A Content-based Movie Recommendation Approach with Link Prediction

Prepared by,

Sohini Bhattacharya, T91/CSE/174037

Srijeet Roy, T91/CSE/174041

Sneha Banerjee, T91/CSE/174047

Under the guidance of

Dr. Sankhayan Choudhury

Professor, Dept. of CSE, CU

Certificate

This is to certify that the project titled "**A Content-based Movie Recommendation Approach with Link Prediction**" submitted by Sneha Banerjee (T91/CSE/174047), Srijeet Roy (T91/CSE/174041) and Sohini Bhattacharya (T91/CSE/174037) under the supervision of Dr. Sankhayan Choudhury, is a bona fide record of the work done in partial fulfilment of the requirements for the award of the degree of Bachelor in Technology, Computer Science and Engineering, University of Calcutta for the session 2020-2021.

Sankhayan Choudhury

[Dr. Sankhayan Choudhury]
Supervisor

[Dr. Rajib Das]
Chairman, PG Board of Studies
Computer Science & Engineering

Acknowledgements

The completion of this report is owed to the authors and researchers who have contributed to this field of study. Their papers and associated resources made it possible for us to propose the potential scope of this project.

We would like to express our gratitude to Dr. Sankhayan Choudhury, Professor, Department of Computer Science and Engineering, University of Calcutta and Mr. Saubhik Goswami, Assistant Professor, Meghnad Saha Institute of Technology, for their guidance and assistance throughout the process.

Name	**Roll No.**	**Signature**
Sohini Bhattacharya	T91/CSE/174037	Sohini Bhattacharya
Srijeet Roy	T91/CSE/174041	Srijeet Roy
Sneha Banerjee	T91/CSE/174047	Sneha Banerjee

Index

1. Introduction

Recommendation is nothing but a process of suggesting something to someone such that the latter finds it interesting and worthy of his/her time and/or money. Recommendation systems are algorithms which work on this idea. The more its efficiency, the better the experience of the users on the social site. Suggestions can be for a variety of entities such as mutual users who are prospective "friends", products of user's interest or events to attend. Efficient recommendation systems can maximize profit of the concerned industry if they can optimize the information about social connections and make accurate and timely predictions.

Recommendation systems find applicability in e-commerce platforms such as Amazon, Myntra or Flipkart, in online movie sites such as Netflix or Amazon Prime, in social networking sites such as Facebook, Twitter or Instagram among other domains. These information about the product preferences of the customers or entertainment interests of viewers or social bonding among users can be derived using a number of parameters. Extensive research work has shown that either user history on the platform or user characteristics and demographics or both can be effectively used to make worthwhile recommendations. The recommender systems are designed such that they learn from previous user-item interactions to get an idea about user preferences, measure the degree of similarity between the chosen item and a prospective item to be recommended using several metrics and suggest the latter in case of high correlation. An additional information about the interests and social standing of the user can further refine this recommendation.

An efficient recommendation system must be modelled to carefully choose the useful information it gets as input, club user and item features with previous usage history and derive functional linkages between them which can accurately explain such past interactions. These latent linkages, if correctly manipulated, will generate a more personalized recommender system per user which can thereby increase the accuracy of our system.

1.1. Role of Link Prediction

Links among a user and an entity can be assumed if there exists some interaction among them. For an online shopping site, if a customer buys a particular product, there is a link between them, i.e., the product is, in some way, of the buyer's use and/or interest. Similarly, on an online streaming platform, a movie or a show watched by a user is somewhat linked to him/her; irrespective of whether he/she liked or disliked it, which can also be captured via user ratings. In social networks, two "friends" or "followers" have a link between them. Also, users following similar pages or liking similar posts or commenting on similar issues or attending related events also have an inherent linkage between them, which if properly recovered, can be of much use. The users are not necessarily individuals but can also be social media groups, real-world organizations or any kind of communities.

In any online social setting, there is a large number of entities which mutually interact, resulting in enormous amount of data, which can be a source of to study the pattern of such interactions, the demographics of the entities and also the social behavior in that setting. But more often than not, this raw data is scattered and needs to be preprocessed to give it an understandable structure from which information derivation becomes easier. For this purpose, conventionally, we represent any social setting as a graph, which can be directed or undirected, weighted or unweighted as the case may. The nodes or vertices represent the users, irrespective of whether they are individuals, groups or organizations. The edges or "links" capture the correlation or previous interaction between two or more such users.

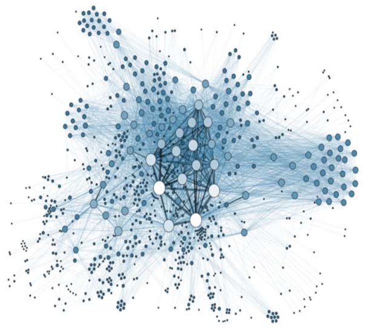


Figure 1: Graphical representation of a social network

In order to employ link information to recommendation systems, not only the existing links will suffice but we also need to understand the possibility of future, currently non-existent links. This is the task of link prediction [1], wherein we calculate the probability of new latent links between two or more nodes as is evident from the entity attributes and the previously recorded entity interactions.

To pose the problem in a more concrete way, let us consider the following snapshot of a social network at some time say t_1. With link prediction, we want to compute the probability if, at a later point in time, say t_2, some of the nodes that were not previously connected will form connections between them (Figure 2).

Link prediction finds wide applicability in domains of Detection, Influence Analysis and Recommendation. **Detection** in a social network is the process of information retrieval by closely monitoring, identifying and manipulating the underlying network structure. Detection can be either Anomaly Detection i.e., detecting fake user accounts or fraudulent transactions in social network or Community detection i.e., identifying groups or communities in social networks or Event detection which guides social network users to discover the latest and/or popular events. **Influence analysis** helps to understand people's social behaviour on social network, i.e., how the choices of one user are affected or "influenced" by the choices of his/her fellow user, and provide a theoretical basis for decision making and public opinion formation. **Recommendation**, as discussed earlier, can be extended beyond social network domain to real life utilities such as e-commerce or online entertainment platforms.

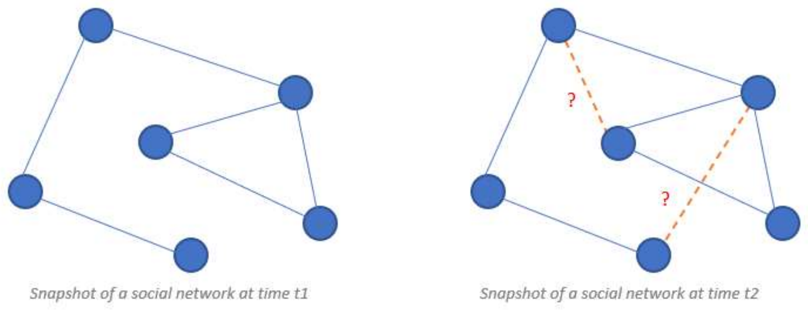

?
?
Snapshot of a social network at time t1
Snapshot of a social network at time t2

Figure 2: Link prediction in a social network.

Link prediction is particularly essential in movie recommendation systems that suggest relevant movies to the users based on their watched movies history. However, link prediction analysis tends to be more challenging with growing number of users and increasing number of movies, i.e., with increase in network complexity. If somehow the latent interactions between any two users and any two movies can be figured out, link prediction can be further eased and optimized. The user similarity in terms of their features and demographics or an idea about similar movies by exploiting their attribute information can bring in substantial improvement. In this paper, we discuss some of the recent developments and frameworks proposed by researchers which they used to solve the link prediction problem and incorporate it in their respective movie recommendation models. We compare the results and try to highlight the novelties and shortcomings of each.

1.2. Approaches of Link Prediction and Recommendation System

1.2.1. Contexts of Link Prediction

Context of an event or situation is the set of conditions or circumstances that surround it [2]. A social network's interpretation can be diverse depending on its context, which in turn can be determined by certain factors such as –

- *Network topology and features.* A network can be very dense with lots of edges between the nodes, or it can be sparse; nature of nodes (users) may vary within the same network and we must take a decision if we want to treat them equally or differently.

- *Information flow.* The content flowing throughout the network can determine its content, e.g., the contents of Facebook is a lot different from that observed in LinkedIn.

- *Analytical factors.* The problem statement might explicitly state what we intend to analyse or it may force us with a set of assumptions with which we ought to interpret the network.

Depending on all these factors, researchers have implemented numerous applications related to link prediction analysis in different network contexts [2].

- *Directed networks.* In a directed network context, all edges between two nodes have directions associated with them. For instance, in a 'follower' network, given two nodes x and y, a connection established between the two can be mutually connected such that x following y or y following x.
- *Weighted networks.* Edges are assigned with weights based on some criteria. For example, a network may have heavier the weights assigned to its links which show stronger ties between nodes.
- *Bipartite networks.* Nodes are partitioned into two different categories such that no two nodes of the same category are adjacent. Typical examples of a bipartite network would be author collaboration network.
- *Dynamic networks.* Most real-life social networks evolve with time - new nodes and edges are formed, many old nodes and edges disappear. Such temporal information can be critical to accurately model the network.
- *Heterogeneous networks.* When networks involve different kinds of nodes and links – each of them semantically belonging to distinct categorises. The types of nodes include users, posts and locations while the links among the nodes include social interaction links, location tags and photo tags.
- *Signed networks.* The relationship between two nodes can be labelled either as positive, negative or neutral. Traditional link prediction in unsigned networks considers all connections as positive links. Be that as it may, social interaction involves both positive and negative relationships such that people form links to distinguish friends from foes and vice versa.

