

News and Movies Recommendation Systems

CS7IS5 - Adaptive Applications Group Report

Saksham Sinha, Conor McKenna, Neenu Vincent
Nabanita Roy, Jitesh Kumar Singh , Radhe Shyam Yadav
Team Jericho

April 15, 2018

Contents

1	Introduction	2
2	Background	2
3	News Recommendation System	4
3.1	System Architecture	4
3.2	News Source	4
3.3	Keyword Extraction	4
3.4	User Model	5
3.5	Modeling approaches	5
3.5.1	Explicit Modeling	5
3.5.2	Implicit Modeling	5
3.6	The Ranking Algorithm	5
3.7	Implementation	5
3.7.1	User model table	6
3.7.2	User Static Preferences	6
3.7.3	User Static Preferences Form	6
3.7.4	User Static Preferences View	6
3.7.5	News Profile Model	7
3.7.6	Like and See More Activity	7
3.8	Challenges Faced	7
3.9	Further Improvements	8
4	Movie Recommendation System	8
4.1	Datasets	8
4.2	Dataset Preprocessing	8
4.2.1	Dataset reading	8
4.2.2	Small dataset metadata dataframe creation	9
4.2.3	Extraction of movie cast and director from the credits dataframe	9
4.2.4	Indexing and cosine similarity calculation	9
4.3	Content-based filtering process	9
4.4	Collaborative filtering process	10
4.5	Hybrid filtering process	10
4.6	Challenges	11
5	Conclusion	11

1 Introduction

Over the last decade, there has been an information explosion due to rise in the use of Internet and social media. There is an abundance of content available throughout the web with huge amounts of data being uploaded or created every second. In this digital age, its becomes highly difficult to search for data of personal interest. Information retrieval systems are developing as well but cannot serve the purpose of correctly finding relevant data according to the ever changing preferences of the user. This gap between the user preferences and relevancy is bridged by adaptive applications, in this context, a recommender system. Adaptive systems and recommender systems have shared goals of identifying user behavior and act based on the captured data for personalization.

Recommendation systems are systems that recommend some data to user, based on the visibility of the user's preferences. The type of data that recommender system will recommend depends on for what purpose is the recommendation system designed for. In this report, we have documented two recommendation systems which we have developed and demonstrated as a part of the coursework requirement - Real-time News Recommendation System and Movie Recommendation System.

Further report is structured into the following sections- section 2 describes the background research conducted before making the systems, section 3 describes the development of News Recommendation system and the problems faced, section 4 describes the development of movie recommendation system and in section 5 we conclude with discussions and future work.

2 Background

Recommender systems help in searching and navigating the vast sea of items online by finding content that corresponds to the users' preferences. These systems provide personalized suggestions for items to users based on ratings estimated for those items(Adomavicius and Tuzhilin, 2005). A lot of recommendation systems have been developed since the year 2000, for example, recommendation of shopping items such as Books, Electronics, Accessories by Amazon.com(Linden et al., 2003), movies recommendation system by Movielens(Miller et al., 2003), or an intelligent news agent that can recommend news to users by learning users preferences and constructing explanations for the reasons that have led to a specific classification of news(Billsus and Pazzani, 1999).

Present research in the adaptive hypermedia field addresses the challenges of target user modeling and domain modeling which aims to capture the user's characteristics as well as activity paradigms and present applications tailored to their interests and benefits(Bull and Kay, 2010). In real-time scenarios, the core of recommender systems are the concepts of Adaptive Hypermedia Systems(AHS) like user modeling, adaptive information filtering, collaborative filtering, hybrid filtering and adaptive navigation.

Adaptive filtering approaches to extract the most relevant information related to a user can be broadly classified into:

1. **Content-based Filtering:** These systems filter and recommend certain data or items to a user that are similar to what that user has liked or preferred in the past. Content based recommendations works by filtering the the items based on the preferences of the user. This has been developed due to the research in information filtering and ordering. One of the earliest pieces of research was done by Belkin and Croft (1992) where they explained how information filtering can be applied to the large amounts of data, streams of data and requires descriptions of individual or group information preferences, often called profiles. Content-based systems are designed mostly to recommend text-based items, the content in these systems is usually described with keywords and metadata.
2. **Collaboration-based Filtering:** These systems filter and recommend data or items to a user based on other user's preferences who has similar features or a similar activity paradigm. Collaborative filtering only utilizes the ratings of training users in order to predict ratings for test user(Jin et al., 2003). According to the authors Breese et al. (2013) collaborative filtering methods fall into two categories:
 - (a) Memory-based algorithms: Memory-based algorithms store rating of users in a training database, and predict the user's ratings based on the corresponding ratings of the users in the training database that are similar to the test user.

