PROM02 Computing Master's Project

Assignment 2

Research Paper

Topic: Knowledge Graph Based Hybrid Movie Recommender System

Name: Chu Siu Kay Alan

Student No: 189222006

Word Count: 6102

References: 18

1 Abstract

Recommender systems are vital to online business nowadays. For online movie industry, it is desired to develop a movie recommender system that is able to predict what movies customers like to watch. According to the paper (Carlos A. Gomez-Uribe and NEIL H., 2015), 80% of movies watched on Netflix came from recommender system. A good movie recommender system is crucial for maintaining users' subscriptions. Most of the recommender systems nowadays employ multiple algorithms to enhance the accuracy. Many studies show that the content-based filtering and the collaborative filtering compliment each other very well. Besides accuracy, interpretable prediction result is also an important aspect of recommender systems. Several studies show that both content-based and collaborative filtering could be implemented using knowledge graphs, which produce interpretable recommendation results. (Amara S. and Subramanian R. R., 2020) This paper proposes a knowledge graph based hybrid movie recommender system (KGBHRS) which leverages both content-based and collaborative filtering techniques on a well defined knowledge graph representation of movie data. The evaluation of KGBHRS shows the system performs well in terms of accuracy. Besides that, KGBHRS produces recommendations that are interpretable and it can generate real-time recommendations based on the latest change of the rating data. The prototype of the proposed hybrid movie recommender system could be accessed through https://achu3080803.shinyapps.io/PROM02_Project/.

2 Introduction

In recent years, the increasing popularity of mobile devices have made the online businesses including online shopping, movie, music, reading platforms and social media sites grow tremendously. Users of these online services very often have to choose what they prefer from overwhelming information. Users may lose interest and patience if they could not find what they want on the online platform within certain period of time. Regarding to the online movie platforms, users could lose interests and drop the subscription if they often could not find any movies that they like. As a result, it is desired to develop an ideal recommender system that suggests movies which are interesting to the users. In order to improve both the accuracy and diversity of the recommendations, most of the recommender systems nowadays employ multiple approaches and techniques. Many studies show that the content-based filtering (CBF) and the collaborative filtering (CF) compliment each other very well. (Amara S. and Subramanian R. R., 2020) CBF recommends items to users based on the similarity between items and users' preference. CF recommends items to users by suggesting the preferred items of the others who are similar to the user. Sometimes CBF could recommend items that are too similar which makes the recommendation over-specialized. (Lu, J. et al., 2015) With the help of CF, recommendations on items that are different to the user's preferred items could be resulted. Hence diversity could be resulted which compensates the potential problem of over-specialized recommendations from CB. On the other hand, CF could produce poor recommendation when there is not enough movie ratings given by the users. In this case, the recommendations made by CB could alleviate this issue. Besides accuracy, interpretable prediction result is also an important aspect of recommender systems. This makes the system easier to maintain and enhance as it allows people to see how the recommendations are generated under the hood. Another important aspect of an ideal recommender system is that it should be able to generate real-time recommendations based on the latest change on the movie ratings. This paper proposes a knowledge graph based hybrid movie recommender system (KGBHRS) which leverages both content-based and collaborative filtering techniques on a well defined knowledge graph representation of movie data. In Section 3, this paper starts with introducing some preliminary concepts that help the user understand the proposed recommender system presented in the later sections. In Section 4, the related work from others are presented. In Section 5, implementation details of the proposed hybrid recommender systems are provided.

3 Preliminary

This section discusses the preliminary knowledge and concepts that help understanding the content presented in later sections of this paper.

3.1 Content Based Filtering

Content Based Filtering (CBF) is one of the most common techniques for implementing recommender systems. CBF generates recommendations based on users' previous behaviour or a pre-defined user profile for new users. The users' profiles basically contain the features of the users' preferred items. The users' profiles could be built by analysing their previous interactions with the items. However, a cold start problem may occur if the user is new as he or she does not have any previous interactions to analyse. In this case, the possible workaround is that the recommender system could allow the user to define his or her preferences as a user profile during account creation. Besides user profile, items also have their own profiles which are made up of the item features. During the recommendation generation, the CBF will try to find the items with features that are similar to the preferred features defined in the user profile and make the recommendation accordingly.

3.2 Collaborative Filtering

Collaborative Filtering (CF) is another popular technique for implementing recommender systems. CF is based on an idea that users with similar tastes in the past will have similar tastes in the future. (Pajuelo-Holguera, F. et al., 2020) For example, both User A and User B has rated movie M with a high rating. This implies that User A and User B have similar preferences. If User B likes another Movie N, CF will assume that User A likes Movie N also and make the recommendation accordingly.

3.3 Knowledge Graph

Knowledge graph (KG) is a graph representation of factual knowledge, which was named by Google and widely adopted for its simplicity of constructing and processing. (Zhang et al., 2020) In a knowledge graph, knowledge is depicted in terms of entities and their relationships.

Figure 3.1 Knowledge graph that depicts an Actor called Tom Cruise

Figure 3.1 shows a possible knowledge graph of an actor. In this knowledge graph, the orange nodes are movies and the purple node in the centre is an actor. The arrows represents the relationship between the movies and the actor. This knowledge graph can be interpreted as that Tom Cruise has acted in the 10 movies represented by the orange nodes.

3.4 Cipher Query Language

Cipher query language is a declarative language that allows for expressive and efficient data querying in a property graph. Cipher was invented by Andrés Taylor when he was working in Neo4j, Inc. in 2011. (Wikipedia contributors, 2021). It was designed to have similar expressive power of SQL which allows querying and updating data in graph databases. Figure 3.2 shows an example of a Cipher query that extracts a movie named “Forrest Gump” with its actors. One can refer to (Cipher Query Language, 2020) for a usage guide on Cipher Query Language.

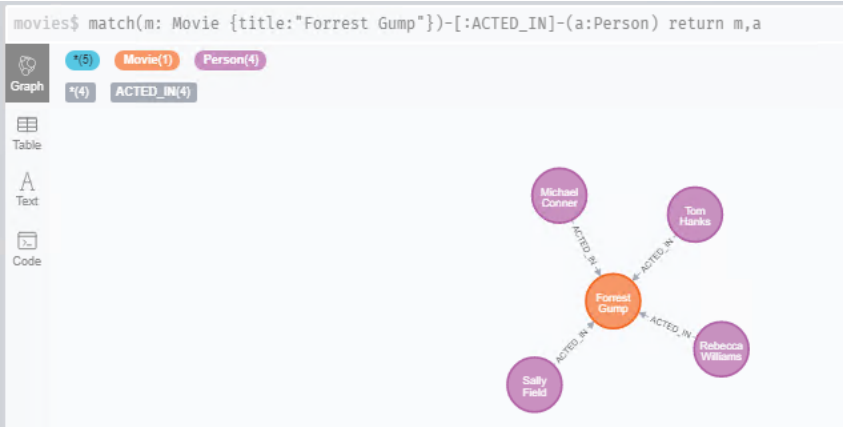


Figure 3.2 Example Cipher query to extract the movie “Forrest Gump” along with its actors

3.5 Similarity Functions

Similarity functions are used to calculate the similarity between two or more objects. They appear in recommender systems very often as most recommendation algorithms need to measure the similarity among different objects during the process of recommendation generation. Both Pearson’s Correlation and Jaccard Similarity are quite commonly used as similarity functions. They will be briefly discussed in the sub-sections below.

3.5.1 Pearson’s Correlation

Correlation is a technique for finding out the relationship between two quantitative, continuous variables, for example, age and blood pressure. Pearson’s correlation coefficient is a measure related to the strength and direction of a linear relationship. (Luthe Marvin, 2019) This metric for the vectors x and y can be calculated in the following way:

$$CORR(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \qquad \text{where} \qquad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

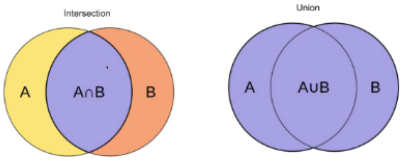
Figure 3.3 Pearson’s Correlation Formula

The Pearson Correlation has a range from -1 to 1. If the Pearson Correlation is 0, that means there is no relations between the vectors x and y. Any value towards 1 means x and y are positively correlated whereas any values towards -1 means x and y are negatively correlated.