1.2.2. Recommendation techniques

Recommendation systems are usually classified according to their approach to rating estimation into different categories [3]:

1.2.2.1. Collaborative methods

These consider that past user item interactions, which are stored in the so-called "user-item interaction matrix" [4]. Interaction information is sufficient to detect similar users or similar items and it makes predictions based on these. The two approaches are instance or memory based and model based:

- *Memory based approaches* work directly with values of recorded interactions. They assume there is no model and are mainly based on nearest neighbours' search. For example, it can find the closest users from a user of interest and suggest most popular items liked by user of interest among his/her neighbours. Types are user-user method and item-item method
- *Model based approach* assumes an underlying model that explains user item interactions and tries to make new predictions based on learnings of the model. Example is matrix factorization.

Advantage of collaborative filtering methods is that negligible user and item information is required, making it useful in many situations where user data is unknown. Also, as more users interact with items, the newer recommendations become more accurate making the system more effective for a fixed set of users and items.

One major disadvantage with collaborative approach is that it considers only past interactions to recommend, it suffers from "cold start problem", that is, it is impossible to recommend anything to new users or recommend a new item to any user. Also, many users have very few item interactions so predicting for them becomes tough. Researchers have proposed a number of strategies to circumvent the cold-start problem:

- *Random strategy.* Recommending random items to new users or new items to random users
- *Maximum expectation strategy.* Recommending popular items to new users or new items to the most active users.
- *Exploratory strategy.* Recommending set of various items to new user or new items to set of various users.

1.2.2.2. Content based methods

These take into account additional information about user or item along with previous interactions to make recommendations. For example, for a movie recommendation system, user's age group, favourite genre, preferred languages or gender can be considered and for the item, that is the movie, its cast, duration, genre or year of release may be considered. Then a model is built based on above features which can explain observed user item interactions. Such modelling can give results that young men like action movies or middle-aged women like family drama, which will ease making new predictions. By looking at a user's profile, the system will suggest relevant movies.

Content based method assume latent interaction model, which again is provided with content that defines representation of users and items, that is, the features. So, they have a very high bias and low variance making it more personalized experience for individual user. On the other hand, content-based techniques can solve the "cold start problem" as new users or new items can be described by their features and relevant suggestions can be done for them by following the model. However, significant research is yet to be performed on content-based methods that involve user demographics.

1.2.2.3. Hybrid methods

They combine collaborative filtering and content-based approaches and are more commonly used [5]. It is of two forms: either two models are independently trained where one uses collaborative filtering method and the other is content based model and then they are combined; otherwise, a single model clubs both approaches by using user and item information as inputs as well as previous user-item interactions.

1.3 Scope of our work

We aim to design a Movie recommendation system using link prediction. All previous work in this domain has considered user ratings and/or movie genre as a basis for making future recommendations to new or existing users. Our idea is to not only restrict the input to previous user-movie interactions but to extend it to the respective feature sets of the users and the movies. This information about the characteristics of the users and the movies will allow us to find the degree of similarity between two or more such entities. The more similar they are, the stronger is the link between them, and this link in turn can be used for recommending similar movies to similar users. Utilizing user attributes will also provide an in-built mechanism to address the cold start problem for users who have been newly introduced to the network or the ones who do not have significant interaction information.

2. Related Works

Recommender systems can be defined as programs which attempt to recommend the most suitable items to particular users by predicting a user's interest in an item. A wide range of work have been done by using both the Content-Based or Collaborative techniques as well as the Hybrid techniques to recommend products [6-20]. Here we are trying to capture the importance of these proposed methods in the context of their beneficence towards the studies of recommendation system in a bipartite network.

In content-based (CB) recommendations the users are recommended items similar to the ones they preferred in the past. These systems, thus, must maintain an item profile that is basically a set of attributes belonging to the item [6-8]. Collaborative filtering (CF) is a technique to predict a user's taste and find the items that a user might prefer on the basis of information collected from various other users having similar tastes or preferences [9-11]. Content based technique does not consider customers' attitude towards the product, which limits the accuracy of recommendation. On the other hand, collaborative filtering technique does not consider the product attributes which are pivotal for a recommender system. Hybrid recommender systems use a combination of both content-based and collaborative filtering methods to provide recommendations. Unifying both approaches help to circumvent the drawbacks of using content-based or collaborative methods alone [12][13].

The above algorithms for recommender systems suffer from many issues. For example, in order to measure item similarity, content-based methods rely on explicit item descriptions. However, such descriptions may be difficult to obtain for items like ideas or opinions. Traditional Collaborative Filtering has the data sparsity problem and the cold-start problem. In contrast to the huge number of items in recommender systems, each user normally only rates a few. Therefore, the user/item rating matrix is typically very sparse. It is difficult for recommender systems to accurately measure user similarities from those limited number of reviews. Another related problem is the cold-start problem. Even for a system that is not particularly sparse, when a user initially joins, the system has none or perhaps only a few reviews from this user. Therefore, the system cannot accurately interpret this user's preference.

In this study, for making movie recommendation systems, several approaches have been proposed to solve the above problems and improve the accuracy of a movie recommender system. In this purpose researchers have used movie and user attributes with user-movie ratings to improve the accuracy of the recommender system. Kim et al. [14], addressing the cold start problem by introducing a model based collaborative filtering approach based on explicit movie ratings. In this model, they have tried to predict actual ratings and subsequently identifies prediction errors for each user by using Matrix Factorization method. This model is then used to recommend movies whose rating predictions are higher for that user. In order to enhance the performance of the system and to solve cold start problem, Pereira et al. [15] posed a hybrid method including both collaborative filtering and demographic information. This approach is based on an existing algorithm, SCOAL (Simultaneous Co-Clustering and Learning) [16], and finds groups of users with common interests and simultaneously build prediction models for recommendation that are specific to each group. Lin et al. in [17], developed a recommendation system based on collaborative filtering and neural network for recommending movies to users. In the proposed model, if a new movie comes in, it used the collaborative filtering based on movie information to predict the user's rating of the movie. If a new user comes in, then the system used user information to find similar users and use collaborative filtering to predict the rating. On the other hand, if it is not a new user, the system used historical rate record to find similar users and use collaborative filtering to predict the rating. Finally, it used matrix factorization to calculate the final score prediction table and choose the Top-N of movies to recommend. Walek et al. in [18], proposed a hybrid recommender system combining a collaborative filtering system using SVD algorithm, a content-based system, and a fuzzy expert system. The proposed system works with favorite and unpopular genres of the user, while the final list of recommended movies is determined using a fuzzy expert system, which evaluates the importance of the movies. The expert system works with several parameters – average movie rating, number of ratings, and the level of similarity between already rated movies. In [19], authors have proposed a model that tries to improve the traditional Collaborative Filtering (CF) algorithm by considering item's characteristics. Here the authors have used global similarity, local similarity and meta similarity in vector form to calculate similarity between users. In global similarity, the authors have considered all different kinds of nodes available in the network and here they measured the similarity based on genres. Whereas in local similarity they have measured the average similarity score based on all the items and finally they compute the meta similarity based on a single item. In this paper the authors have classified the item part into another level to get more accurate similar users in terms of items but they did not consider the user's features to measure similarity between users. In this paper [20], authors have tried to improve the traditional collaborative filtering method by adding a link prediction (improved Friendlink algorithm) mechanism. Here they have first tried to correlate users using traditional CF algorithm and then increase the similarity score between those highly correlated people by applying an improved Friendlink algorithm. In the link prediction mechanism, they have considered the following topological features od users: degree of nodes, node popularity, number of edges between two users but they didn't consider movie features which may lead to inaccurate result for similar movie recommendation.

The existing approaches mentioned above either have not taken user's and movie attributes into consideration simultaneously or used it separately or lack in its proper inclusion which may affect in the accuracy of a movie recommender system. The intended solution addresses these problems and comes out with a better recommendation approach that may serve the purpose of the real world and address the cold-start problem

3. The Problem & The Proposal

3.1. Problem Statement

Given a user-item (movies) interaction system, the objective is to predict future links between the users and items based on past interactions between the entities which is personalized with the help of attributes that characterize the users and items. In our problem involving the heterogenous network, there exists no explicit connections qualifying interactions among instances of the same entity type and the attributes characterizing the entities do not have any assumed correlation. Furthermore, the links are weighted to highlight the sentiment, and the intensity thereof, of the users towards the items. Exploiting the intensity of such sentiments (from positive and strongly positive to negative and strongly negative), the link prediction problem can be posed as a classification task that classifies unforeseen links as positive (hence recommendable) or negative (not recommendable).