- (b) Model-based algorithms: Model-based algorithms build models that can fit the training data well and predict the ratings of test users using the trained models.
- 3. **Hybrid Filtering:** These systems usually combine the results from the first two approaches i.e. content based filtering and collaboration based filtering to recommend the items to a user. According to the authors [Adomavicius and Tuzhilin \(2005\)](#), a hybrid recommender system can be made by below listed approaches-
 - (a) Implementing collaborative and content-based methods separately and combining their predictions.
 - (b) Incorporating some content-based characteristics into a collaborative approach
 - (c) Incorporating some collaborative characteristics into a content-based approach
 - (d) Constructing a general unifying model that incorporates both content-based and collaborative characteristics.

Our project on News Recommender System is based on the Topic detection and tracking (TDT) methodology which aims to categorize the incoming news into known topics ([Wang et al., 2018](#)). In their research paper, [Wang et al. \(2018\)](#) proposed a time-effective news recommender system which ranks the topics based on the a score determined by a scoring heuristic. The heuristic function is based on the number of documents from which the keyword is extracted and greatest number of days that a word continuously appears for as a keyword ([Wang et al., 2018](#)). They experimentally demonstrated the effectiveness of their system but they did not apply the principles of this recommender system on massive datasets. In another research on news recommender system conducted by [Lu et al. \(2015\)](#) focuses on content-Based collaborative filtering and implicit feedback. The implicit feedbacks are derived from user activities, such as browsing items, items which are marked liked, disliked or favorite, etc ([Lu et al., 2015](#)).

Another big application of recommender system can be seen with movies. [Adomavicius et al. \(2005\)](#) explains that recommendation of various products in an online store, or movies, it may not be sufficient to consider only users and items rather it is also important to incorporate the contextual information of the user's decision scenario into the recommendation process. Authors developed a multidimensional recommendation model(MD model) to provide recommendations incorporating contextual information and makes recommendations based on multiple dimensions and, therefore, extends the classical two-dimensional (2D) $Users \times Items$ paradigm. For example, to recommend a movie to a person who wants to see it in a movie theater on a Saturday night, authors' method will use only the ratings of the movies seen in movie theaters over the weekends, if it is determined from the data that the place and the time of the week dimensions affect the moviegoers' behavior. Moreover, they combined local machine learning methods with On-Line Analytical Processing (OLAP) and marketing segmentation methods to predict unknown ratings.

Another approach to movie recommendation is by using matrix factorization. [Zhao et al. \(2016\)](#) used matrix factorization method for recommending movies to user. In traditional matrix factorization, the problem can be formulated as inferring missing values of a partially observed User-Item matrix X where each row represents a user u and each column an item v . Then, one can model user/item preference within each matrix entry x_{uv} by low-rank factor matrices $U \in \mathbb{R}^{k \times n}$, respectively, where the u -th user and the v -th item are represented by U_{*u} and V_{*v} , corresponding to the u -th and v -th column of preference matrices U and V . Authors used the Singular Value Decomposition (SVD) on the User-Movie preference matrix. The prediction is given by-

$$\hat{x}_{uv} = b_{uv} + U_{*u}^T V_{*v}$$

, where b_{uv} denotes a baseline estimate for an unknown rating x_{uv} :

$$b_{uv} = \mu + b_u + b_v$$

, and μ is the overall average rating, b_u and b_v indicates the biases of user u and movie v , respectively.