3.5.2 Jaccard Similarity

Jaccard similarity can be used for comparing two binary vectors (sets). From Figure 3.4, we can see that the Jaccard similarity can be derived by dividing the size of the intersection by the size of the union of the sample sets. (Luthe Marvin, 2019)

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Figure 3.4 Jaccard Similarity Formula

4 Implementation Details

This section shows the implementation details of the proposed knowledge graph based hybrid recommender system (KGBHRS).

4.1 System Requirements

One of the major functions of KGBHRS is to generate accurate and diversified movie recommendations to customers in real-time based on the current online data. Besides generating movie recommendations, it should also allows staff to do some analysis on the movie data and allows the staff to look at how the recommendations are generated under the hood. Each specific functional requirement is captured as user story as shown in Table A.3 in Appendix A. Regarding to non-functional requirements such as performance, resilience and availability, please refer to Table A.4 in Appendix A for details.

4.2 Dataset

In this project, the movie dataset (ml-small-latest) from MovieLens (F. Maxwell Harper et al., 2015) was used to build the prototype of the KGBHRS. This dataset describes 5-star rating and free-text tagging activity from MovieLens. It contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018. Details of the dataset could be find in the table A.1 under Appendix A.

4.3 Data Pre-processing

The original dataset (ml-small-latest) from MovieLens does not have much details about the movies other than genres. Therefore pre-processing on the data was done to enrich the movie information by leveraging the OMDB API. For each movie, its actors, directors, production company, country, movie description and movie poster URL were loaded to the database from OMDB by calling the OMDB API as depicted by Figure 4.1.

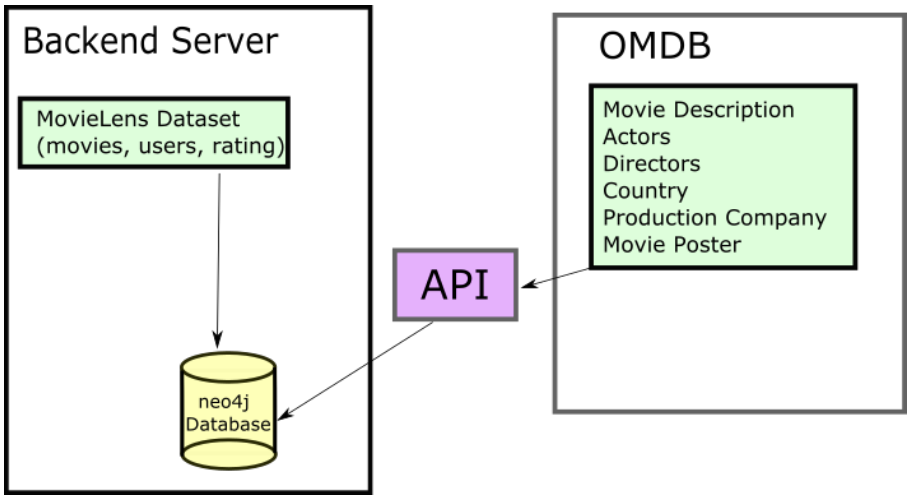


Figure 4.1 Data Pre-processing

4.4 System Design

This section discusses about the design of architecture design, knowledge graph schema and the user interface of the KGBHRS.

4.4.1 Architecture Design

This section discusses the architecture design of the movie recommender system KGBHRS. Figure 4.2 depicts the proposed architecture for the production movie recommender system.

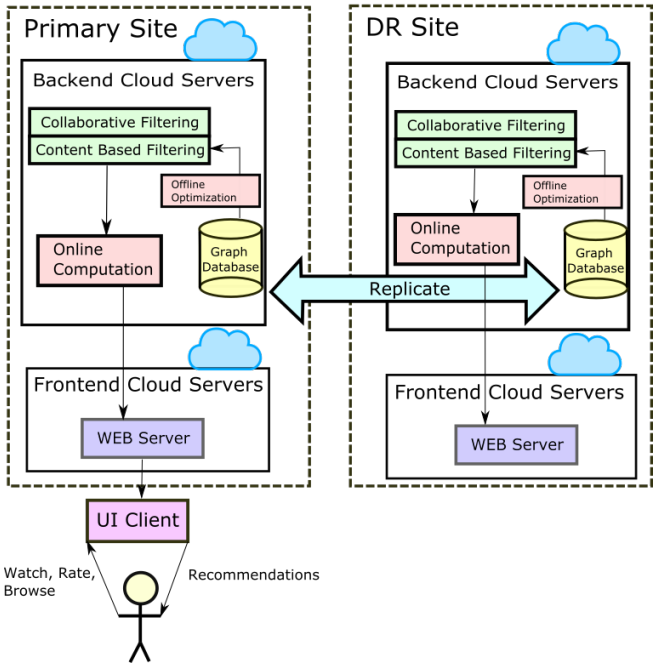


Figure 4.2 Proposed architecture of KGBHRS in Production

In order to satisfy the resilience and availability requirements, the proposed architecture design consists of a primary site and a disaster recovery (DR) site. The data is synchronised from primary to DR in real time to guarantee no data lost after failover. In the frontend, there is a WEB server to serve the WEB application to the customers. When the frontend servers need to obtain the recommendations, it will request the backend server to perform the online computation by applying both collaborative filtering and content based filtering to the graph database. Once the recommendations are generated, the result will be transferred back to the frontend server. The online computation module is also responsible for updating the graph database so that the graph data can be kept up to date all the time. For example, the customer may rate a new movie during the session. The online computation will re-compute the recommendation based on the updated data. Every night, the offline optimisation batch is run to optimise the hyper parameter of the proposed Content Based Filtering algorithm. This proposed design is inspired by the architecture design used in Netflix (Xavier A., 2019)

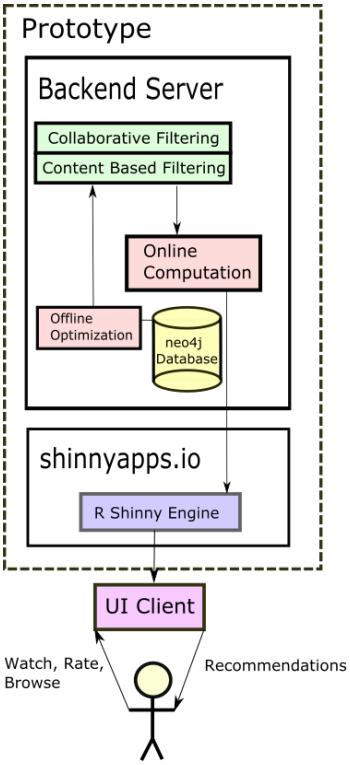


Figure 4.3 Proposed architecture of KGBHRS Prototype

As the prototype is to mainly demonstrate the functional requirements of the movie recommender system, the architecture of the prototype is simplified as shown in Figure 4.3. Since the MovieLens dataset does not include the movie poster images, a third party API provided by OMDB is used in the backend server to obtain the URLs retrieving the movie poster images. The URLs are then loaded to the Movie objects in the graph database. The R Shinny server (shinnyapps.io) connects directly to the backend server to access the neo4j graph database. Whenever users request for movie recommendations, the online computation will be invoked to generate recommendations in real time using both CBF and CF techniques. The offline optimisation is run very night to optimise the hyper parameter of the proposed content based filtering for each user.

4.4.2 Knowledge Graph Schema Design

A neo4j database is used as the backend database for storing the movie, user and review data in a graph representation. Figure 4.4 shows the meta graph of the database. The meta graph represents the labels and relationship-types defined in the database. From it, one could study how the labels are connected. Please refer to (Meta Graph, 2020) for more information about meta graphs.

