THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Recommendation Systems

A Study of Different Approaches

by

Arjun C

School of Information Technology and Electrical Engineering,

University of Queensland.

Submitted for the degree of Master of Computer Science

Date of Submission: 06 November 2017

Arjun C
44632339
a.c@uqconnect.edu.au
2, Sovereign Street, Indooroopilly, QLD 4068

06 November 2017

Prof Michael Bruenig
Head of School
School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia QLD 4072

Dear Professor Bruenig,

In accordance with the requirement of the Degree of Master of Computer Science in the School of Information Technology and Electrical Engineering, I submit the following thesis entitled

"Recommendation Systems – A Study of Different Approaches"

The thesis was performed under the supervision of Prof. Neil Bergmann. I declare that the work submitted in this thesis is my own, except as acknowledged in the text and footnotes, and has not been previously submitted for a degree at the University of Queensland or any other institution.

Yours sincerely

Arjun C

To my parents and well wishers

Acknowledgements

I would like to thank my supervisor, Prof. Neil Bergmann, for providing me with the opportunity to work on this case study. His guidance helped me in tackling issues since the beginning of the thesis, be it coming up with an idea or helping out by providing insights on the important aspects of machine learning to achieve the intended goal.

I would also like to extend my gratitude to Ms. Saskia Van Ryt, who guided me through the complete process, sharing her insights and constructive feedback on the work being carried out.

I express my deepest gratitude to my parents and my family who provided me with all the love and support and to whom I am indebted for my life. Furthermore, I thank all the people who have believed in me and made me who I am today.

Abstract

Recommendation systems, also called recommender engines, are the tools that help in suggesting users, the items of their interest, on the basis of their past behaviour. Given the amount of content made available on the internet today and the shift towards online shopping, it is vital to provide a consolidated view of things the users are interested in, than everything available in-store. Since the user to item relationship is sparse, it is important to come up with effective ways to make this happen. Also, it is very essential to consider the relevant business requirement and the problem statement. In this study, some of the basic approaches for building a recommendation system like popularity based method, content filtering and collaborative filtering are investigated. Also, a very simple yet effective technique using Auto-Encoders is employed and evaluated, which is an inexpensive way in terms of computation and time when compared to the matrix factorization method.

Table of Contents

Acknowledgements v

Abstract vii

List of Figures x

Introduction 1

1.1 Motivation 4

1.2 History of Recommendation Systems 5

1.3 Recommender Systems - Problem Statements 6

1.4 Aim and Scope 7

1.5 Research Questions 7

Literature Review 8

2.1 How it all began 8

2.2 Recommendation systems today 11

2.3 State of the art 14

2.4 Evaluation 15

Theory 18

3.1 Popularity Based Recommender Systems 19

3.1.1 Problem Statement 19

3.1.2 When to use this type of system? 19

3.1.3 Sample Data 20

3.1.4 Implementation 20

3.1.5 Evaluation 20

3.2 Content Based Filtering 21

3.2.1 Problem Statement 21

3.2.2 When to use this type of system? 22

3.2.3 Sample Data 22

3.2.4 Implementation 23

3.2.5 Evaluation 24

3.3 Collaborative Filtering 24

3.3.1 Problem Statement 24
3.3.2 When to use this type of system? 25
3.3.3 Sample Data 26
3.3.4 Memory based method 26
3.3.5 Model Based method 29
Implementation and Results 35
4.1 Data Analysis 35
4.2 Implementation details and results 40
4.2.1 Popularity based recommender system 40
4.2.2 Content Based Method 41
4.2.3 Collaborative Filtering – Matrix Factorization 43
4.2.4 Collaborative Filtering – Autoencoder 45
Discussion 48
5.1 Popularity Based Recommendation systems 49
5.2 Content based filtering method 49
5.3 Collaborative filtering – Matrix Factorization 50
5.4 Collaborative filtering – AutoEncoders 51
Conclusions 53
6.1 Conclusion 53
6.2 Future Work 54
References 56

List of Figures

Figure 1. Recommendation based on search for Rubber Bath Duck on amazon.com 2

Figure 2. Recommendation for movies similar to Batman Begins (2005) 3

Figure 3. Facebook friend recommendations based on existing connections 3

Figure 4. Rise of tapestry (a) email document overload (b) distribution lists (c) conventional filtering (d) collaborative filtering [2] 9