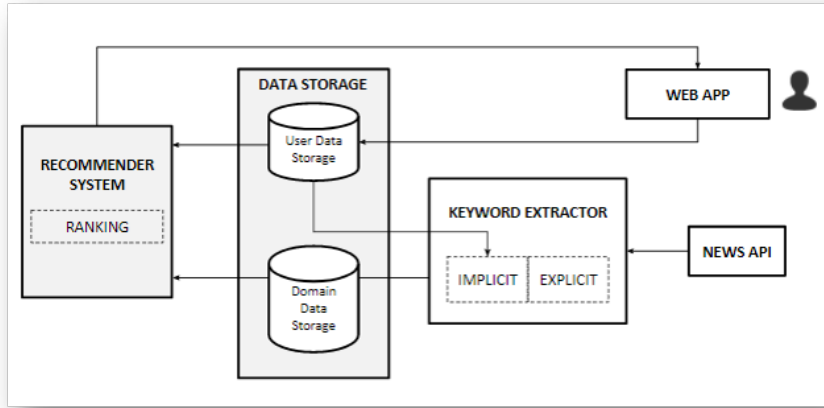


Figure 1: System Architecture.

3 News Recommendation System

3.1 System Architecture

The architecture of the news recommender system was designed to capture the keywords and categorize the incoming news into a set of pre-defined categories. The system does not only act in the user-selections but also on the actions captured at the back-end denoted as "Implicit" in the "Keyword Extractor" component of Figure 1. The data extracted from the user interaction in the application and the news articles are then modeled for the ranking based on a scoring-ranking algorithm.

3.2 News Source

We used the NewsAPI at <https://newsapi.org/> which provides the following information for each news:

- source name
- news title
- url
- author
- description
- published time

3.3 Keyword Extraction

For the project, we tried to extract keywords from scratch by experimenting with

- Latent Dirichlet allocation(LDA) Topic Modelling
- Using Stanford Named Entity Recognizer(NER) tagger¹ and then connecting to DBPedia in order to derive the meaning of the entities identified.
- TextRazor API ²

Since, first two approaches were time-consuming and does not fit into the concept of real-time news recommendation, therefore we resorted to use the TextRazor API which provides NLP services including keyword and topic extraction. We categorized the incoming news into the following categories:

- Politics
- Sports
- Arts, Culture and Entertainment
- Finance, Business and Economics
- Science and Technology
- Miscellaneous

¹<https://nlp.stanford.edu/software/CRF-NER.html>

²<https://www.textrazor.com/>

3.4 User Model

We have used feature based user model in our system as it allows for fine-grained user modeling. Each user is encoded in terms of categories as features and tuple-type encoding has been used to encode i.e. each user has 6 set of features and each has a value for example User 1 is encoded as {science: 1.5, economics: 0.5, politics: 0.0, sports: 1.5, arts: 0.0 , misc: 0.0}.

3.5 Modeling approaches

3.5.1 Explicit Modeling

A user can select their preferences from the list of pre-defined news categories. These are static preferences. The user can also modify the selections. According to these selections the scores of the categories are boosted. A weight of 0.5 will be added to the scores of the selected categories. For example, if a user likes Politics and Economics as his preferred categories, scores for these categories will be boosted the system will recommend more news from Politics and Economics and these news will be shown first.

3.5.2 Implicit Modeling

A user can like or dislike the news as well as click on the link and view the news. We capture these actions to boost(for like and news view) or apply a penalty(for dislike) on the category scores.

3.6 The Ranking Algorithm

Show More	Like	Weight
0	0	0.1
0	1	0.5
1	0	0.5
1	1	0.9

Table 1: Weight assignment table

Table 1 shows the table of dynamic updates of scores for the categories. For each interaction of the user, the score will be updated with the weight assigned to that interaction. Since we are capturing two interactions show more and like, if a user performs either one of the interaction for a news, than that news's category score will be boosted by the weight assigned.

The ranking is done by calculating the likelihood of each category to be displayed in the UI. The main objective of the ranking system is to rank the categories considering user's static and dynamic changes so that more news of preferred category is shown in th UI sorted according to rank. Along with this, we should also display news from other categories with least or no scores for the user to get to know the happenings in other least preferred categories. Hence, a likelihood of 0.05 (5% out of 100% likelihood) is maintained for each category if the user has any static or dynamic preferences. Since we have 6 defined categories, 5% for each of them taking around 30% of the total 100% likelihood. The remaining 70% is modelled according to user's preferences. As mentioned earlier the scores of each category will be boosted according to implicit and explicit preferences. These scores are then normalized by dividing each individual category scores with the total scores calculated by summing up the scores of all the categories. This is then converted into percentages. These percentages are mapped to 70% scale. Thus, the final likelihood percent of each category is obtained by summing up the likelihood calculated in 70% scale and 5% set earlier. If the user has no static or dynamic preferences, equal likelihood percent is assigned to all the categories. That is, 16.66%.