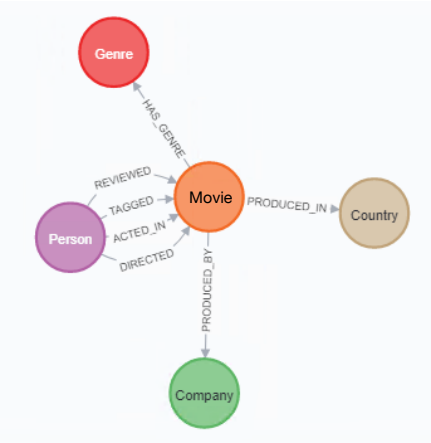
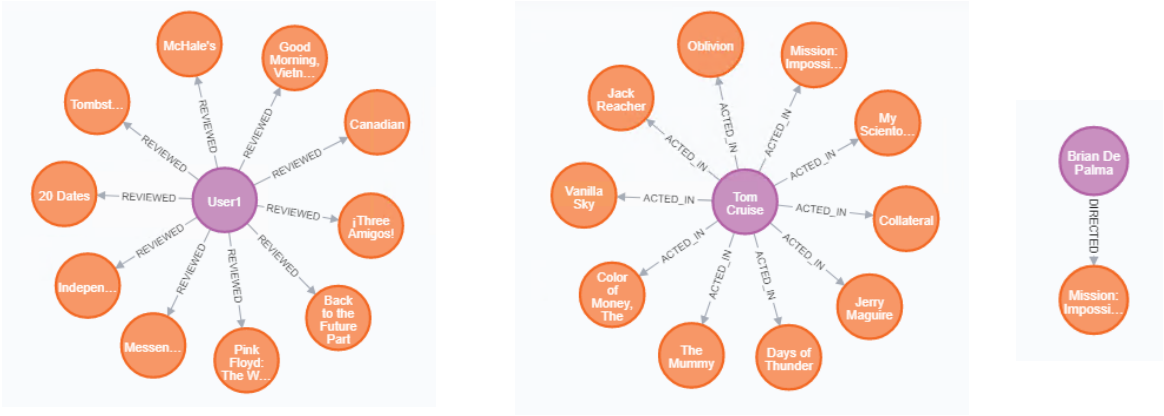


Figure 4.4 Meta graph

As shown in Figure 4.4, there are 5 labels defined. They are Movie, Person, Genre, Company and Country. The Movie label is for labelling nodes that contain movie properties such as movie title, production year, etc. The Genre label is for labelling nodes that represent movie genres. The purpose of genres is to categorise movies into different types such as action, comedy, drama, etc. The Country label mainly serves as indicating which country the movie is produced in. The Company label is for labelling nodes that contain information of the movie production companies. The Person label is used to label any nodes that represent persons. A person can have different relationships with the movie. He/she can be an actor, a director or a user. Figure 4.5 shows the different roles a person could be.



User Actor Director

Figure 4.5 Different relationships between persons and movies

For detailed definition of the nodes and relationships, please refer to Table A.2 in Appendix.

4.4.3 User Interface Design

The movie recommender system has four pages in total. The first page is the movie recommendation page which is accessible by customers. This page shows all the movies recommended by the system once the customers have logged in to the system.

In the movie recommendation page as shown in Figure 4.5, there are five rows of movies suggested. The first row displays the ten most popular movies. The second row displays the ten most recent movies rated by the users. The third row displays the top ten movies that are similar to the users' most favourite movies. This row of movies are actually recommended by using the proposed content-based recommendation algorithm KGCBF1. The fourth row displays the movies liked by the people who are similar to the user. This row of movies are recommended by using the proposed collaborative filtering algorithm KGCF1. The fifth row shows the movies that are acted by the actors or actresses that appear in the user's favourite movies. This row of movies are actually recommended by using the proposed content-based recommendation algorithm KGCBF2. Please refer to Section 4.5 for details of the algorithms. In the second row of recommendations, the five stars displayed underneath the movie posters indicate the rating the current user has given. However, the stars displayed underneath the movie posters in the other rows This row of movies are actually recommended by using the proposed content-based recommendation algorithm KGCBF1. indicate the average rating of the movies.

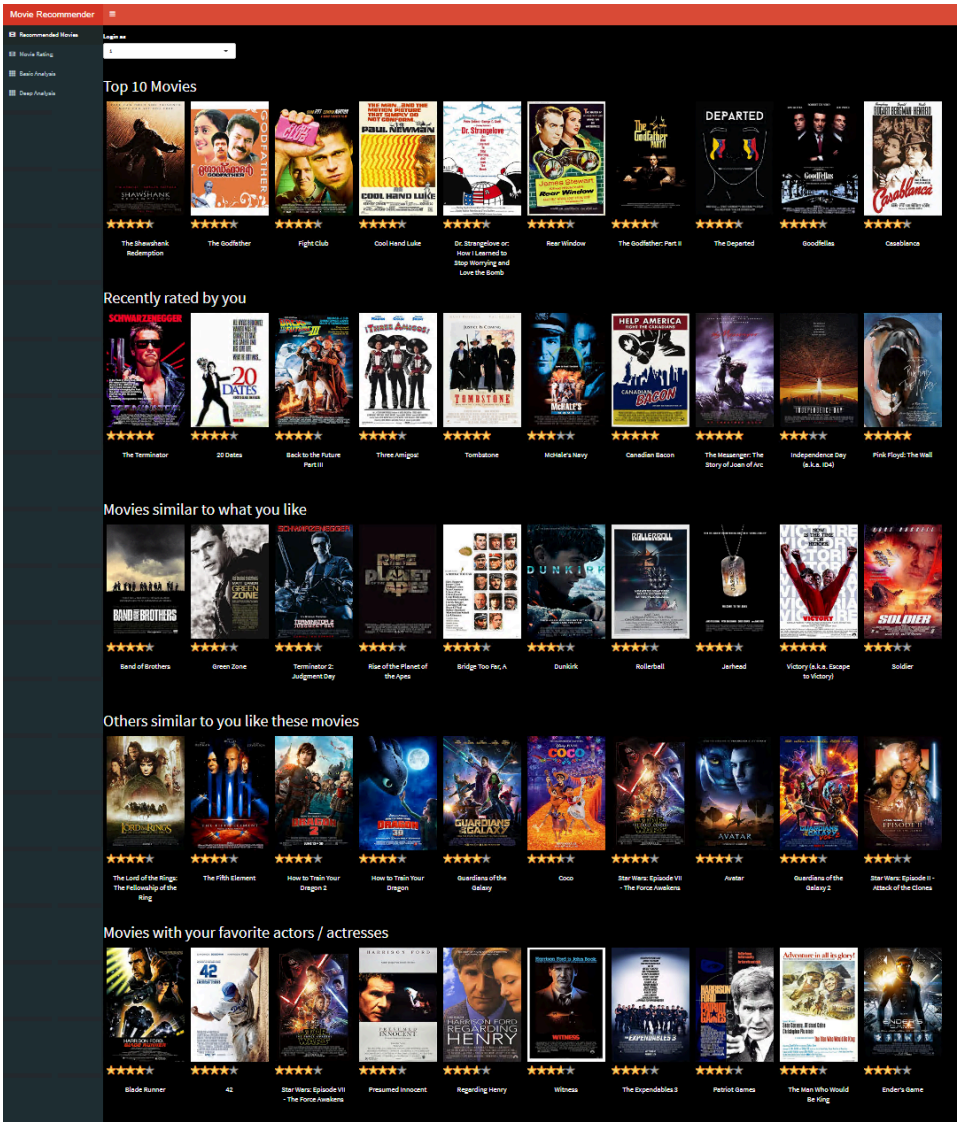


Figure 4.5 Movie Recommendation Page

The second page is the Movie Rating page which allows users to search a set of particular movies by movie name and allows the users to give star rating to one of the movies in the search result. Figure 4.6 shows the Movie Rating page.