With the above-mentioned features, we can design the network as a weighted and attributed bipartite graph which will facilitate the visualization as well as implementation of the proposed solution approach.

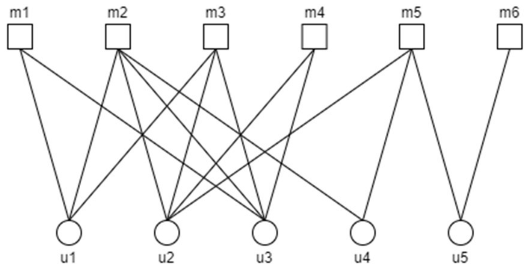


Figure 3: User-Item Bipartite Graph

3.2. Problem Formulation & Proposal

Given a weighted and attributed bipartite graph, $G = (V_G, E_G)$ where the vertex set $V_G = M \cup U$, where M is the set of items (movies) such that $\forall M_i \in M$, M_i is characterized by the attribute set $\{m_1, m_2, \dots, m_x\}$ and U is the set of users such that $\forall U_i \in U$, U_i is characterized by the attribute set $\{u_1, u_2, \dots, u_y\}$, and the edge set E_G captures user-movie interaction as a set of weighted edges between M and U, where weights correspond to the rating provided by a user in U to a movie in M, the objective is to predict future undirected edges between nodes in M and U. Based on the weights of each link, the links can be labelled as positive (hence recommendable) or negative (not recommendable). Designing a binary classification model that can predict this label for an unforeseen link between a user and a movie, when fed with embeddings capturing the basic and derived attributes of the entities, can produce a solution for the recommendation problem.

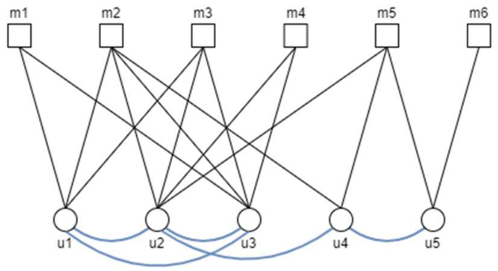


Figure 4: User-User Similarity Links

For a given user node $u_i \in U, U \subset V_G$, characterized by the attribute set $\left\{u_1^i, u_2^i, \dots, u_y^i\right\}$, it is possible to obtain an embedding consisting of these basic attributes along with feature vectors capturing its interactions with movie nodes and features of other user nodes most similar to it as measured from both topological and behavioral perspectives. Let $M_{u_i} = \left\{M_{u_i}^1, M_{u_i}^2, \dots, M_{u_i}^p\right\}$ be the p movie nodes connected to the user node u_i where each movie is attributed by its own attribute set.

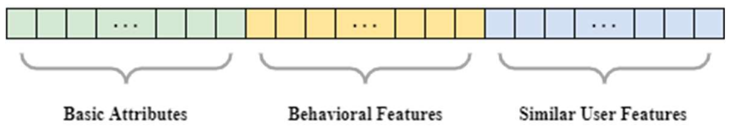


Figure 5: User Embedding

A user's behavior in terms of interactions with the movies can be described by combining the movie attributes in a meaningful way. For example, combining the genre information about all the movies a user has watched can provide quantitative insights about the preferences of the user. Using the primitive attributes, the behavioral traits along with the position of the user node in the network topology, we can define similarity metrics that will enable us to establish latent links between the users (Figure 4). These latent links qualify the similarity of the users in the aforementioned aspects. As a result, we can extract a more generalized feature vector for a user. Combining all the above segments of features, we obtain a final user embedding (Figure 5).

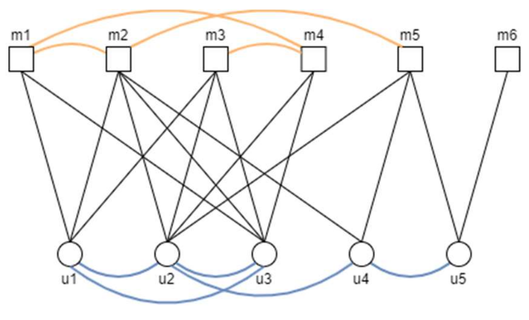


Figure 6: Latent Links among Movie Nodes

Similarly, for a given movie node, we already have the attributes that characterize the movie. Along with it, we can obtain information about the other movies in the system which are most similar to the concerned movie and find latent links among the movies based on such similarity scores (Figure 6). The movie embedding (Figure 7) is simply the attribute vector and similar movies feature vectors, appended next to each other.

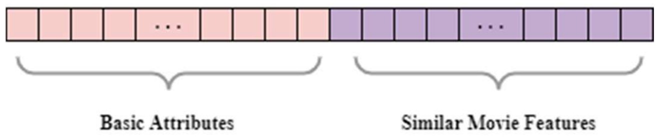


Figure 7: Movie Embedding

For a given user-movie pair, we can define the corresponding user and movie embeddings. The combination of these two embeddings can be used as an input to a binary classifier that will predict whether the potential link between the user and the movie will be positive or negative. The n-dimensional joint embedding is nothing but a data point in the n-dimensional space where the binary classifier attempts to learn the best decision boundary to separate the two classes.

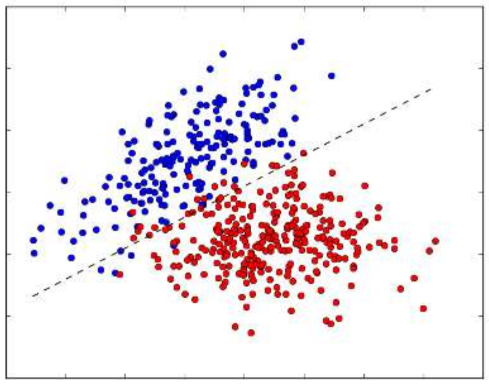


Figure 8: Illustration of a Binary Classifier (projected onto a 2D space)

4. Experimentation

4.1. Dataset

The proposed model was evaluated on the MovieLens dataset [21]. This dataset consists of 1 million movie ratings provided by 6040 users and 3883 movies. The entire dataset is provided in three separate files – one each for users, movies and ratings. The reason behind selecting this dataset is that it contains user demographic information, which most benchmark datasets in this domain lack. The features of the dataset are further discussed in the following sections.

4.1.1. Users

The user dataset contains basic demographic information about the 6040 unique users who joined MovieLens in 2000. Each entry in the dataset has the following attributes:

- *User ID.* Each user is assigned a unique identification number.
- *Gender.* The gender of a user is denoted by an "M" for male and "F" for female.

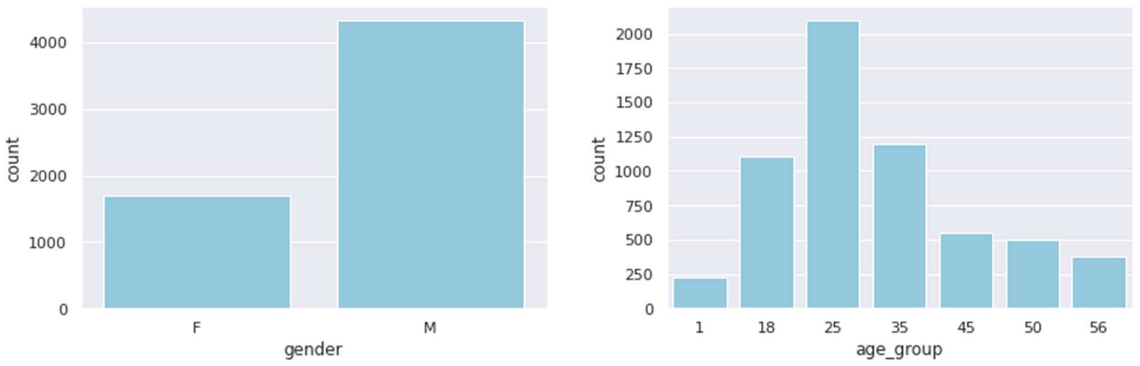


Figure 9: Gender and Age Group Distributions in the Users Dataset

- *Age.* Based on their age, users are segregated into age groups (Table 1) and assigned a tag.

Age Tag	Age Group
1	"Under 18"
18	"18-24"
25	"25-34"
35	"35-44"
45	"45-49"
50	"50-55"
56	"56+"

Table 1: User age groups

- *Occupation.* Each user can have only one of the occupations as shown in Table 2 and the corresponding tag is assigned to the user.

