendation system using computational intelligence J Vis Lang Comput, 25 (2014), pp. 667-675

[5] Jafarkarimi H, Sim ATH, Saadatdoost R (2012) A naïve recommendation model for large databases. Int J Inf Educ Technol 2:216–219

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

[7] Choi, S.-M., Han, Y.-S.: A content recommendation system based on category correlations. In: The Fifth International Multi-Conference on Computing in the Global Information Technology, pp. 1257–1260 (2010)

[8] Wilson, D.C., Smyth, B., O'Sullivan, D.: Sparsity reduction in collaborative recommendation: A case-based approach. IJPRAI 17(5), 863–884 (2003)

[9] Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., Riedl, J.: GroupLens: Applying Collaborative Filtering to Usenet News. Communications of the ACM 40(3), 77–87 (1997)

[10] Shardanand, U., Maes, P.: Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In: Proceedings of CHI 1995, Denver, CO (1995)

[11] M. Deshpande, G. Karypis, "Item-based top-N recommendation algorithms", ACM Trans. Inf. Syst. 22 (2004) 143-177

[12] D.W. Oard and J. Kim, "Implicit Feedback for Recommender Systems", Proc. 5th DELOS Workshop on Filtering and Collaborative Filtering, pp. 31–36, 1998

[13] S. Funk, "Netflix Update: Try This At Home", http://sifter.org/˜simon/journal/20061211.html, 2006

[14] J. Bennet and S. Lanning, "The Netflix Prize", KDD Cup and Workshop, 2007. www.netflixprize. com.