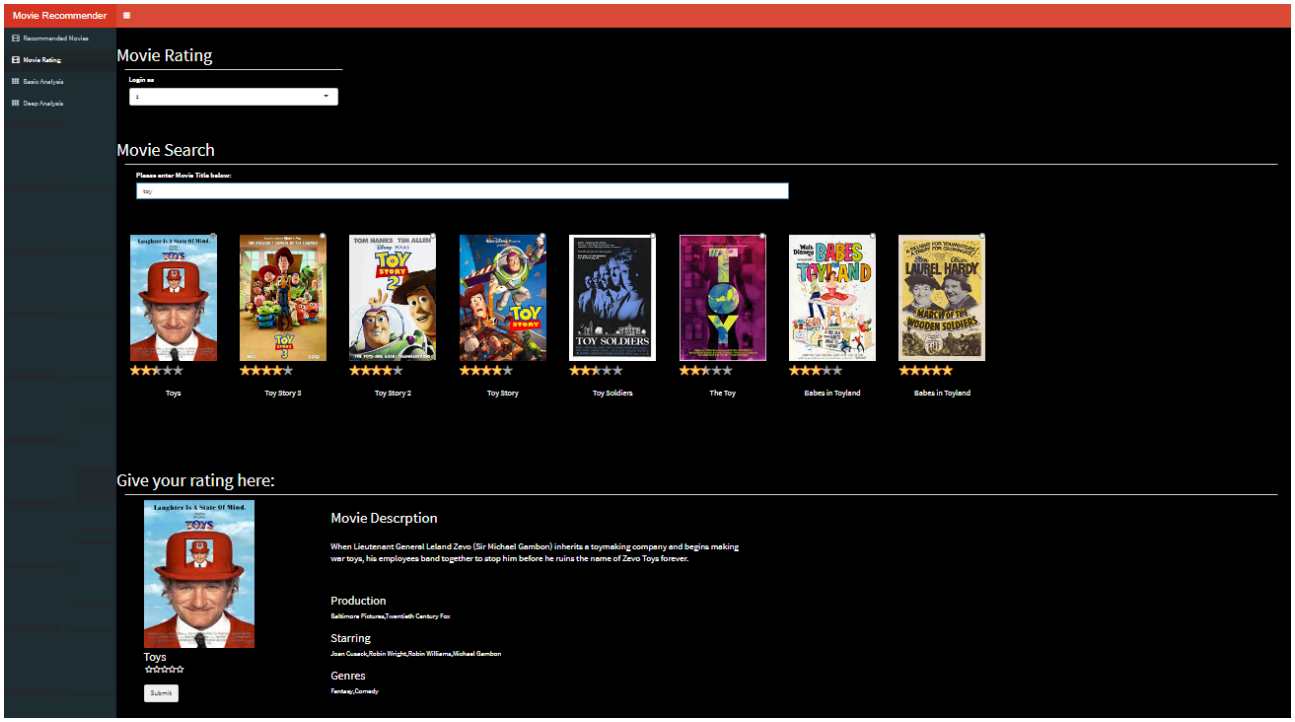


Figure 4.6 Movie Rating Page

The third page is a maintenance page which allows staff to perform some basic analysis on the movie data. The analysis may allow the staff to do the following:

1) Find out the top 10 most popular movies within a specific time period
2) Find out the most popular or least popular movie tags and genres within a certain time period
3) Find out the movie rating histogram
4) Find out the average number of ratings given by users for each year

Figure 4.7 shows the Basic Analysis page.

Figure 4.7 Basic Analysis Page

The last page is the Deep Analysis page which allows staff to pinpoint a particular movie recommendation and examine the reasoning behind it. Figure 4.8 shows the layout of the Deep Analysis page. The first drop down box allows a staff to choose a particular user. The second drop down box allows a staff to choose a particular movie recommendation category for conducting analysis. The movie recommendation categories include the following:

1. Movies similar to what you like
2. Others similar to you like these movies
3. Movie with your favourite actors / actresses

Below the drop down boxes, the top ten movies that the chosen user like most are displayed. Once the staff has chosen the values from the two drop down boxes, the system will display the top ten movies that satisfy the chosen criteria. As shown in Figure 4.8, the category “Movies similar to what you like” is chosen and the top ten movies that are similar to user’s favourite movies are displayed. Finally, the bottom section displays the graph that explains the reasoning behind the recommendation.

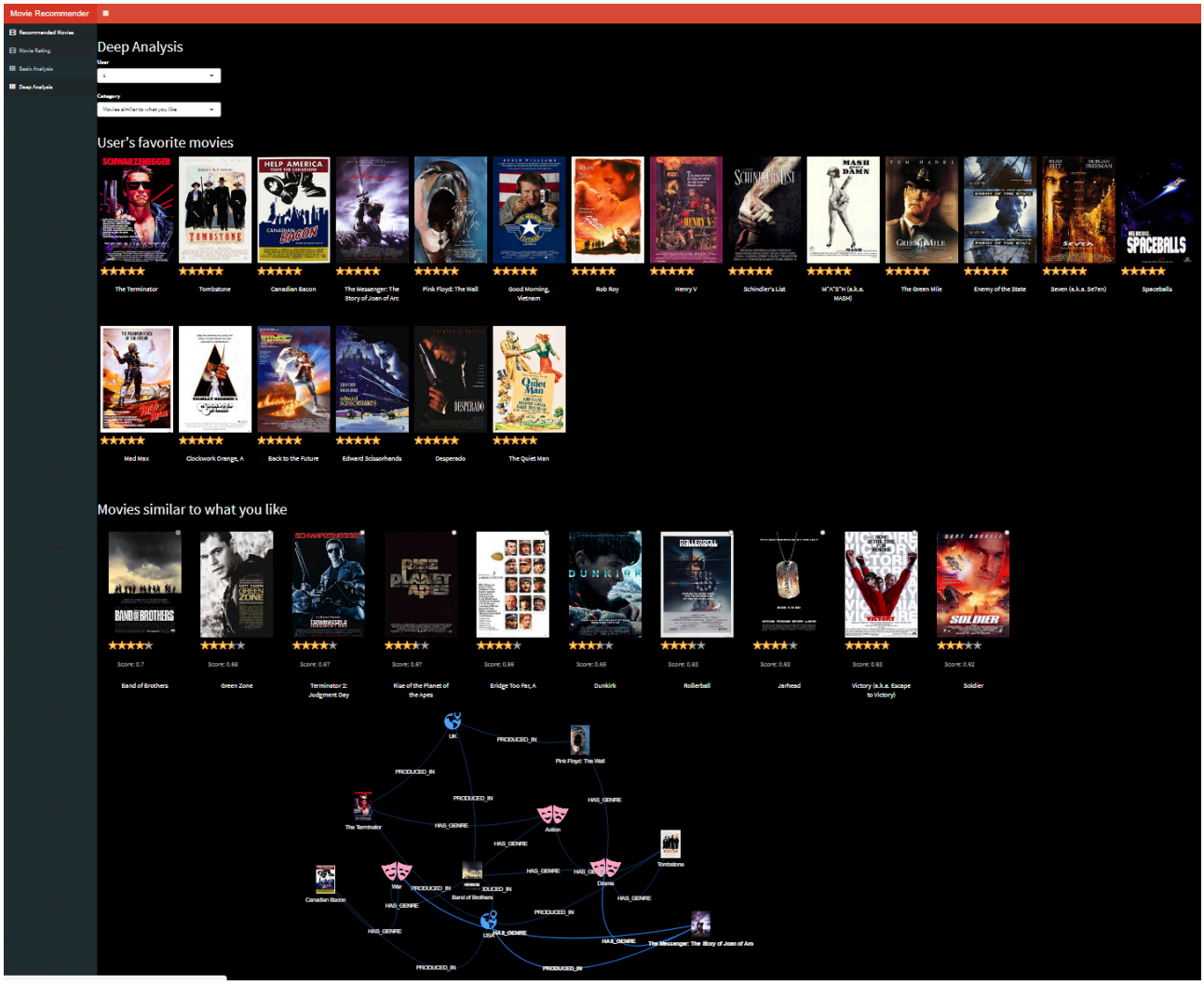


Figure 4.8 Deep Analysis Page

4.5 Proposed Algorithms

This section proposed the recommendation algorithms used in the KGBHRS. As shown in Section 4.3.3, there are five rows of movie recommendations in the Movie Recommendation page. The first two rows do not involve any recommendation algorithms. This sub-sections discuss each of the recommendation algorithms used in the third, fourth and the fifth rows of the recommendations.

4.5.1 Algorithm 1 (KGCBF1)

In a traditional content-based filtering, user profile containing the features of user's preferred items needs to be built and updated. The user profile is then used to compare all the available items for determining the recommendation result. In this approach, extra processing is needed to analyse the movies that are recently rated to keep the user profile up to date. Instead of based on a user profile, the recommendation generation should be based on the top N favourite movies most recently rated. This is not only simpler but it also allows the user's latest preferences to have greater impact on the recommendations. Hence, the recommender system may adapt better to the changes on user's preference. The recommender system tries to find the top ten movies that are most similar to the N most recent favourite movies. The similarity between the recommended movies and the user's favourite movies is calculated by using the Jaccard Similarity algorithm based on the common features such as actors, directors, genres, production company and country. This algorithm is a content-based filtering algorithm implemented by using a knowledge graph. Therefore this algorithm is named as KGCBF1. KGCBF1 is used to retrieve the third row (Movies similar to what you like) of movie recommendations in the Movie Recommendation page. Since the optimal value of N may differ from user to user, an offline optimization process is designed to find out the optimal value of N for each user.

4.5.2 Algorithm 2 (KGCF1)

KGCF1 is a collaborative filtering algorithm implemented using knowledge graph. This algorithm is used to generate recommendations in the fourth row (Other similar to you like these movies) of the recommendation page. In this algorithm, the first step is to find which other users are similar to the target user by analysing the ratings given in the past. The users are considered as similar if they have rated the same set of movies with similar ratings in the past. The similarity between the target user and the others are calculated using the Pearson Correlation formula based on the ratings given on the common movies that they have rated. The next step is to retrieve the favorite movies of the users who are similar to the target user. Then for each candidate movie, a score is calculated by multiplying the Pearson Correlation coefficient with the rating given by the other users. Finally, the top ten of movies with highest scores are returned as the recommendation result.