3.7 Implementation

We used Django framework with python 3.5 and SQL lite for implementation of the system. For our news recommendation application, it was important to create a responsive web app that was able to change the parameters used in adapting the application without disrupting the user experience

or flow of using the application. To do this, we used a number of backend entry points that are posted to using AJAX requests from the front end.

3.7.1 User model table

To be able to allow for unique user registration and login into our application, we implemented the Django Auth User Model, which provides everything we needed to create a signup page and table in the database for users. Once we had this basic user model, we were able to expand upon it to add more flexibility to the program.

3.7.2 User Static Preferences

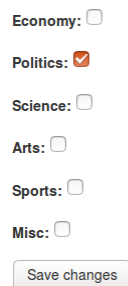
This was where we ended up storing the boolean values for a user's preferences based on news categories, as well as the dynamic values for each as decimal values. The model follows this trimmed down structure:

This model references the user table using the currently signed in user as a foreign key. We don't assume anyone's preferences at the beginning, so their static preferences are set to false, and their dynamic values are set to 0.0 as a default.

3.7.3 User Static Preferences Form

When the user first registers with the application, they are presented with a page with where they can set their static preferences via a form input. Figure 2 shows the user static preference form.

User Profile test



The form contains six rows, each with a category name and a checkbox: Economy, Politics, Science, Arts, Sports, and Misc. The Politics checkbox is checked, while the others are unchecked. Below these rows is a 'Save changes' button.

Figure 2: User static preferences form.

These static preferences can be changed at any time from the User Profile tab. When these settings are confirmed, a weight of 0.5 is added to the corresponding dynamic preference value (ie: Economy \Rightarrow dy_economy). These values are then used to influence the order that the news appears in.

3.7.4 User Static Preferences View

First off, to be able to access the form the user must of course be signed in to the application. Depending on whether the request to the update_profile endpoint is a GET or POST the user will either be served with the preferences form, or will be redirected to the browse view on successful posting. When the request method is POST, a new UserStaticPrefsForm is created in the database to store the new static preferences. Since the dynamic preferences aren't accounted for in the form, they would normally be overwritten. To combat this, the old UserStaticPrefs object is stored into a temporary variable, and its dynamic preference values are set for the new object before it's saved. The old model is deleted so there is only one entry per user, making looking up by userID simpler.

3.7.5 News Profile Model

Before being able to record user click activity, we needed to create a new model that references both the user and the news object in question. To do this, we created the NewsProfileModel, which consists of the following:

1. User - Foreign Key to User table
2. News - Foreign Key to News table
3. show_more - Boolean
4. relevance - Boolean

3.7.6 Like and See More Activity

A NewsProfileModel is created, or updated if already exists, when either the Like button is pressed or the link to the news article is clicked.

News Recommendation App



Figure 3: User clicking on like button for the sports news.

Once one of these are clicked, an ajax post request is made to the backend of the system. The database is then queried to see if an object already exists, and if not it creates a new one, else it updates the existing one. Figure 3 shows user clicking the like button for the first news. The ajax request carries data that is used to check the relevant boolean field to true. We don't provide the option to unlike. The application then applies additional weights to the user's preferences based on whether they have clicked for "show_more" or for "relevance", with having clicked both combined adding more weight than they would individually.

3.8 Challenges Faced

- Topic modeling: The first challenge was to correctly identify the topics associated with the news as we are not using a well-defined dataset with categorical information. Besides, the NewsAPI also does not provide information on which category the news fits into. We finally used TextRazor to solve this problem.
- Collaborative Filtering: Since we did not have user data except for the ones that we created for testing the application, we could not use Machine Learning for collaborative filtering and therefore we encountered a cold-start setting.

Overall, the limitation of this project was in the cold-start problem we faced because we could have also trained a model for collaborative filtering and news article classification using text classification techniques. Besides, gender and age of the users are also beneficial for recommendation systems but this requires a huge dataset again. Therefore, the movie recommendation use-case was built to demonstrate the different adaptive filtering techniques that can be applied on recommender systems.