Tag	Occupation	Tag	Occupation
0	"other" or not specified	11	"lawyer"
1	"academic/educator"	12	"programmer"
2	"artist"	13	"retired"
3	"clerical/admin"	14	"sales/marketing"
4	"college/grad student"	15	"scientist"
5	"Customer service"	16	"self-employed"
6	"doctor/health care"	17	"technician/engineer"
7	"executive/managerial"	18	"tradesman/craftsman"
8	"farmer"	19	"unemployed"
9	"homemaker"	20	"writer"
10	"K-12 student"		

Table 2: User occupations

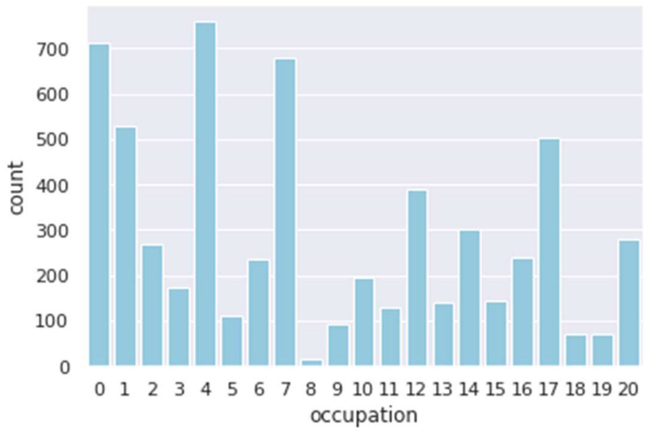
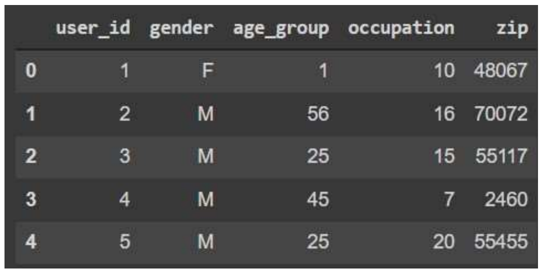
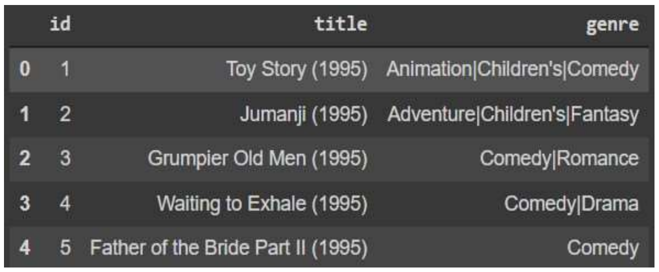


Figure 10: Occupation distribution in Users Dataset

- *Zip-code.* It is a real numeric code which maps to user's location (Figure 10).



	user_id	gender	age_group	occupation	zip
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	2460
4	5	M	25	20	55455



	id	title	genre
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

Figure 11: Users and Movies Datasets

4.1.2. Movies

The movie dataset consists of information, namely, movie ID, title and genres of 3883 unique movies.

- *Titles.* The movie titles are identical to titles provided by the IMDB, including year of release which is mentioned in parentheses following the title.

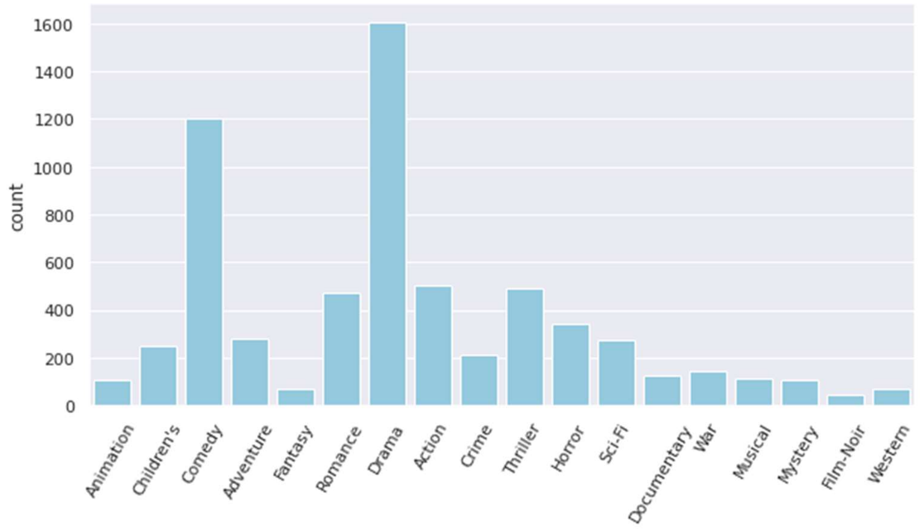
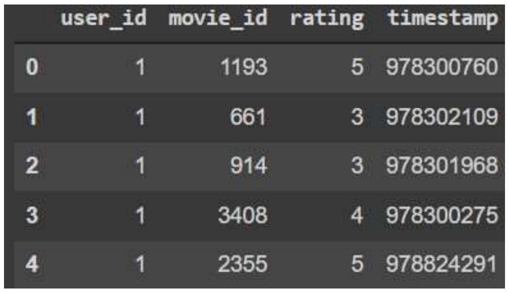


Figure 12: Movie Genre Distribution

- *Genre(s).* A movie can belong to one or more genres. In the latter case, the genres are pipe-separated. The movies can only belong to the following set of 18 genres (Figure 12).



	user_id	movie_id	rating	timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

Figure 13: Ratings Dataset

4.1.3. Ratings

The rating dataset provides the rating information by linking users and movies to capture how much a given user has rated a given movie which he/she has watched, as well as the exact timestamp of such rating. It consists of 1000209 of such ratings.

- *UserID*. It identifies a unique user from the user dataset who has rated a particular movie.
- *MovieID*. It identifies a unique movie from the movie dataset which has been rated by a particular user.
- *Rating*. Users rate a movie on a 5-star scale (Figure 14), i.e., from 1-5, 1 being the lowest and 5 being the highest. There is no provision for fractional or decimal point ratings, i.e., whole-star ratings allowed only.

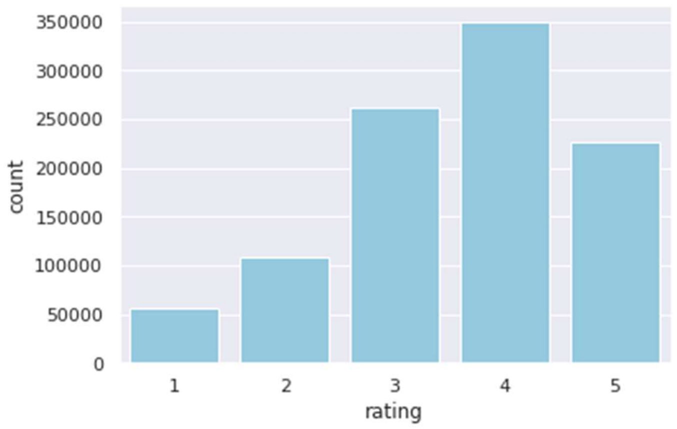


Figure 14: Movie Ratings Distribution

- *Timestamp*. The temporal information about a particular rating.

Each user has to have at least 20 ratings to be included in the Rating dataset.

4.1.4. The Movies Dataset

The degree of information that can be collected about the movies from the above dataset is not sufficient for drawing any meaningful conclusion which can assist our model derive appropriate results. So, additionally we have used another dataset called “The Movies Dataset” [22]. This dataset contains metadata on 45,000 movies from the MovieLens Dataset.

We have used this dataset by finding the common movies from our original dataset and merging the additional information obtained from the prior in order to extract richer embeddings for the movies. The usage details will be enumerated later.

The extended movie dataset thus generated is consistent and factually correct to proof. The data has been unbiasedly collected for usage in our solution and not to manipulate the final outcome.

4.2. Data Preprocessing

4.2.1. Rating dataset

The original rating dataset had information about user ID, movie ID, rating value and timestamp.

- Timestamp has little significance in determining the sentiment of the rating. Hence, it is omitted.
- Each rating corresponds to a user-movie interaction. There are 1000209 such ratings.

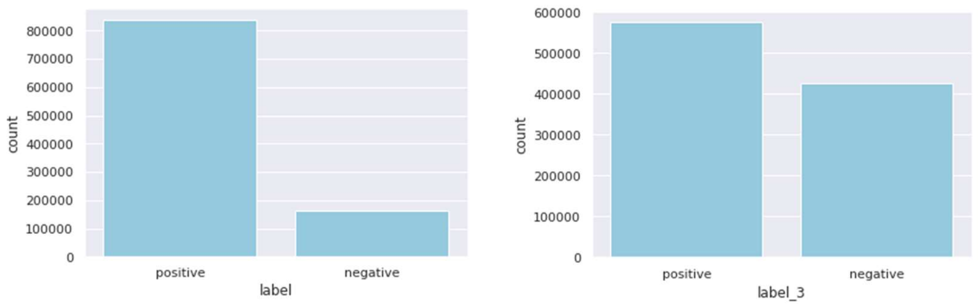


Figure 15: Label distribution based on different thresholds

- In order to pose the problem as a binary classification task, we label each rating as either "positive" or "negative" based on the exact rating (between 1 and 5) given by the user. If the threshold value is chosen as 3, i.e., a rating of at least 3 is labelled as positive, then the dataset becomes unbalanced (about 80% positive to 20% negative) ('label' in Figure 15). So, the threshold value was chosen as 4 ('label_3' in Figure 15), as shown in Table 4.

User rating	User-Movie interaction (Class label)
>=4 (i.e., 4 or 5)	Positive
<4 (i.e., 1 or 2 or 3)	Negative

Table 3: Positive and negative labels

- The processed rating dataset is segregated into a training set and a testing set using Scikit-Learn's train_test_split. The training set contains 75% of the ratings and the testing set contains the remaining 25%. Hence, the size of training and testing sets are 750156 and 250053 respectively.