Figure 5. Flow of Documents through Tapestry system [2] 10

Figure 6. Grouplens Architecture [7] 11

Figure 7. Netflix Recommendation System Architecture [8] 12

Figure 8. Netflix Recommendation System - Offline Task Architecture [8] 13

Figure 9. Confusion Matrix 17

Figure 10. Sample data for Popularity based method 20

Figure 11. Sample Data Subset for Content Based Filtering 22

Figure 12. K-Nearest Neighbor algorithm for classification 23

Figure 13. Collaborative Filtering Intuition 25

Figure 14. Sample Data for Collaborative in general 26

Figure 15. User-based (a) and Item-based (b) collaborative filtering 26

Figure 16. User Based Collaborative Filtering Example 27

Figure 17. Item Based Collaborative Filtering Example 28

Figure 18. Matrix Factorization Algorithm [6] 31

Figure 19. AutoEncoders – Typical Architecture 32

Figure 20. Movie Genre Word Cloud representation 35

Figure 21. Movies - Word Cloud 36

Figure 22. Number of Movies by Genre [17] 37

Figure 23. Genres by Year [17] 37

Figure 24. Average ratings by year [17] 38

Figure 25. Occupation vs Genre [17] 38

Figure 26. Age vs Genre with gender information [17] 39

Figure 27. Input data - Popularity based method 40

Figure 28. Recommendations generated by Popularity based method 41

Figure 29. Input data for content based recommendation system 42

Figure 30. Training RMSE for Matrix Factorization 45

Figure 31. Autoencoder model summary 46

Figure 32. AutoEncoder RMSE 47

Figure 33. Autoencoder Rating prediction for a subset of movies for a user 47

Chapter 1

Introduction

Internet, its birth and advancement has revolutionized the way users consume data. Resources that were once expensive and sluggish are now made available in seconds. We now have collaborative online encyclopedia like Wikipedia which provides easy information access to users. The rise of social media has completely changed the way people interact with the web. The number of active users of social media sites has increased beyond our wildest dreams. The immense growth of the social media and ecommerce sites has resulted in an information overload problem. We had too much information than we could process and find meaning in it. It was hard to make sense of the available data even with narrowed area of interest. Fortunately, with advancement in the computation sciences, we have reached a point where we are currently able to filter the necessary data and extract some meaning out of it. In fact, many online applications and websites now make full use of the user interactions with the internet, and the interactions between the users to improve their online experience.

Sites like Wikipedia, personal blogs, social media sites like Facebook, Twitter etc., digital content sites like YouTube, Netflix etc., e-commerce sites like Amazon, eBay etc. are some of the examples using which users can both contribute (by rating a product or reviewing some of the items) and gain information from (by accessing reviews or ratings by other users). Also, there are surveys conducted online offering discounts on purchases. This gives out a lot of personalized information. It is very critical for the sites to make use of this data to improve the user experience, thus increasing the profits. It is critical to understand that in this fast world, no user would want to look for needle in a haystack.

Considering the focus area for this thesis, the goal is to make sense out of data depicting user-item, user-user and item-item relationships. Contextually, an item is any product or entity of interest. User is any internet end user who has rated or reviewed an item. This serves as the data for our information filtering system, Recommendation system.

Recommendation systems are the tools that make use of this aforementioned data to identify what a user is most likely to invest his time or money on. This method helps in personalizing the content for a user, thereby improving user experience and not to forget, hits for the content or the sales of the item. The system comes up with suggestions for the users called recommendations. These recommendations play a significant role in some of the critical decision making processes. Also, it helps the users in accessing the items they like without investing more time on searching. These systems emerged in the last decade and are making the web experience exceptional. There have been many approaches proposed and being followed at the moment. Recommendation systems could be categorized as Popularity based (recommending most popular items to all users), Content based filtering (recommending items of similar features based on user preferences) and Collaborative filtering methods (recommending items based on user's past rating history and other users' rating hostory). These approaches will be discussed in the later chapters. Collaborative filtering has been the most commonly used algorithm to build personalized recommendations as they are not reliant on any manual feature classification. These systems are used in some of the most popular websites including Amazon, eBay, Moviefinder, and Netflix.

We come across recommendation systems every day in our life. Going through some of the real world implementations would help us in understanding what these systems are capable of. Looking at the image below, with just one search for "rubber bath duck" on amazon.com, the system is able to recommend the items similar to the previous search. The system could come up with suggestions even without the user purchasing any item or rating the items.