4.5.3 Algorithm 3 (KGCBF2)

Similar to KGCBF1, KGCBF2 is another content-based filtering algorithm. However, instead of using users' most recent favourite movies, it uses users' favourite actors / actresses as a base to generate the movie recommendations. The first step is to find out the ten actors / actresses that appear the most in all the movies rated in the past by the user. A score is then calculated for each actor / actress based on the number of appearance. In other words, the actors / actresses with higher appearance will have a higher score. The next step is to find the unwatched movies that are acted by the actors / actresses and a score of each unwatched movie is then calculated by summing up the score of the actors / actresses. Hence, the more favourite actors / actresses the movie has, the higher score the movie will have. Finally, the list of unwatched movies will be sorted by their score in descending order and the sorted list of the unwatched movies will be return as the fifth row (Movies with your favorite actors / actresses) of recommendations in the Movie Recommendation page.

4.6 Software and Hardware

Both software and hardware used in this project are summarised as shown in Table 4.1 below.

Software		**Hardware / Platform**	
Frontend Layer	R Shinny	**Frontend Platform**	shinnyapps.io
Programming Languages	R, Cipher	**Backend Server**	Intel(R) Core™ i7-2600 CPU @ 3.40GHz
Database	Neo4j		

Table 4.1 Hardware and Software

Due to the limited timeframe for delivering a prototype on a WEB platform, R shinny was chosen as the tool for developing the movie recommender prototype. R Shinny delivers the benefits shown as below:

1. It allows users to develop both machine learning models and the WEB interface on the R language platform. Developers are not required to know other programming languages.
2. Users can host the WEB application for free in “www.shinnyapp.io".
3. There are many various ready-to-use WEB widgets for developers to build the user interface quickly.

Regarding to the backend database, Neo4j is a good candidate because it is the leader of graph databases. Most importantly there is a R package that allows the integration between Neo4j and the R applications.

5 Evaluation

5.1 A/B Testing

In an online movie company such as Netflix, A/B testing is often used to evaluate a new recommender algorithm. A/B testing is the process of comparing a baseline algorithm A to a new algorithm B. According to (Carlos A. Gomez-Uribe and NEIL H., 2015), in the A/B tests done in Netflix, different members are randomly assigned to different experiences that they refer to as cells. For example, each cell in an A/B test could map to a different recommendation algorithm, one of which reflects the default algorithm to serve as the control cell in the experiment—other cells in the test are the test cells. They then let the members in each cell interact with the product for two to six months. Finally, the resulting data are analyzed to answer questions below:

1. Are members finding the part of the product that was changed relative to the control more useful?
2. Are members in a test cell streaming more on Netflix than in the control?
3. Are members in a test cell retaining their Netflix subscription more than members in the control?

The answers to the questions above will determine whether the new recommendation algorithm wins or not.

5.2 Cumulative Gain and Discounted Cumulative Gain

A/B testing may not be feasible sometimes as it requires a fair amount of users’ participation and a long time to conduct. Therefore an offline test may be the only option to consider when evaluating the recommendation algorithms. Cumulative Gain (CG) is one of the most common metrics for evaluating recommendations which are in the form of N top ranked items. CG is the sum of the graded relevance values of all results in a search result list. The CG at a particular rank position p is expressed as shown in Figure 5.1.

$$CG_p = \sum_{i=1}^{p} rel_i$$

Figure 5.1 Cumulative Gain Mathematical Expression

CG does not take the rank of an item in the result list into the consideration of the usefulness of a result set. In other words, the ranking of the relevant item does not affect its CG value. On the other hand, Discounted Cumulative Gain (DCG) penalises the relevant item that appears in lower ranks. In other words, the relevant item that appears in a higher rank will have a higher DCG value than the one in the lower rank. The DCG at a particular rank position p is expressed as shown in Figure 5.2

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

Figure 5.2 Discounted Cumulative Gain Mathematical Expression

5.3 Evaluation of the KGBHRS

5.3.1 Baseline Algorithms

Baseline CBF

This is a standard content-based filtering algorithm implemented by using vector manipulation. For each user, Vector A is built to contain user's favorite movie genres. Then a vector of genres is built for each movie that describe which genres the movie belong to. The similarity between a user profile and a movie is then calculated the Jaccard similarity. The algorithm will recommend the top N movies that are most similar to the user profile.

Baseline CF

This is a collaborative filtering algorithm implemented by using the famous ALS Implicit technique.

5.3.2 Evaluation of Algorithm Performance

As A/B testing requires a fair amount of users' participation which is not possible for this project at this stage, offline evaluation using CG and DCG becomes a viable alternative for evaluating the proposed recommendation algorithms, KGCBF1, KBCF1 and KGCBF2. A baseline CBF and a baseline CF are also evaluated so that comparison could be made between the proposed algorithms and the baseline algorithms.

In the evaluation process, the first step is to sort the ratings by the timestamp in ascending order for each user. Then for each user, the ratings are divided into two halves and the second half of the ratings are masked. After that, each candidate algorithm is executed with only the first half of the ratings as input to generate the top ten movie recommendations for each user. After that, the generated recommendations is verified by using the second half of the ratings to determine whether the recommended movies are relevant. If the recommended movie appears in then second half of the ratings, the movie is considered as relevant. For the relevant movies, the rel_i is assigned with 1. The CG and DCG are then derived from the rel_i for each movie in the recommendation result. Finally the sum of CG and DCG for all movies for all users will be

collected as the final score of each algorithm. Figure 5.3 shows an example of the evaluation process of KGCBF1.

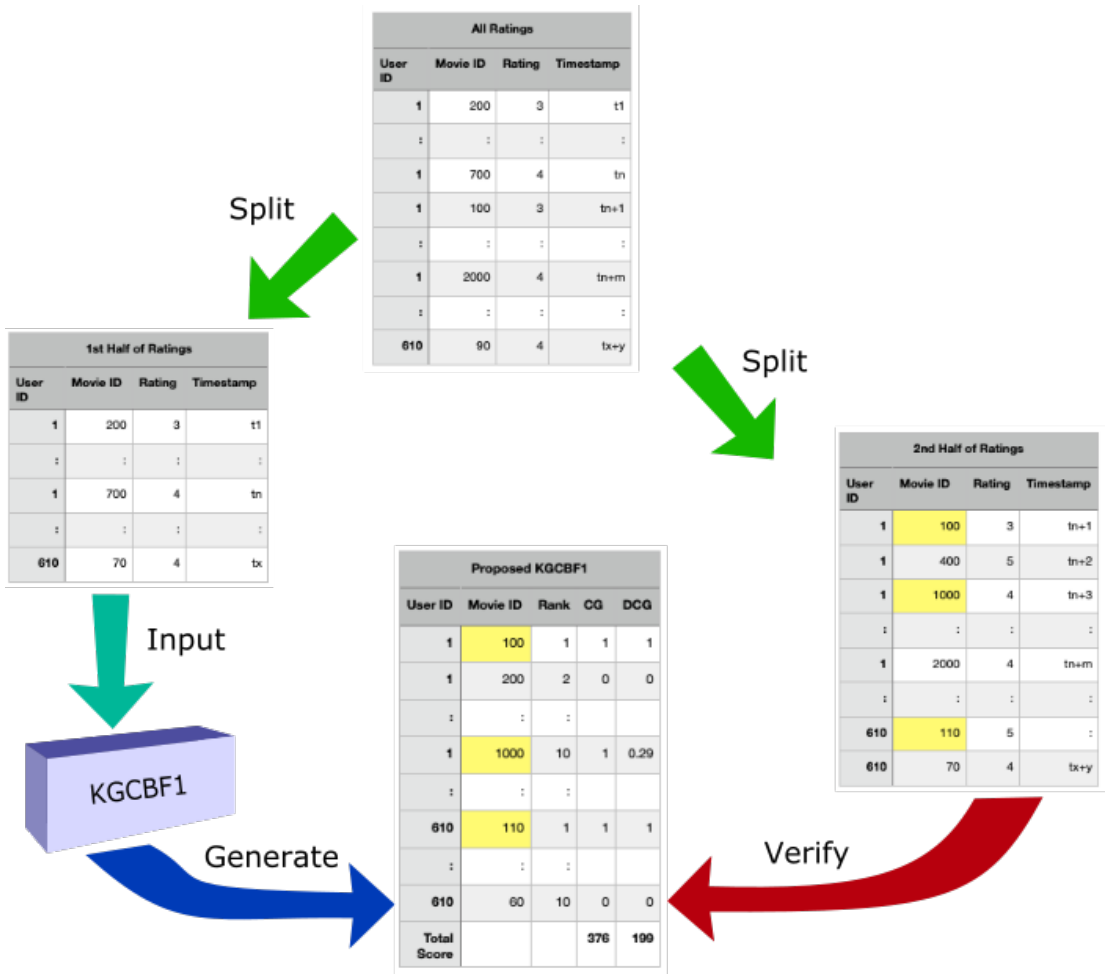


Figure 5.3 Evaluation Process

Table 5.1 shows the evaluation summary of both baseline and proposed algorithms. As shown in Table 5.1, both KGCBF1 and KBCBF2 outperform the baseline CBF. The proposed KGCF1 also outperforms the baseline CF.