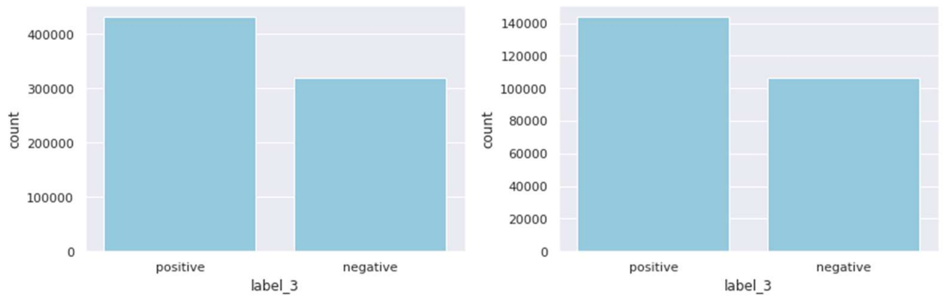


Figure 16: Training and test set distributions

4.2.2. User dataset

The original user dataset had information about User ID, age group, gender, occupation and zip code

- Gender was captured as M/F for male/female. This information is converted into a single-element binary format, where F maps to 0 and M maps to 1.
- The age tag represented the age group of the user, but the tag values were not consecutive. For computational ease, we map the tags to increasing numerical values as shown in Table 5.

- Occupation had to be one from the 21 numerical tags. For clarity, a 21-length one-hot encoding of the occupation classes is considered. The idea behind using one-hot encoding is to promote equal dissimilarity between all the occupations. Supposing, user A is an educator, B is a scientist and C is a homemaker. There is no way to deduce whether A is more similar to B or to C. As our model lacks the context awareness to understand the degree of similarity between the occupation of two or more users, we design all the occupation categories to be equidistant is terms of dissimilarity.

Age tag	New age tag
1	0
18	1
25	2
35	3
45	4
50	5
56	6

Table 4: User Age Groups

- Zip-codes were provided as real numbers. Some of the outliers and erroneous values were replaced with the median value of the entire set of zip-codes. Then, the raw zip-codes were converted into categorical "zip-zones". The entire set of zip-codes was grouped into equal subsets, each consisting of 5000 consecutive codes, and the zip-zones were named in a numerically increasing manner, from zone 0 to 19. Accordingly, a user is assigned the zip-zone which contains the zip code he/she has entered.

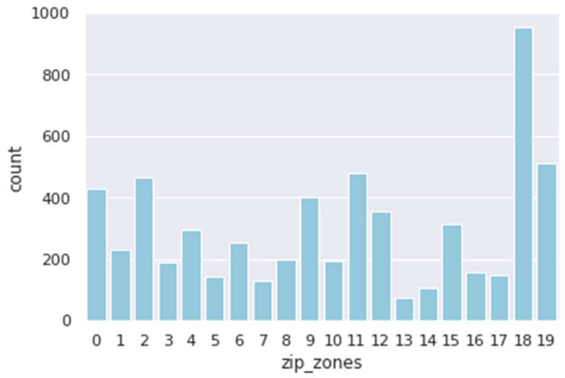


Figure 17: Zip zones distribution in Users dataset

4.2.3. Movie dataset

We have merged 3883 unique movies from the MovieLens dataset by matching their title and year of release with those in The Movies Dataset. This has provided us with additional information for each movie such as its original language, production country, IMDb rating, number of user-votes it has received on IMDb to name a few. This is followed by processing for computational ease.

- The year of release is made into a separate column.
- Genre is converted into an 18-length binary feature vector, with all elements as 0 initially. Each element is assigned a genre, i.e., the element at the 0th position corresponds to "Action", the 17th position corresponds to "Western" and so on. A movie can have one or more than one genre so one-hot encoding is ruled out. For a given movie, the elements which correspond to its genres are converted from 0 to 1, thus generating an 18-length binary feature vector with at least one 1.
- Original release language can only be one. This is handled by using one-hot encoding to select the release language out of 34 unique languages in The Movies Dataset.

- Production country can be one or more than one. Hence, it is handled by using a binary feature vector which signals the production countries from a set of 70 unique countries.
- IMDb rating and number of user votes are used to derive a “score” for a given movie. The intuition behind this score is to capture the net rating (in terms of audience sentiment) of a movie along with its popularity (in terms of the number of users who have rated the movie). Supposing a movie A has 9 rating and is rated by 20 users, whereas a movie B has 9 rating as well but is rated by 20,000 users, the model will be trained to conclude that B has a better popularity score than A. To achieve this, we use the product of the average user rating and the number of user votes as the popularity score.

4.3 Creating the Bi-Partite Graph

As is evident from the rating dataset, the only interactions which exist are those between users and movies. As we know, mathematical graphical representation is an intuitive way to visualize network structures. We consider users as nodes of one type (S_1) and movies as nodes of another type (S_2). There are neither any user-user link nor any movie-movie links in our network, only user-movie links or edges exist. This results in formation of a bipartite graph, which by definition is “a graph whose vertices can be divided into two disjoint and independent sets S_1 and S_2 such that every edge connects a vertex in S_1 to one in S_2”.

Using the training dataset, we construct a bipartite graph having the following specifications:

- 750156 edges, the same as the number of ratings (user-movie interactions) in the training dataset.
- 9708 nodes of which 6040 are from one partite set, corresponding to the 6040 users who have rated some movies and 3668 belong to the other partite set corresponding to the 3668 unique movies assigned a rating by users in the training dataset.

An interesting observation to note here is that the neighbors of a node corresponding to a user, are nodes that all correspond to movies and vice versa. No user node has another user node as a neighbor and no movie node has another movie node as its neighbor. We use this characteristic in order to define novel comparison metrics among the nodes.

4.4. User-User Similarity

In this section, we define the various comparison metrics that we used to capture the similarities between two given users from different perspectives. A linear combination of these metrics is considered as the net similarity between the users.

4.4.1. Topological Metrics

We use some popular common neighbor-based metrics that take advantage of the network topology to infer the similarity of two nodes in the network [23]. While most of these metrics are based on the degree of connectedness of a given pair of nodes, some context-specific intuitions justify the use of these metrics.

(i) Common Neighbors (CN) - Common neighbors captures the idea that two users who have a friend in common are more likely to be introduced than those who don’t have any friends in common. In other words, two vertices are more likely to be connected if they are connected to the same set of other vertices. Mathematically,

$$s_{xy} = |\Gamma(x) \cap \Gamma(y)|$$

where $\Gamma(x)$ is the set of nodes adjacent to user node x and $\Gamma(y)$ is the set of nodes adjacent to user node y.

(ii) Jaccard Coefficient (JC) - The Jaccard similarity coefficient (JC) compares members for two sets to see which members are shared and which are distinct. In other words, it measures the proportion of common neighbours in the total number of neighbors. It reaches its maximum if $\Gamma(x) = \Gamma(y)$, i.e., if all neighbours are common to both vertices. Mathematically,

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

(iii) Sørensen Index (SI) - The Sørensen index equals twice the number of elements common to both sets divided by the sum of the number of elements in each set. It measures the relative size of an intersection of neighbours' sets. Mathematically,

$$s_{xy} = \frac{2|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)| + |\Gamma(y)|}$$

(iv) Leicht-Holme-Newman Similarity (LHN) - Leicht, Holme and Newman proposed a measure of vertex similarity based on the concept that two vertices are similar if their immediate neighbors in the network are themselves similar. Mathematically,

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)| * |\Gamma(y)|}$$

(v) Salton Cosine Similarity (SCS) - The Salton cosine similarity measures the cosine of the angle between columns of the adjacency matrix, corresponding to given vertices. This measure is commonly used in information retrieval. Mathematically,

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{\sqrt{(|\Gamma(x)| * |\Gamma(y)|)}}$$

(vi) Hub Promoted Index (HPI) - The Hub Promoted Index measure assigns higher scores to links adjacent to hubs (high-degree vertices), as the denominator depends on the minimum of the degrees of the vertices of interest. Mathematically,

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{\min \{|\Gamma(x)|, |\Gamma(y)|\}}$$

(vii) Hub Depressed Index (HDI) - The Hub Depressed Index measure assigns lower scores to links adjacent to hubs (high-degree vertices). It penalises large neighbourhoods. Mathematically,

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{\max \{|\Gamma(x)|, |\Gamma(y)|\}}$$

(viii) Resource Allocation (RA) - Resource allocation measure is motivated by a resource allocation process. It measures how much resource is transmitted between nodes x and y. Mathematically,

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{|\Gamma(z)|}$$

where $\Gamma(z)$ is the set of neighbors of z, which, in turn, is a common neighbor of nodes x and y.

(ix) Adamic-Adar Index (AA) - Adamic and Adar develops simple counting of common neighbours by assigning weights to nodes inversely proportional to their degrees. This means that a common neighbour, which is unique for a few nodes only, is more important than a hub (high-degree vertex). Mathematically,

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}$$

In the context of a bipartite user-item interaction network, Adamic-Adar index translates to the notion which discourages us to conclude that two users are similar based on popular movies that they have both watched. Degree of a movie node is the number of user nodes it is connected to and thus it can be interpreted as its popularity. Two users who show interest in niche movies, i.e., movie nodes with lower degrees, are likely to be more similar than those who share popular common movies.