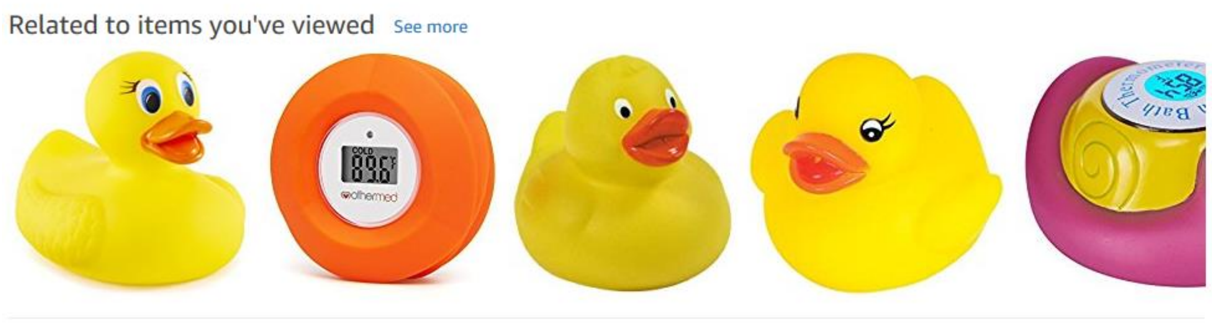
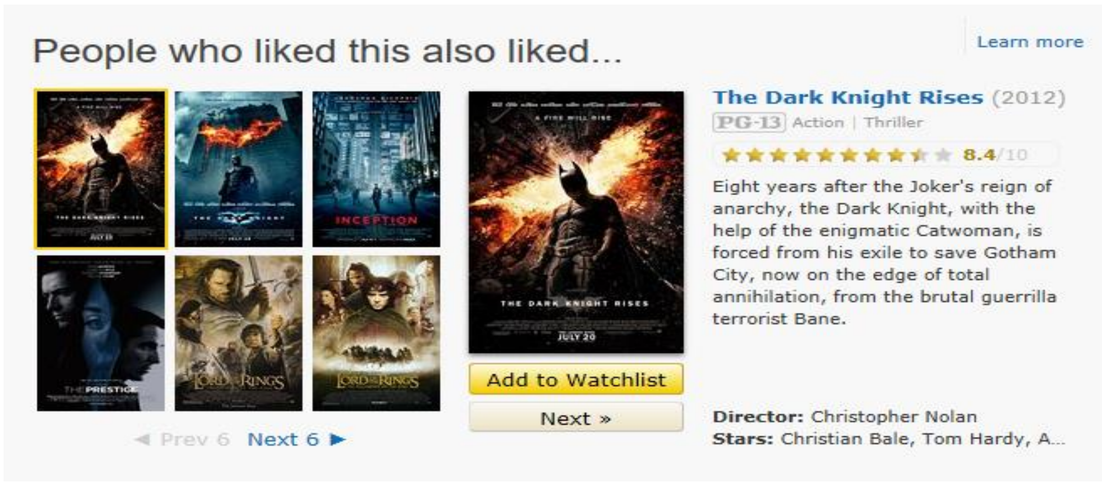


Figure 1. Recommendation based on search for Rubber Bath Duck on amazon.com


People who liked this also liked...

Learn more

The Dark Knight Rises (2012)

PG-13 Action | Thriller

8.4/10

Eight years after the Joker's reign of anarchy, the Dark Knight, with the help of the enigmatic Catwoman, is forced from his exile to save Gotham City, now on the edge of total annihilation, from the brutal guerrilla terrorist Bane.

Add to Watchlist

Next »

◄ Prev 6 Next 6 ►

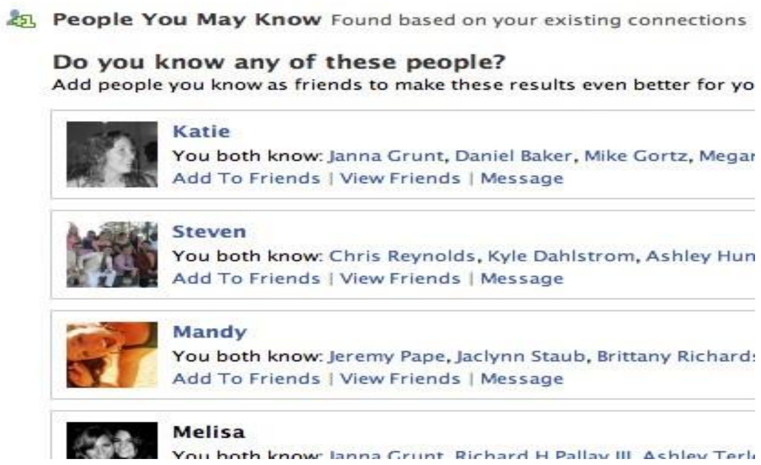
Director: Christopher Nolan

Stars: Christian Bale, Tom Hardy, A...