Evaluation Summary		
	CG	**DCG**
Baseline CBF	79	43
Baseline CF	439	203
Proposed KGCBF1	376	199
Proposed KGCF1	618	285
Proposed KGCBF2	414	190

Table 5.1 Algorithm Evaluation Summary

5.3.2 Evaluation of Interpretability

Besides recommendation accuracy, building a recommender system that returns highly interpretable recommendation results is another important goal of this project. Since there is no objective way to evaluate the interpretability of the recommendation results, this section will explain how the recommendation results can be interpreted in the system and let the reader decide.

In the KGBHRS, there is a page called Deep Analysis which allows staff to analyze a particular recommendation of a particular user. Once a target movie is selected to be analyzed, a portion of knowledge graph will be displayed at the bottom of the page. This little knowledge graph allows the staff to see the reasoning behind the recommendations as the graph shows how the target movie is related to the objects and factors that are considered during the recommendation generation for all three proposed algorithms. Each of the example below explains how the recommendation results of each algorithm could be analyzed.

Example 1 - Analyzing KGCBF1 result

The staff would like to analyze why the movie “Terminator 2: Judgement Day” is recommended to User 1 under the category “Movie similar to what you like”. To start the analysis, the staff first needs to navigate to the Deep Analysis page. Then he needs to choose the login ID as 1 and choose the category as “Movie similar to what you like”. After that, the top 10 movies that are similar to what User 1 likes will appear in the middle of the page. Then the staff needs to select the movie “Terminator 2: Judgement Day”. After that, a graph as shown in Figure 5.4 that explains the reasoning behind the recommendation will appear at the bottom of the page. From the graph, the staff can find out the starting point of the reasoning is from the four favorite movies highlighted in the red box. Besides that, the staff could find out why the system thinks the target movie highlighted in the green box is similar to the four favorite movies by looking at the linkages between them.

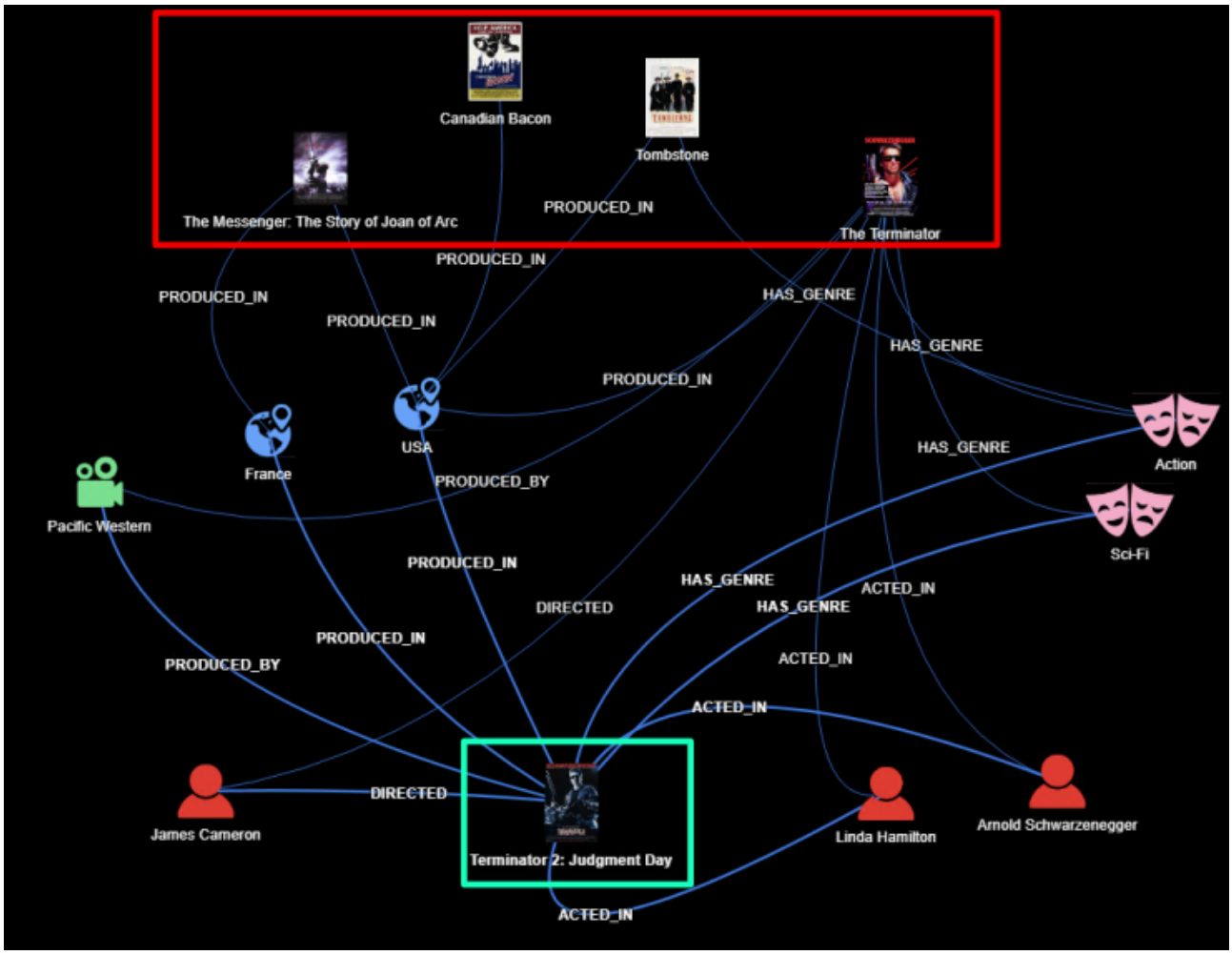


Figure 5.4 Knowledge graph that explains the reasoning behind the recommended movie "Terminator 2: Judgement Day"

Example 2 - Analyzing KGCF1 result

The staff would like to analyze why the movie "The Fifth Element" is recommended to User 1 under the category "Others similar to you like these movies". The staff can start the analysis by selecting login ID as 1 and category as "Others similar to you like these movies" in the Deep Analysis page. Then the top 10 recommended movies will appear in the middle of the page. After the staff has selected the movie "The Fifth Element", the graph as shown in Figure 5.5 will appear at the bottom of the page. From the graph, the staff can find out that User 210 is the starting point of reasoning as the system thinks User 210 is very similar to User 1. The graph also explains why the system thinks User 210 is similar to User 1 by showing all the common movies that they both rated in the past. At last, the graph shows that the recommended movie "The Fifth Element" is a favorite movie of User 210. This is why "The Fifth Element" is recommended to User 1.