3.9 Further Improvements

- The current categories can be further sub categorized.
- Gender is an important factor in article recommendation and that can be considered for recommendations.
- Extracting current events happening across the globe, for example football match scores in a column.
- Dynamic preference values could decay over time as user interest in certain topics begin to fade.

4 Movie Recommendation System

4.1 Datasets

We have used Movielens datasets curated by [Harper and Konstan \(2015\)](#) and is publicly available at [Kaggle](#) . They have curated and assembled two versions of datasets which are -

1. Full Dataset: This dataset consists of 26,000,000 ratings and 750,000 tag applications applied to 45,000 movies by 270,000 users. It also includes tag genome data with 12 million relevance scores across 1,100 tags.
2. Small Dataset: This dataset comprises of 100,000 ratings and 1,300 tag applications applied to 9,000 movies by 700 users.

We have used the small dataset to build the recommendation system as working on full dataset requires lots of computing power and time.

Description of dataset files are as follows-

1. **movies_metadata.csv:** The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.
2. **keywords.csv:** Contains the movie plot keywords for the MovieLens movies. Available in the form of a stringified JSON Object.
3. **credits.csv:** Consists of Cast and Crew Information for all our movies. Available in the form of a stringified JSON Object.
4. **links_small.csv:** Contains The Movie Database(TMDB) and the Internet Movie Database(IMDB) IDs of a small subset of 9,000 movies of the Full Dataset.
5. **ratings_small.csv:** The subset of 100,000 ratings from 700 users on 9,000 movies.

Column 'id' in the metadata.csv, keywords.csv, credits.csv and ratings.csv corresponds to TMDB id in the links_small.csv.

4.2 Dataset Preprocessing

4.2.1 Dataset reading

We read each csv file as a pandas dataframe. Each movie has multiple genres which are expressed as a list of dictionaries having genre id and genre name. Hence we dropped all the ids of the genre and kept only genre names.

All the numbers present in the dataset files were read as string, hence we converted those columns to integer or float depending on the type of column. For example, all columns with ids and vote counts were changed from string to integer and votes average was changed from string to float. Then we merged the keywords.csv dataframe and credits.csv dataframe to metadata.csv dataframe on the column id.

4.2.2 Small dataset metadata dataframe creation

After reading in all the data, we had information of all the 45,000 movies with its respective keywords and credits. As we wanted to work with the small dataset, we took the subset of this dataframe if the id were present in the links_small.csv dataframe as each id in the metadata dataframe corresponds to TMDb id in the links_small.csv.

4.2.3 Extraction of movie cast and director from the credits dataframe

The credits dataframe had cast column, crew column and id of the movie where cast and crew column are list of dictionaries but were read as a string. Hence, we used literal_eval parser to parse the cast and crew information as a list of dictionaries from the metadata dataframe. We added two more columns in the metadata dataframe which were cast size and crew size. From the newly created crew column, we extracted director based on if the job title of that crew member was director and was saved as a new column director and the director name was appended three times to put more weight on the director in the metadata. Rest of the crew information was discarded.

Now stemming was performed using the Snowball stemmer on the keywords column. We created metadata column in the small metadata dataframe by merging the columns keywords, cast, director and genre for each of the 9000 movies. This metadata column will be used for content based filtering.

4.2.4 Indexing and cosine similarity calculation

We changed the index of small metadata dataframe to title and the index series was saved as user will be searching based on the title, hence searching based on title can be vastly improved. Later, we created a term vector matrix on metadata column using the CountVectorizer with 'word' analyzer, ngram token range from 1 to 2 and english stop words and calculated the cosine similarity for that matrix. This cosine similarity will be used later in the content based filtering.

4.3 Content-based filtering process

For content based filtering and recommendation, a user will input the movie title that the user liked and on the basis of the input and metadata of the movie entered, other movies having closely related metadata will be returned. Cosine similarity will be used to find the closely related movie titles which have similar metadata.