Figure 2. Recommendation for movies similar to Batman Begins (2005)

The above image shows how IMDB recommends related movies based on the searched movie. Searching for the movie “Batman Begins (2005)”, the system recommends the movies very closely related to that movie along with the movies with similar properties and users with similar interests.

The following image shows the Facebook recommendations to add friends based on the existing connections. These recommendation help us in getting in touch with friends that we are actually related to rather than random people on Facebook. This is one of the best examples of personalization.


People You May Know Found based on your existing connections

Do you know any of these people?

Add people you know as friends to make these results even better for yo

Katie

You both know: Janna Grunt, Daniel Baker, Mike Gortz, Megar

Add To Friends | View Friends | Message

Steven

You both know: Chris Reynolds, Kyle Dahlstrom, Ashley Hun

Add To Friends | View Friends | Message

Mandy

You both know: Jeremy Pape, Jaclynn Staub, Brittany Richard:

Add To Friends | View Friends | Message

Melisa

You both know: Janna Grunt, Richard H Pallay III, Ashley Terl

Figure 3. Facebook friend recommendations based on existing connections

The thesis mainly concentrates on implementing popularity based method, content based filtering method, collaborative filtering method using matrix factorization and autoencoders.

The following sections in this chapter introduce the motivation behind choosing this topic (Section 1.1) and provide a very brief introduction to history of Recommendation Systems (Section 1.2). The scope of this thesis and the problem statement are addressed in the subsequent sections.

1.1 Motivation

The most widely discussed motivation amongst the recommender systems research community has been unaltered for a very long period of time. It could be described as follows. Users are not able to make meaningful use of information given the rapid rate of information growth (e.g. With the amount of products being available for sale compared to what was available a decade ago, it is hard for the users to decide on what to buy or sometimes even how to look for them). Recommendations can solve this problem. It is assumed that huge amount of data is already available. This has to be used to improve business. Recommender systems that use personalized views have the potential to increase profitability for the business, encouraging people to actively use their site. It was noted that two thirds of the rented movies on Netflix were the movies recommended by the system. Almost 35% of the product sales on Amazon.com were recommendations. Recommendations on Google news has resulted in 38% increase of clicks [1]. It is evident that personalized experience is a key component for the success of a site.

Algorithms of Machine learning in general have been implemented as libraries in many programming languages for easy usage. For tasks like classification or regression, neural networks etc., currently there are a huge number of packages available. There have been a variety of algorithms proposed for the implementation of recommendation systems. But, when it comes to available packages, there is nothing much on recommendation systems. The publications all talk about recommendation systems, but when it comes to implementation, there are things that do not really agree with the theory published.

From a research perspective, it is one of the major applications of machine learning that we come across every day. The implementation of recommender systems covers some of the critical topics like neighborhood approach, matrix factorization, auto-encoders using deep learning etc. These concepts equip an interested machine learning enthusiast with some very interesting tools required given the current industry requirements. It is also noteworthy that there are a large number of job opportunities for people with basic understanding of recommender systems.

To sum up, data availability, user need for recommendations, company profitability, unavailable ready-to-use implementations and current industry trend given the job opportunities were the major reasons behind choosing this topic for the thesis.

1.2 History of Recommendation Systems

Recommendation systems have evolved in the past decade. They have changed the way internet is being used today. The first recommender system, Tapestry [2], was an experimental system developed at Xerox, Palo Alto Research Center in the early 1990s. The motivation behind Tapestry was the increased usage of email at the time. Users were receiving huge amount of documents in their mail. One of the methods followed to reduce the number of incoming mails was to provide mailing lists, enabling users to subscribe only to the lists they are interested in. In practice this did not work out as lists were not exactly as per user interests. Another approach was to let the users filter the mails they were interested in. This was the first implementation proposed that introduced the idea of collaborative filtering. This meant that people collaborated to help others in filtering documents by annotating their responses to the documents they were interested in. Tapestry used annotations to refer to what the document was related to. Considering the current social media scenario, annotations are what tags are today. This means, users could tag documents/articles as “funny” or “news” etc. A filter for a user ‘User A’ would be something like “Documents tagged as news by ‘User B’ and ‘User C’”. So, whenever documents are annotated as news by ‘User B’ and ‘User C’, ‘User A’ would receive the document by mail. This gave a general idea of what users liked and how similar different users were.