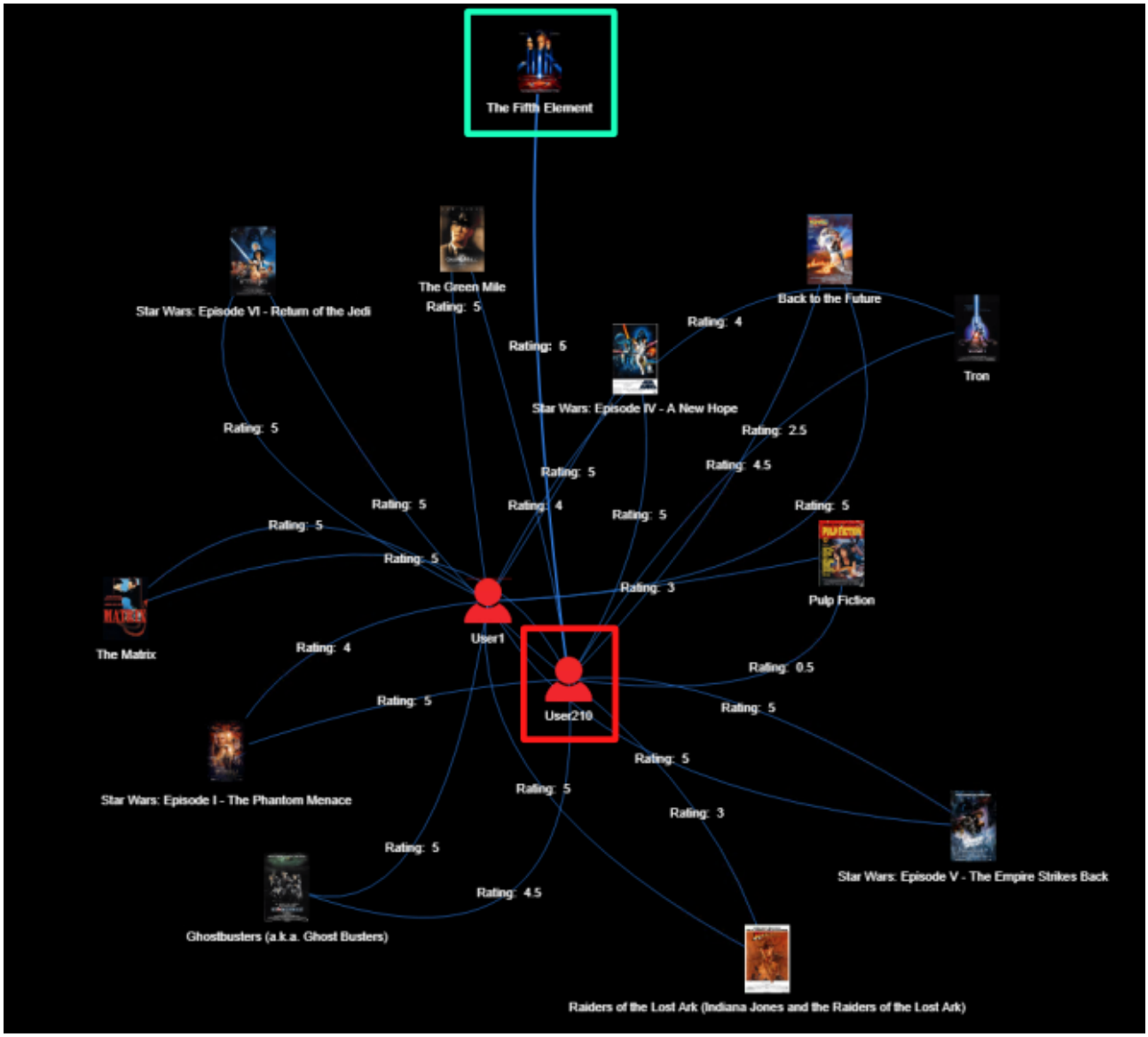


Figure 5.5 Knowledge graph that explains the reasoning behind the recommended movie "The Fifth Element"

Example 3 - Analyzing KGCBF2 result

The staff would like to analyze why the movie “Blade Runner” is recommended to User 1 under the category “Movies with your favorite actors/actresses”. The staff can start the analysis by selecting the login ID as 1 and the category as “Movie similar to what you like” in the Deep Analysis page. Then the top 10 movies that are acted by the favorite actors / actresses of User 1 will appear in the middle section of the page. Then the staff can select the movie “Blade Runner” to generate a graph as shown in Figure 5.6 that explains the reasoning behind the recommendation. From the graph, the staff can find out that “Blade Runner” is recommended because of the actor Harrison Ford, who is User 1’s favorite actor. The system thinks Harrison Ford is the favorite actor because he appears in all the watched movies that appear in the graph.

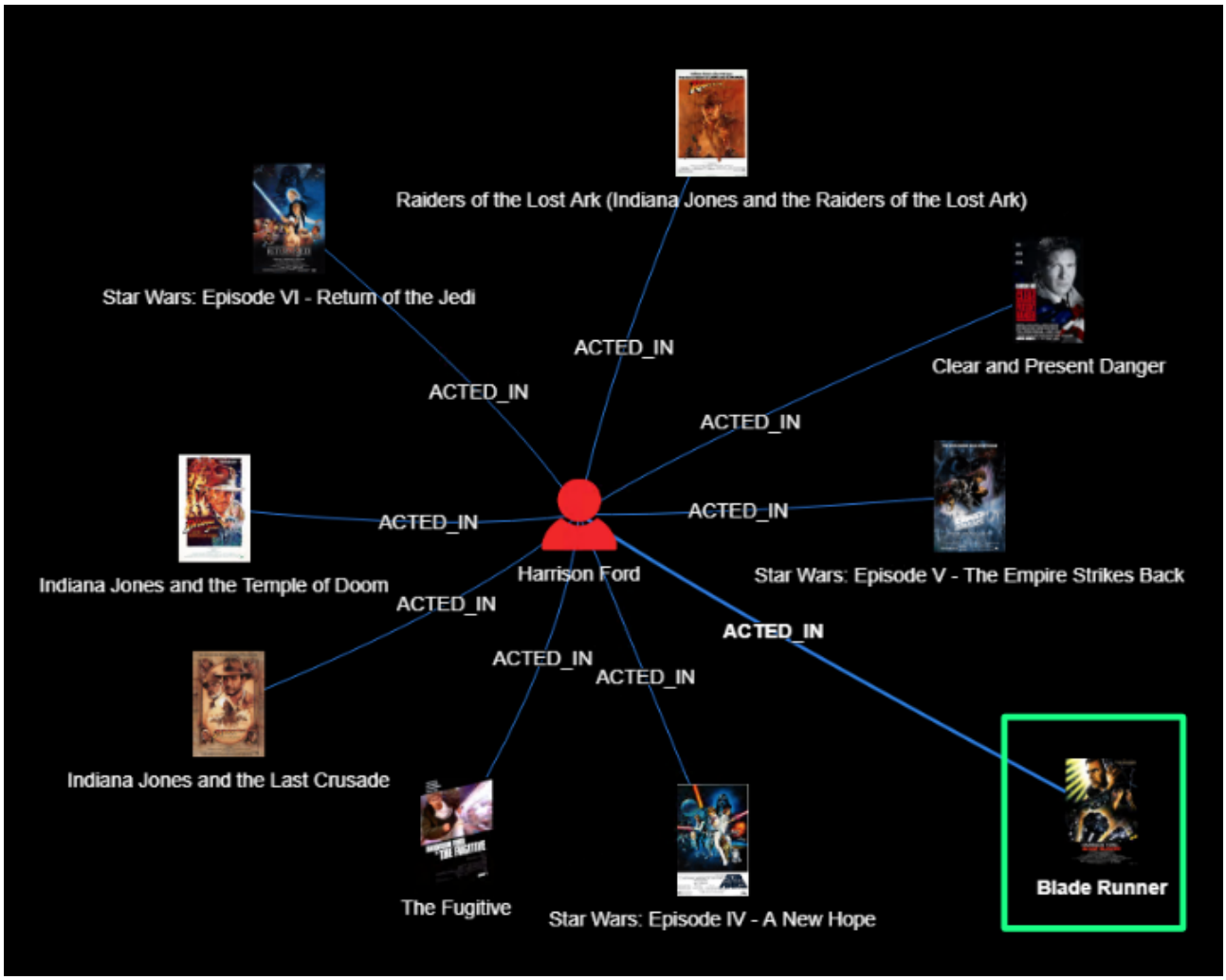


Figure 5.6 Knowledge graph that explains the reasoning behind the recommended movie “Blade Runner”

6 Future Work

Although the evaluation result seems satisfying, some further improvement could be made to the KGBHRS. In recent years, knowledge graph embedding becomes a hot topic. Knowledge graph embedding can be used to predict the missing links between the entities in the graph such that the knowledge graph will become more complete. This is especially useful when the user-item interactions are very sparse. According to (Zhang et al., 2020), there are three types of knowledge graph embedding. The first type is the translation-based models represented by TransE, TransH and TransR. The second type is the deep learning based models represented by SME, MLT and NTN. The third type is semantic similarity based models such as DistMult. In (Zhang et al., 2020), it proposes the KGECF model and implements a collaborative filtering recommender system using this model. The result looks promising. Hence, applying knowledge graph embedding to the KGBHRS could be one of the possible future work items.