The following process is followed when the user inputs the movie title-

1. The index of the title is found from the indices series.
2. Movies indices with similarity scores for that index is taken using the cosine similarity calculated above.
3. The movies indices are ordered according to the similarity scores in the descending order.
4. The top 26 movies indices were taken from the whole list of similar movies which have either a common director or a common cast or share a common genre or have similar keywords.
5. Since, we wanted to recommend the movies which are also popular from the selected 26 movies, we extracted the title, vote counts and votes average.
6. Mean value was calculated from the vote average of each movie and a threshold of 60th percentile of the total vote counts was set.
7. A weighted rating was calculated using the votes average, votes count, mean and vote count number at 60th percentile. The IMDB weighted rating equation was used to calculate the rating.

$$weightedrating(WR) = (v \div (v + m)) \times R + (m \div (v + m)) \times C \quad (1)$$

Where:

R = average for the movie (mean) = (Rating)

v = number of votes for the movie = (votes)

m = minimum votes required to be listed in the recommendation which is set as 60th percentile vote count.

C = the mean vote across the whole 26 movies.

8. The movies were again sorted according to the weighted rating in the descending order and the first 10 movies were selected as the recommendation.

Using the metadata and popularity we made the recommendations that were the most popular movie titles which had similar metadata.

Content based recommendation for The Dark Knight						
	title	vote_count	vote_average	year	wr	
7648	Inception	14075	8	2010	7.872621	
8613	Interstellar	11187	8	2014	7.843131	
6623	The Prestige	4510	8	2006	7.662461	
3381	Memento	4168	8	2000	7.641301	
8031	The Dark Knight Rises	9263	7	2012	6.936548	
6218	Batman Begins	7511	7	2005	6.923913	
1134	Batman Returns	1706	6	1992	6.206438	
132	Batman Forever	1529	5	1995	5.676520	
9024	Batman v Superman: Dawn of Justice	7189	5	2016	5.224783	
1260	Batman & Robin	1447	4	1997	5.167624	

Figure 4: Top 10 movie recommendations if the user likes The Dark Knight.

Figure4 shows the results of content based recommendations of the top 10 movie titles for a user if that user likes The Dark Knight movie. According to the weighted rating, movie title such as Inception, Interstellar etc. which are from the same director and are highly popular are recommended first, while the other movies having the same genre and have similar keywords here "Batman" have been recommended lower.

4.4 Collaborative filtering process

For the recommendations based on the collaborative filtering, we used the python scikit surprise package which provides many state of the art implementations for building the recommendation system. For collaborative filtering, we used the state of the art Matrix factorization algorithm Singular Value Decomposition as used by Zhao et al. (2016).

For this, we used the ratings.csv dataset and read it as a pandas dataframe and loaded it into the scikit surprise dataset. Surprise dataset allows for easier splitting of dataset into training and testing. We split the dataset into 5 folds for training and testing. The results of evaluation is shown in the figure5.

Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
MAE (testset)	0.6910	0.6955	0.6899	0.6905	0.6876	0.6909	0.0026
RMSE (testset)	0.8971	0.9041	0.8939	0.8981	0.8922	0.8971	0.0041
Fit time	6.72	6.70	6.76	6.96	4.08	6.25	1.09
Test time	0.24	0.28	0.23	0.14	0.25	0.23	0.05

Figure 5: Mean Absolute Error(MAE) and Root Mean Square Error(RMSE) of the test set

We received a mean RMSE of 0.8971 which was good signal for us and standard deviation of 0.0041 showed that the model was not overfitting and that the data was not skewed. We used the trained model for predicting how much a user would like a given movie based on the ratings from the similar users for that movie. These rating predictions will be used later for the hybrid recommendations of movies for a user.

For example, if we need the prediction of a rating for a movie "Toy Story" with the id=862 from a user with id=1, the collaborative filtering will look at the ratings given by the similar users for the movie "Toy Story" and predict the same for the user with id=1, in this case, 2.62 out of 5.

4.5 Hybrid filtering process

For the hybrid filtering process, we follow the same process of finding the recommendations from the content based filtering upto finding the top 26 movies according to similarity scores for the

similar metadata, and on those 26 recommended movies we apply the SVD trained model to predict how much a user will like those recommended movies based on the other similar users ratings for those 26 movies. At the end, we sort the recommendations based on the predicted ratings and present the final top 10 recommendations to the user.