4.4.2. Common Movie Rating Similarity (CMRS)

In this section, we introduce an intuitive similarity metric that compares two users based on their sentiment towards the movies. Beyond only considering the sentiment of users with respect to their overall genre preference (as discussed in the section 6.3.3), this metric observes their behavior from a low-level. For two given users, we find which are the common movies which they both have watched, their corresponding ratings for each such common movie and then perform element-wise rating comparison.

The process of finding movies that two given users have both watched, is facilitated by the graphical representation of the data and it boils down to finding the common neighbors of two user nodes in the graph. For two users u and v, let $M = \{M_1, M_2, \dots, M_n\}$ be the n common movies. Let $R_u = \{R_{1u}, R_{2u}, \dots, R_{nu}\}$ be the ratings provided to the movies in M, as provided by user u and $R_v = \{R_{1v}, R_{2v}, \dots, R_{nv}\}$ be the same for user v. To compare the two ratings vectors, we use cosine similarity. Instead of comparing the absolute values of the individual vector elements, cosine similarity takes into account the relative orientation (cosine of the angle between two vectors) of the vectors in space. Mathematically, cosine similarity of two vectors A and B in the n-th dimension is defined as,

$$cosine\ similarity = \frac{A \cdot B}{||A||\ ||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2}\ \sqrt{\sum_{i=1}^{n} {B_i}^2}}$$

The cosine similarity of the vectors R_u and R_v is defined as the common movie rating similarity (CMRS).

4.4.3. Genre Preference Similarity (GPS)

From a general user-behavior perspective, we can define a similarity metric that compares the genre preferences of two given users and conclude how similar they are in terms of the types of movies they tend to watch more often.

For a given user node u in the bipartite graph, all of its neighbors, $\Gamma(u)$ are movie nodes and from the data available, we can accumulate the genre information for each such movie. Let $GENRE_m$ be the one-hot encoded vector of the genre details of a movie m; then for a given user u, we obtain a count vector, $GENRE^u$ by elementwise adding the one-hot encoded vectors, $GENRE_m$ of all the movies that user has watched, i.e., for all $m \in \Gamma(u)$.

Let u and v be two users. Let $GENRE^u = [G_1^u, G_2^u, \dots, G_{18}^u]$ (as there are 18 genres) be the genre preference vector of user u and $GENRE^v = [G_1^v, G_2^v, \dots, G_{18}^v]$ be that for v. To compute the genre preference similarity (GPS) between the two users, we again consider cosine similarity of the two vectors $GENRE^u$ and $GENRE^v$.

4.4.4. User Feature Similarity (UFS)

The idea is to explore a pair of users' basic attributes or demographics to find the degree of similarity between them. This is crucial in order to address the cold start problem which makes it difficult to recommend relevant movies to a new user or a dormant user due to nil or less previous user-movie interactions respectively.

A user p, newly introduced to the network, has watched or rated no movies; it's a user node of degree 0 in the bipartite graph. Hence, the topological metrics, which are essentially based on the connection of a node to the network, as well as metrics such as CMRS and GPS similarity cannot be defined. Using user feature similarity, we can tackle this issue. Our ultimate goal is to exploit user demographics along with past user-movie interactions (wherever they exist) to get an optimal output.

We design a user feature vector for each user with specifications as follows:

- Occupation, a 21-length vector, is maintained in one-hot encoding format. Each occupation is equally dissimilar to the next, so as to get unbiased results.
- Unit-length integer type Zip-zones represent closeness in location, i.e., users from zip-zones 0 and 1 are concluded to be more similar in proximity compared to users from zip-zones 0 and 10.
- Gender is maintained in binary format as a unit-length switch.

- Age group is maintained as numerically increasing age tags, via a unit-length integer type vector.

For two users u and v, user feature similarity score is the cosine similarity between vectors F_u and F_v, where F_i is the combined vector of user attributes in the abovementioned formats for user i.

4.4.5. Net Similarity between Users

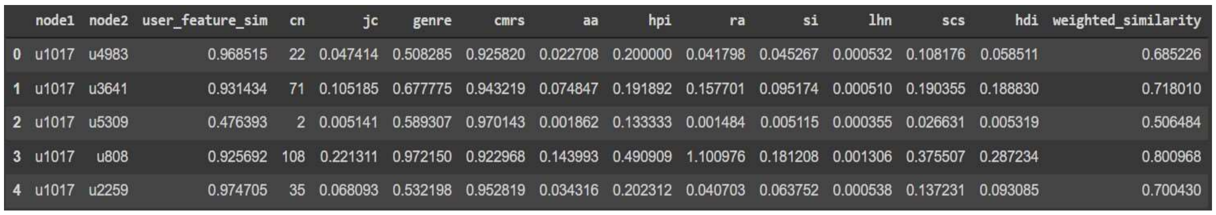
It is understandable that the net or overall similarity between two users is a function of the similarity metrics designed above. The net similarity is computed as the weighted sum of CMRS, UGS, UFS and the topological metrics. The weight to be assigned to each of the metrics is directly proportional to the contribution of that metric in deriving the overall similarity. This degree of importance is unknown and so the weights cannot be directly deciphered. As a consequence, we employ these weights as hyperparameters as shown in Table 6.

Metric	UFS	CMRS	GPS	AA	JC	HPI	RA	SI	LHN	SCS	HDI	CN
Weight	0.4	0.2	0.2	0.1	0.025	0.025	0.01	0.01	0.01	0.01	0.01	0

Table 5: Weights assigned to different metrics

The intuition behind the weights assigned to the respective metrics is as follows:

- User feature similarity (UFS) score is assigned the highest weight as it will tackle the cold start problem single-handedly for new incoming users. Furthermore, increased emphasis on basic user attributes translates to a more personalized recommendation system, that will not be solely dependent on the structural properties of the network a user is part of.
- As stated earlier, CMRS captures the user behavior from a low-level as it compares users' sentiments towards individual movies. In other words, CMRS decides how similarly the user pair will react, positively or negatively, to the movie, which is important to make effective recommendations. Hence, CMRS is assigned the next highest weight. Users' genre preference similarity, which captures user behavior from a comparatively higher level than CMRS, is equally important and is assigned the same weight as CMRS.
- Topological metrics, other than common neighbors, are assigned comparatively lower weights so as to minimize dependency on network structure. The number of common neighbors is not involved in the computation of net similarity since it does not reflect user similarities in a consistent scale. For example, let us assume user u and v have watched 100 and 80 movies respectively and the number of common neighbors the two nodes have in the graph representation is 40. On the other hand, let p and q be two more users who have watched 30 and 25 movies respectively and the common neighbors count for them is 20. Although the metric score for the previous pair is higher, we can clearly see that users in the latter pair is should be treated as more similar.



```
   node1  node2  user_feature_sim   cn        jc     genre      cmrs        aa       hpi        ra        si       lhn       scs       hdi  weighted_similarity
0  u1017  u4983          0.968515   22  0.047414  0.508285  0.925820  0.022708  0.200000  0.041798  0.045267  0.000532  0.108176  0.058511             0.685226
1  u1017  u3641          0.931434   71  0.105185  0.677775  0.943219  0.074847  0.191892  0.157701  0.095174  0.000510  0.190355  0.188830             0.718010
2  u1017  u5309          0.476393    2  0.005141  0.589307  0.970143  0.001862  0.133333  0.001484  0.005115  0.000355  0.026631  0.005319             0.506484
3  u1017   u808          0.925692  108  0.221311  0.972150  0.922968  0.143993  0.490909  1.100976  0.181208  0.001306  0.375507  0.287234             0.800968
4  u1017  u2259          0.974705   35  0.068093  0.532198  0.952819  0.034316  0.202312  0.040703  0.063752  0.000538  0.137231  0.093085             0.700430
```

Figure 18: User-User Similarity

4.4.6. Top 'N' Similar Users

After comparing each user with all the existing users in the training set, we obtain the N most similar users from the results. For our solution, $N = 10$, i.e., we are only concerned with 10 most similar users for a given user. The attributes and behaviors of these similar users allow us to implicitly generalize a user's characteristics in the user space. The basic attributes of the 10 users were combined via element-wise addition of individual feature vectors where each categorical attribute was converted to its one-hot encoding form. The combined vector thus obtained is used as part of the user embedding.

4.5. Movie-Movie Similarity

4.5.1. Movie Feature Similarity

Various information about the movies like genre, original release language, production company, average user rating and number of user ratings (together forming popularity score) are clubbed to form a complete movie feature vector. Categorical attributes can be represented in a binary feature vector form (such as, movie genres and production countries) and one-hot encodings (e.g., original language) depending on whether a movie can have more than one values of the said attribute at the same time or not. To compute similarities between a given pair of movies, we consider the cosine similarity of the corresponding movie feature vectors.