7 Conclusion

The goal of this project is to develop a knowledge graph based hybrid movie recommender system which produces recommendations that are both relevant and interpretable. This paper proposes the movie recommender system KGBHRS which applies the proposed CBF and CF algorithms to make recommendations to users. According to the evaluation, the individual proposed CBF and CF algorithms outperform the baseline CBF and CF respectively. Also, the recommendation results are demonstrated to be interpretable through a graph representation. As a result, KGBHRS doubtlessly achieves the goal of this project. Applying the knowledge graph embedding technique to KGBHRS would be the future direction of this project.

8 References

Amara, S. and Subramanian, R. R. (2020) ‘Collaborating personalized recommender system and content-based recommender system using TextCorpus’, 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Advanced Computing and Communication Systems (ICACCS), 2020 6th International Conference on, pp. 105–109. doi: 10.1109/ICACCS48705.2020.9074360.

Carlos A. Gomez-Uribe and NEIL H. (2015) ‘The Netflix Recommender System: Algorithms, Business Value, and Innovation’, Netflix, Inc.

Cipher Query Language, (2020) Neo4j Developer Guide. <https://neo4j.com/developer/cypher/> (accessed March 28, 2021).

F. Maxwell Harper and Joseph A. Konstan. (2015) The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. <https://doi.org/10.1145/2827872>

Hahsler M., J. et al. (2020) ‘recommenderlab: A Framework for Developing and Testing Recommendation Algorithms’, Southern Methodist University Available at:https://cran.r-project.org/web/packages/recommenderlab/vignettes/recommenderlab.pdf.

Jeffrey S. S., Ivan S., Kevin C. (2017) ’Comparing Data Science Project Management Methodologies via a Controlled Experiment’, Proceedings of the 50th Hawaii International Conference on System Sciences | 2017

Lin Y., Liu Z., Sun M., Liu Y, Zhu X. (2015) ‘Learning Entity and Relation Embeddings for Knowledge Graph Completion’, Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence | 2015

Lu, J. et al. (2015) ‘Recommender system application developments: A survey’, Decision Support Systems, 74, pp. 12–32. doi: 10.1016/j.dss.2015.03.008.

Luthe Marvin (2019) *Calculate Similarity — the most relevant Metrics in a Nutshell,* Towards Data Science. <https://towardsdatascience.com/calculate-similarity-the-most-relevant-metrics-in-a-nutshell-9a43564f533e> (accessed March 29, 2021).

Meta Graph, (2020) APOC User Guide 4.1. <https://neo4j.com/labs/apoc/4.1/database-introspection/meta/> (accessed March 28, 2021).

Pajuelo-Holguera, F. et al. (2020) ‘Recommender system implementations for embedded collaborative filtering applications’, Microprocessors and Microsystems, 73. doi: 10.1016/j.micpro.2020.102997.

Mu R. and Zeng X. (2018) ‘Collaborative Filtering Recommendation Algorithm Based on Knowledge Graph’, Mathematical Problems in Engineering Volume 2018, Hindawi, Article ID 9617410.

Lullya V., Laubleta P., Stankovica M., Radulovicb F. (2018) ‘Enhancing explanations in recommender systems with knowledge graphs’, SEMANTiCS 2018 – 14th International Conference on Semantic Systems, Procedia Computer Science 137 (2018) 211–222

Walek, B. and Fojtik, V. (2020) ‘A hybrid recommender system for recommending relevant movies using an expert system’, Expert Systems With Applications, 158. doi: 10.1016/j.eswa.2020.113452.

Wikipedia contributors. (2021) "Knowledge graph," Wikipedia, The Free Encyclopedia, Available at:https://en.wikipedia.org/w/index.php?title=Knowledge_graph&oldid=1013661080 (accessed March 27, 2021).

Wikipedia contributors. (2021) “Cypher (query language),” Wikipedia, The Free Encyclopedia. Available at: https://en.wikipedia.org/w/index.php?title=Cypher_(query_language)&oldid=1003275913 (accessed March 28, 2021).

Xavier A. (2019) ’Big & Personal: data and models behind Netflix recommendations’. BigMine '13: Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications August 2013 Pages 1–6 Available at: https://doi.org/10.1145/2501221.2501222

Zhang, Y., Wang, J. and Luo, J. (2020) ‘Knowledge Graph Embedding Based Collaborative Filtering’, IEEE Access, Access, IEEE, 8, pp. 134553–134562. doi: 10.1109/ACCESS.2020.3011105.

Appendix A

File name	Column	Column Description
movies.csv	movieId	Identifier of a movie used in MovieLens
	title	Title of a movie
	genres	A list of genres of a movie in which genres are delimited by the character "|"
links.csv	movieId	Identifier of a movie used in MovieLens
	imdbId	Identifier of a movie used in IMDB
	tmdbId	Identifier of a movie used in TMDB
ratings.csv	userId	User ID of MovieLens
	movieId	Identifier of a movie used in MovieLens
	rating	Rating given by a user on a movie
	timestamp	The time when the rating was given by the user
tags.csv	userId	User ID of MovieLens
	movieId	Identifier of a movie used in MovieLens
	tag	Tag given by a user to associated with a movie
	timestamp	The time when the tag was given by the user

Table A.1 MovieLens Dataset

Nodes	
Movie	
<id>	neo4j internal ID
movieId	Movie ID to uniquely identify a movie
title	Title of a movie
year	Production year of a movie
poster	Poster URL
imdbId	Movie ID in IMDb (for linkage to IMDb)
tmdbId	Movie ID in TMDb (for linkage to TMDb)
Genre	
<id>	neo4j internal ID
name	Name of the person
Company	
<id>	neo4j internal ID
name	Company name
Country	
<id>	neo4j internal ID
name	Country name
Person	
<id>	neo4j internal ID
name	Name of the person
loginId	Movie Recommendation System Login ID of the person

Relationships	
REVIEWED	
<id>	neo4j internal ID
rating	Movie rating given by a user
review_date	Date of the movie review
timestamp	UTC timestamp of the movie review
ACTED_IN	
<id>	neo4j internal ID
DIRECTED	
<id>	neo4j internal ID
HAS_GENRE	
<id>	neo4j internal ID
PRODUCED_BY	
<id>	neo4j internal ID
PRODUCED_IN	
<id>	neo4j internal ID

Table A.2 Attributes of nodes and relationships

User Story ID	As A <Type of user>	I want to <perform some task>	So that I can <achieve some goal>
C1	As a customer	I want to be given a set of recommendations based on the ratings that I have given in the past.	So that I can select the movies that I like without much navigation.
C2	As a customer	I want to be given a set of recommendations based on the popularity of the movies.	So that I can select the movies that I like without much navigation.
C3	As a customer	I want to be given a set of recommendations based on the similarity of the movies that I liked before	So that I can select the movies that I like without much navigation.
C4	As a customer	I want to see the recent movies that I have rated with my ratings	So that I can be reminded which movies I have rated with good ratings or bad ratings
C5	As a customer	I want to see the average ratings of the recommended movies	So that I can have an ideas how the others rated the recommended movies
C6	As a customer	I want to be given a set of recommendations based on what the others similar to me like	So that I can select the movies that are liked by the people who are similar to me
S1	As a staff	I want to check the top 10 movies at a specific period. Also the number of ratings given has to be at least equal to the minimum number of ratings specified	So that I can know the top 10 movies that are popular at what period of time
S2	As a staff	I want to check which movie genres and tags are popular / unpopular	So that I can determine which types of movies the company should upload to the system
S3	As a staff	I want to check the distribution of the movie ratings	So that I can have an ideas what ratings are generally given by the customers
S4	As a staff	I want to check the number of ratings given per customer for each year	So that I can have an ideas how active customers are regarding to rating movies
S5	As a staff	I want to see the reasoning behind the recommendations	So that I can determine whether the recommendations make any sense

Table A.3 Functional Requirements

NFR ID	Description
NFR1	The page that displays the recommended movies to customers should be loaded within 3 seconds
NFR2	The page that displays the statistics information to staff should be loaded within 3 seconds
NFR3	The system should be accessible by using WEB browser or mobile application
NFR4	The system should have resilience
NFR5	The system should be 99.99% available
NFR6	The system should be able to recover or failover to DR within 15 minutes when disaster happens in the primary site
NFR7	The system should be scalable

Table A.4 Non-functional Requirements