Hybrid recommendation for The Dark Knight				
	title	year	id	est
3381	Memento	2000	77	3.427190
6623	The Prestige	2006	1124	3.353943
7648	Inception	2010	27205	3.332674
5943	Thursday	1998	9812	3.061507
8613	Interstellar	2014	157336	3.041633
6218	Batman Begins	2005	272	2.973763
8031	The Dark Knight Rises	2012	49026	2.970170
4021	The Long Good Friday	1980	14807	2.899048
7362	Gangster's Paradise: Jerusalem	2008	22600	2.890833
5809	Point Blank	1967	26039	2.864778

Figure 6: Hybrid recommendations for a user who liked The Dark Knight movie.

Figure 6 shows the movie recommendations for the user who liked the movie The Dark Knight. The movie titles are found from the content based recommendation and ordering is done using the collaborative filtering. In contrast to content based recommendation results in figure 4, where the Inception was rated as top recommendation for the user has now shifted to 3rd position due to collaborative filtering rating prediction for that movie is lower for that user. Similarly, movie Thursday which wasn't recommended as a top 10 recommendation from the content based recommendation, is now at the 4th position because the user might like this much more as similar users have rated it quite high.

4.6 Challenges

The implementation of Movie Recommender System was much complicated and a time-taking process. The challenges that we faced are:

- Processing huge dataset: It is a time-consuming process and hard to locate errors.
- Understanding and implementing the core concepts of content-based, collaborative and hybrid filtering
- Since we had already worked on an use-case, we had less time to implement the Movie Recommendation System and therefore we could not implement the UI for it.
- Since our objectives were slightly unique from other projects, we had to invest time in understanding which python package is beneficial to use.

5 Conclusion

In conclusion, we successfully implemented a functioning adaptive application with personalised recommendations in our News Recommendation web app. We were able to better understand and put various adaptive techniques into practice, and gained valuable insight into the thought process behind building adaptive applications. Having to overcome real world challenges such as the cold-start scenario we encountered and having to figure out how to best overcome it was a useful experience. Issues such as this and the issue of being unable to include collaborative and hybrid adaptive filtering techniques led us to explore how we would implement these in a separate domain, leading to our Movie Recommendation system. Time constraints and difficulty with collaborative implementation prevented us from integrating these techniques into the main News Recommendation application, but the proof of concept can be seen.

References

- Adomavicius, G., R. Sankaranarayanan, S. Sen, and A. Tuzhilin (2005, January). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.* 23(1), 103–145.
- Adomavicius, G. and A. Tuzhilin (2005, June). 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* 17(6), 734–749.
- Belkin, N. J. and W. B. Croft (1992, December). Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM* 35(12), 29–38.
- Billsus, D. and M. J. Pazzani (1999). A personal news agent that talks, learns and explains. In *Proceedings of the Third Annual Conference on Autonomous Agents*, AGENTS '99, New York, NY, USA, pp. 268–275. ACM.
- Breese, J. S., D. Heckerman, and C. M. Kadie (2013). Empirical analysis of predictive algorithms for collaborative filtering. *CoRR abs/1301.7363*.
- Bull, S. and J. Kay (2010). Open learner models. In *Advances in intelligent tutoring systems*, pp. 301–322. Springer.
- Harper, F. M. and J. A. Konstan (2015, December). The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* 5(4), 19:1–19:19.
- Jin, R., L. Si, and C. Zhai (2003). Preference-based graphic models for collaborative filtering. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03, San Francisco, CA, USA, pp. 329–336. Morgan Kaufmann Publishers Inc.
- Linden, G., B. Smith, and J. York (2003, Jan). Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7(1), 76–80.
- Lu, Z., Z. Dou, J. Lian, X. Xie, and Q. Yang (2015). Content-based collaborative filtering for news topic recommendation. In *AAAI*, pp. 217–223.
- Miller, B. N., I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl (2003). Movielens unplugged: Experiences with an occasionally connected recommender system. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, New York, NY, USA, pp. 263–266. ACM.
- Wang, Z., K. Hahn, Y. Kim, S. Song, and J.-M. Seo (2018, Feb). A news-topic recommender system based on keywords extraction. *Multimedia Tools and Applications* 77(4), 4339–4353.
- Zhao, L., Z. Lu, S. J. Plan, and Q. Yang (2016). Matrix factorization+ for movie recommendation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pp. 3945–3951. AAAI Press.