4.5.2. Top 'N' Similar Movies

Similar to user-user similarity computation, we compute movie-movie similarities and obtain the N most similar movies for each movie. The aim, again, is to characterize a movie with features of a wider number of movies in an attempt to generalize the individual vectors. For our proposed solution, we consider $N = 10$, i.e., we collect features of top 10 similar movies and club the vectors in a way similar to the users' case, i.e., by elementwise adding the one-hot encoding or binary feature vectors of categorical attributes and by taking mean of real valued attributes (namely, year of release and popularity score).

4.6. Prediction Model

Each data sample in the training set consists of a three tuple: *(user_id, movie_id, label)* where the label is positive if the rating is greater than 3 and negative otherwise. For a given user with ID *user_id* and a given movie with ID *movie_id*, we obtain their user and movie embeddings using the techniques discussed in section 4.4 and 4.5.

The user embedding consists of three parts – user's basic attribute vector (of length 24), user's genre preference vector (of length 18) and similar users feature vector (of length 50). As a result, we obtain an embedding of length 92. The final movie embedding is a 248-element feature vector, where the first 124 elements are movie basic features and the last 124 elements are features of top N similar movies.

The input to the prediction model is a joint embedding vector of user and movie, as available in the training set, of length 340. The classifier attempts to predict the class label for this vector. In the testing set, this knowledge is exploited and the rating values are taken as a use case. A user and a movie are extracted, and the model, after learning, is expected to accurately predict whether the user will rate the movie >=4, i.e., whether a positive link is expected to exist between the given user-movie duo. A positive link will in turn mean that the movie can be recommended to the user.

5. Evaluation Metrics & Results

5.1. Evaluation Metrics

5.1.1. Root Mean Square Error (RMSE)

RMSE, also called Root Mean Square Deviation (RMSD) is the standard deviation of the residuals (prediction errors), i.e., a measure of how spread out these residuals are. Residuals are a measure of how far from the decision boundary data points are. In other words, it records how concentrated the data is around the line of best fit. Mathematically,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}$$

where y_i is the observed value or the ground truth and $\hat{y}_i$ is the predicted value or hypothesis of the ith datapoint in the dataset of size N.

5.1.2. Mean Absolute Error (MAE)

MAE is a measure of errors between paired observations expressing the same phenomenon. Mathematically,

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{N}$$

5.1.3. Accuracy

Accuracy is the fraction of predictions the model gets right., and is formulated as

$$Accuracy = \frac{Number\ of\ correct\ classifications}{Total\ number\ of\ classifications}$$

It can also be expressed as,

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

where, TP = True Positive, FP = False Positive, TN = True Negative, FN = False Negative.

5.1.4. Precision

Precision is the fraction of relevant instances among the retrieved instances. It finds what proportion of positive identifications was actually correct. Mathematically,

$$Precision = \frac{TP}{TP + FP}$$

5.1.5. Recall

Recall is the fraction of relevant instances that were retrieved. It finds what proportion of actual positives were identified correctly. Mathematically,

$$Recall = \frac{TP}{TP + FN}$$

5.1.6. F1 score

F1 score is a function of Precision and Recall. It is also a measure of the test's accuracy. Mathematically,

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{(FP+FN)}{2}}$$

5.2. Results

5.2.1. Random Forests Classifier

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes become our model's prediction.

Estimators	Depth	RMSE	MAE	Train Accuracy	Test Accuracy	Precision	Recall	F1
100	10	0.5905	0.3487	0.6561	0.6513	0.6417	**0.8904**	0.7458
	20	0.5404	0.2920	0.7932	0.7080	0.7034	0.8503	0.7699
	25	0.5325	0.2836	0.8908	0.7164	0.7177	0.8350	0.7719
	28	0.5318	0.2829	0.9336	0.7171	0.7219	0.8258	0.7704
	30	0.5328	0.2839	0.9561	0.7161	0.7237	0.8184	0.7681
	35	0.5359	0.2871	**0.9851**	0.7129	**0.7239**	0.8086	0.7639
25	28	0.5389	0.2905	0.9228	0.7095	0.7185	0.8128	0.7628
50		0.5345	0.2856	0.9308	0.7144	0.7208	0.8211	0.7677
150		0.5310	0.2819	0.9347	0.7181	0.7224	0.8273	0.7713
200		0.5304	0.2813	0.9354	0.7187	0.7228	0.8281	0.7718
250		**0.5301**	**0.2810**	0.9361	**0.7190**	0.7230	0.8281	**0.7720**

Table 6: Performance of Random Forest Models

A random forest is a meta-estimator that uses averaging to improve the predictive accuracy and control over-fitting. Number of estimators is the number of decision trees in the forest and depth specifies the depth of each decision tree. It is observed that the performance the model improves as we increase the number of decision trees. However, after a while the improvement becomes too insignificant compared to the computational overhead. In our case, a random forest model with 250 estimators, each of which has a depth of 28 yields the best test accuracy of 71.9%.

5.2.2. Decision Tree

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences. Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Depth	RMSE	MAE	Train Accuracy	Test Accuracy	Precision	Recall	F1
5	0.6046	0.3656	0.6345	0.6344	0.6613	**0.7457**	0.7009
10	0.5833	0.3403	0.6633	0.6597	0.6928	0.7326	0.7122
15	**0.5765**	0.3324	0.7083	**0.6676**	0.6971	0.7453	0.7204
16	0.5769	**0.3328**	0.7205	0.6672	**0.6984**	0.7406	0.7189
17	0.5773	0.3332	0.7347	0.6668	0.6943	0.7503	**0.7213**
20	0.5832	0.3402	**0.7819**	0.6598	0.6930	0.7323	0.7121

Table 7: Performance of Decision Tree Models

The criterion parameter determines how the impurity of a split will be measured. Supported criterion include "gini" for the Gini impurity and "entropy" for the information gain. For our model, **entropy** outperforms "gini", hence, entropy is used. We use a depth of 15 in the decision tree instance that achieves the best test accuracy scores.

5.2.3. Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable.

Iterations	RMSE	MAE	Train Accuracy	Test Accuracy	Precision	Recall	F1
100	0.6050	0.3660	0.6340	0.6340	0.6471	**0.7986**	0.7149
500	**0.6039**	**0.3647**	**0.6354**	**0.6353**	**0.6488**	0.7965	**0.7151**

Table 8: Performance of Logistic Regression Models

We allow the logistic regression model to run for a specific number of iterations due to constraints on computational resources. Additionally, the overhead involved in the training the model for higher number of iterations does not yield any significant improvement.

5.2.4. Extra Trees (Extremely Randomized Trees Classifier)

Extra Trees Classifier is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a forest to output its classification result. It is similar to Random Forest Classifier, although differs in the manner of construction of the decision trees in the forest.

Estimators	Depth	RMSE	MAE	Train Accuracy	Test Accuracy	Precision	Recall	F1
50	25	0.5635	0.3175	0.7835	0.6825	0.6775	**0.8537**	0.7555
	30	0.5534	0.3063	0.8708	0.6937	0.6937	0.8362	0.7583
	35	0.5500	0.3025	0.9295	0.6975	0.7024	0.8216	0.7573
	40	0.5516	0.3042	**0.9715**	0.6958	**0.7085**	0.7995	0.7513
	34	0.5500	0.3025	0.9246	0.6975	0.7019	0.8233	0.7578
	36	0.5495	0.3020	0.9407	0.6980	0.7047	0.8168	0.7567
	37	0.5493	0.3017	0.9494	0.6983	0.7064	0.8125	0.7558
	38	0.5504	0.3030	0.9575	0.6970	0.7064	0.8089	0.7542
100	37	0.5476	0.2999	0.9498	0.7001	0.7075	0.8151	0.7575
150	37	0.5465	0.2987	0.9531	0.7013	0.7083	0.8164	0.7585
200	37	**0.5463**	**0.2985**	0.9533	**0.7015**	0.7082	0.8170	**0.7587**

Table 9: Performance of Extra Trees Models

5.2.5. Support Vector Classifier

The objective of the support vector machine is to find an optimal hyperplane in an N-dimensional space that distinctly classifies the data points. Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, the margin of the classifier is maximized.

C	Iters.	Kernel	RMSE	MAE	Train Accuracy	Test Accuracy	Precision	Recall	F1
1	500	poly	0.6546	0.4285	0.5724	0.5715	0.5738	0.9885	0.7261
1	1000	poly	0.6523	0.4255	0.5753	0.5745	0.5747	0.9972	0.7292
1	2000	poly	0.6532	0.4267	0.5746	0.5733	0.5744	0.9940	0.7281
1	1000	rbf	0.7034	0.4948	0.5068	0.5052	0.5659	0.5961	0.5806
1	1000	linear	0.6707	0.4498	0.5506	0.5502	0.5706	0.8777	0.6916
1	1000	sigmoid	**0.6500**	**0.4226**	**0.5785**	**0.5774**	0.5799	0.9601	0.7231
1	1000	sigmoid	0.7547	0.5696	0.4300	0.4304	**0.6295**	0.0210	0.0406
10	1000	sigmoid	0.6530	0.4264	0.5744	0.5736	0.5746	0.9937	0.7281
0.1	1000	sigmoid	0.7535	0.5677	0.4314	0.4323	0.6101	0.0330	0.0625
10	1000	poly	0.6523	0.4254	0.5752	0.5746	0.5748	0.9971	0.7292
0.1	1000	poly	0.6524	0.4256	0.5752	0.5744	0.5745	**0.9993**	**0.7296**
100	1000	poly	0.6549	0.4288	0.5720	0.5712	0.5737	0.9877	0.7258

Table 10: Performance of SVMs

C is the regularization parameter which is inversely proportional to the strength of regularization. Kernel specifies the kernel type to be used in the algorithm. Supported kernels are 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. In this experimentation, we have not used custom kernels. Among the supported types, polynomial kernel appears to yield comparatively better results, however the accuracy scores are not satisfactory and further experimentation is required. Moreover, due to computational constraints, the SVM instances were allowed to converge within a couple of thousand iterations.

5.3. Comparative Study

We compared our results with existing state-of-the-art solutions that try to solve the recommendation problem under similar circumstances. Su et. al. [19] evaluated the performance of their proposed recommender system on another MovieLens dataset [24], which, however, does not include user demographics. They compare their model with other baseline systems and present the performance with respect to two main metrics – RMSE and MAE (Table 11).

Methods	CF-SV	CF-Pearson	CF-Cosine	IOS	MLCF	IGPE	User-MRDC
MAE	**0.688**	0.703	0.704	0.700	0.705	0.735	0.697
RMSE	0.936	0.951	0.947	0.945	0.954	0.987	**0.932**

Table 11: Performance of State-of-the-art Models [cite]

Although our model was trained on a different MovieLens dataset and made use of user demographics, we consider comparing our results on the same metrics as our problem domain as well as overall dataset characteristics are not highly dissimilar than the approach proposed by Su et. al. In our experimentation, the model that yielded best test accuracy (random forest model) produce RMSE and MAE values of 0.5301 and 0.2810 respectively, which are considerably better than the ones documented in Table 11.

With the help of this result, we can infer that the improvement was due to the integration of user attributes in the computation of embeddings and further model training. The dataset Su et al. used share the same user-item interaction characteristic as our dataset and the only difference lies in the fact that they did not take into account user attributes while predicting the nature of future connections between users and movies.

6. Discussions

6.1. Limitations

6.1.1. Dataset

One of the primary motivations behind this project was to address the cold-start problem. The most effective resolution to the problem is to use entity attributes which are independent of the topological features and regardless of its extent of inclusion into the concerned network, we can derive some characteristics that will enable us to suggest better recommendations.

However, there is a noticeable scarcity of benchmark datasets which include user demographics. The MovieLens 1 million dataset is the only dataset we could find which matches our requirement and hence, we could evaluate our model on only one dataset. Since the dataset is comparatively large (with approximately 6000 users and 4000 movies), the user-item interaction matrix thus obtained is sparse and its repercussions on the model performance is unknown due to computational and time constraints.

6.1.2. Computational Aspects

Due to the size of the dataset, we had difficulties procuring the embeddings and training the model with the limited computational resources available to us. Consequences of longer training time of the models remain unknown. We could have performed a greater number of experiments if more computational resources were available.

6.2. Future Scope

6.2.1. Context awareness

We could have delved deeper into further information provided by The Movies Dataset such as movie production company, cast and crew, director etc. However, utilizing this information properly demands more context awareness. Moreover, as mentioned in section 4.2.2., determination the semantic distance between different values of categorical attributes, e.g., occupation, can be more effectively done using a context aware model.

6.2.2. Experimentation with other datasets and models

As open-source datasets with user demographics are not readily available, our solution approach could not be evaluated extensively. Other models, such as those employing deep learning approaches can be evaluated on these datasets may lead to deeper insights on how to modify low-level implementations to yield better results.

6.2.3. SVM Kernels

Due to resource constraints, extensive experimentation could not be performed on SVM models following our approach. SVMs are powerful classification models, especially with high-dimensional datapoints. Further studies can be conducted with SVMs, employing custom kernels and allowing flexible learning time.

6.2.4. Hyperparameters

The hyperparameters used in our implementation were conceived from an intuitive standpoint. Rigorous experimentation with these values might yield interesting results.

7. Conclusion

We have proposed a novel content-based approach to movie recommendation system exploiting the bipartite nature of the underlying network and by posing the problem as a binary classification. User attributes along with movie attributes can reveal insights on behavioral aspects of the users. Using the information gathered, it is possible to establish links between the nodes of each partite set (users and movies) which correspond to similarity measures between the nodes (user-user similarity and movie-movie similarity). Embeddings that combine basic attributes, behavioral features and features of highly similar entities were fed into a binary classifier that predicts a class label for the concerned user-movie pair. Following this approach, experimentation on the MovieLens 1 million dataset show promising results in comparison to content-based models that do not consider user-attributes, which further solves the cold-start problem.

References

[1] Liben-Nowell D, Kleinberg J M (2007), The link-prediction problem for social networks. Journal of the American Society for Information Science and Technology, 58(7): Pages 1019-1031

[2] Daud N N, Ab Hamid S H., Saadoon M, Sahran F (2020), Applications of Link Prediction in Social Networks: A Review. Journal of Network and Computer Applications, DOI: 10.1016/j.jnca.2020.102716.

[3] Adomavicius G, Alexander T (2005), Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE transactions on knowledge and data engineering, vol. 17, no. 6.

[4] Baptiste Rocca, Introduction to recommender systems, Towards Data Science.

[5] Basu C, Hirsh H, Cohen W (1998), Recommendation as Classification: Using Social and Content-Based Information in Recommendation. Recommender Systems. Papers from 1998 Workshop, Technical Report WS-98-08, AAAI Press.

[6] Lu J, Wu D, Mao M, Wang W, Zhang G (2015), Recommender system application developments: A survey, Elsevier, 2015.

[7] Susan G, Speretta M, Chandramouli A, Micarelli A (2007), User Profiles for Personalized Information Access, in The Adaptive Web: Methods and Strategies of Web Personalization, Springer, Berlin, Heidelberg, pp.54–89. DOI: 10.1007/978-3-540-72079-9_2.

[8] Pazzani M J, Billsus D (2007), Content-Based Recommendation Systems, in The Adaptive Web: Methods and Strategies of Web Personalization, Springer, Berlin, Heidelberg, pp. 325–341. DOI: 10.1007/978-3-540-72079-9_10.

[9] Cacheda F, Carneiro V, Fernández D, Formoso V (2011), Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems, ACM Trans. Web, vol. 5, no. 1, pp. 1–33. DOI: 10.1145/1921591.1921593.

[10] Goldberg K, Roeder T, Gupta D, Perkins C. Eigentaste (2001), A constant time collaborative filtering algorithm, in Inf. Retr., pp. 133–151.

[11] Chen J, Wang H, Yan Z (2018), Evolutionary heterogeneous clustering for rating prediction based on user collaborative filtering. Swarm Evol Comput.1:35–41.

[12] Melville P, Mooney R J, Nagarajan R (2002), Content-boosted collaborative filtering for improved recommendations, in Proc. of 18th National Conf. on Artif. Intell., Alberta, Canada, pp. 187–192.

[13] Burke R (2002), Hybrid recommender systems: Survey and experiments. User modelling and user-adapted interaction, 12(4):331-370.

[14] Kim H N, El-Saddik A, Jo G S (2011), Collaborative error-reflected models for cold-start recommender systems. Elsevier, Decision Support Syst.51(3):519–31.

[15] Pereira A L, Hruschka E R (2016), Simultaneous co-clustering and learning to address the cold start problem in recommender systems. Elsevier, Knowledge Based Syst.1:11–9.

[16] Deodhar M, Ghosh J. Scoal (2010), a framework for simultaneous co-clustering and learning from complex data, ACM Trans. Knowledge Discovery Data 4 11

[17] Lin C H, Chi H (2019), A novel movie recommendation system based on collaborative filtering and neural networks. In: International conference on advanced information networking and applications. Cham: Springer;p. 895–903.

[18] Walek B, Fojtik V (2020), A hybrid recommender system for recommending relevant movies using an expert system. Elsevier, Expert Systems with Application. 13:113452.

[19] Su Z, Zheng X, Ai J, Shen Y, Zhang X (2020), Link prediction in recommender systems based on vector similarity, Published in the journal Elsevier Physica A.

[20] Farashah M V, Etebarian A, Azmi R, Dastjerdi R E (2021), A hybrid recommender system based-on link prediction for movie baskets analysis, Published on Journal of Big Data, Springer.

[21] GroupLens – MovieLens 1 Million Dataset: https://grouplens.org/datasets/movielens/1m

[22] Banik R, Kaggle – The Movies Dataset: https://www.kaggle.com/rounakbanik/the-movies-dataset

[23] Wang P, Xu B W, Wu Y R, Zhou X Y (2014), Link Prediction in Social Networks: the State-of-the-Art, Science China. Information Sciences, November, DOI:10.1007/s11432-014-5237-y

[24] GroupLens – MovieLens 100K Dataset: https://grouplens.org/datasets/movielens/100k/