Later, similar concept was extended for Usenet, a project by GroupLens, which dealt with internet discussion forum, which grew so big making it impossible for a single user to manage [3]. The GroupLens project later implemented the MovieLens movie recommender system. The datasets were then made open-source and available to the research community. This shifted focus from news toward filtering movies.

The initial success of recommender systems can mostly be linked to e-commerce businesses that implement them. Schafer et al. put forth a discussion on a number of these examples [4, 5]. Amazon.com and CDNow.com, implement recommenders to build customer loyalty and to increase profitability. Last.fm's implementation for collecting user-music listening behavior, for personalized radio stations and music recommendations to their subscribers is worth noting.

The latest and the most significant event related to recommender systems was the Netflix prize in 2006. Netflix, an online DVD rental company from the US, released user-movie ratings dataset challenging the users to outperform their existing system by at least 10%. They also offered a million dollar cash prize for the winners. The offered amount shows how much of an increase in the profits it could make with better recommender system [6].

1.3 Recommender Systems - Problem Statements

Recommender systems are built to guide the users. They could be used to solve one of the following problem statements.

- Popularity based method: What are the most popular items to recommend?
- Content based filtering (CB): What are the items with similar features? Who are the users with similar features? (Item features or User features are explicitly known in this case.)
- Collaborative filtering (CF): What would the user rate an item given his ratings to other items and other users' ratings? (Item features or User features are not explicitly known in this case.)

The problem statement for each type of implementation will be discussed further in the later chapter along with the explanation of the method. Recommender systems/algorithms are implemented and trained with the completely available user-ratings matrix. These results can then be used to build recommendations. Training and building models process is repeated in regular time intervals. The main reason behind this is that the algorithms take a very long time to reach a stable point wherein it could be used for recommendations. Calculations are computationally expensive given the number of users, items and the data sparsity. It was seen that some of the algorithms require exponential complexity to solve the problem.

1.4 Aim and Scope

The primary goal of this thesis is to understand the basics of Recommendation systems, the implementation aspects of it and the issues faced when building them. This study focuses on building recommendation systems on the MovieLens 1-Million dataset.

This thesis covers study of some of the widely used methodologies like popularity based approach, an approach for content based filtering (k-nearest neighbor method) and two approaches for collaborative filtering - model based method(Matrix factorization method and Auto-Encoder method). Topics like collaborative filtering – memory based approach (since they are just an extended version of content based filtering method), hybrid recommendations and a variety of other algorithms are beyond the scope of this thesis.

1.5 Research Questions

As mentioned before, recommendation systems play a significant role in today's internet. The implementations evaluated in the thesis, the solution provided for Netflix Prize challenge have been considered state-of-the-art techniques. This thesis tries to answer the following questions by building these methods.

- What are the different ways of building Recommendation systems?
- What type of system suits best for what kind of requirement?
- How expensive are these approaches?
- How do we evaluate the effectiveness of the system?
- Is the evaluation of the systems really meaningful?

Chapter 2

Literature Review

Let us start by reviewing previous research in recommendation systems and the existing approaches. Section 2.1 introduces classical recommendation systems, including the very first use of the term collaborative filtering. In Section 2.2, we will have a look at Netflix's recommender system architecture and how they are trying to solve this problem. In Section 2.3, we will look at popularity based method, content based filtering and collaborative filtering. Lastly, in section 2.4, an introduction to how these systems are evaluated is given.

2.1 How it all began

In the early 1990's, Internet was not that predominant. A very few people used online news services or e-mail services. Let us go through some of the pioneering systems that led to the outbreak of the modern day recommender systems.

The first of all systems (experimental implementation) that considered personalization in recommender systems was Tapestry, which was developed by Xerox. The following explain the problem encountered by e-mail users back then that were using the mail services for news article or any document subscription.

i. Mails contained documents and articles which were not relevant to the users.
ii. The above problem could be solved by creating mailing lists. (E.g. Funny articles list, News list etc.) This could help users to subscribe users only to the interested mailing lists.
iii. However, the list of documents of interest for a particular user never usually mapped to created lists. To solve this, filters were implemented using which users could subscribe to documents belonging to any mailing lists but filtering what they need using queries.
iv. This filtering could be improved by using human help by tagging documents as belonging to particular type. So, whenever some users tag the documents as belonging to a particular

type, the users who have subscribed and filtering such documents receive them by mail. Thus came the idea of collaborative filtering.

The above scenarios are depicted as follows.

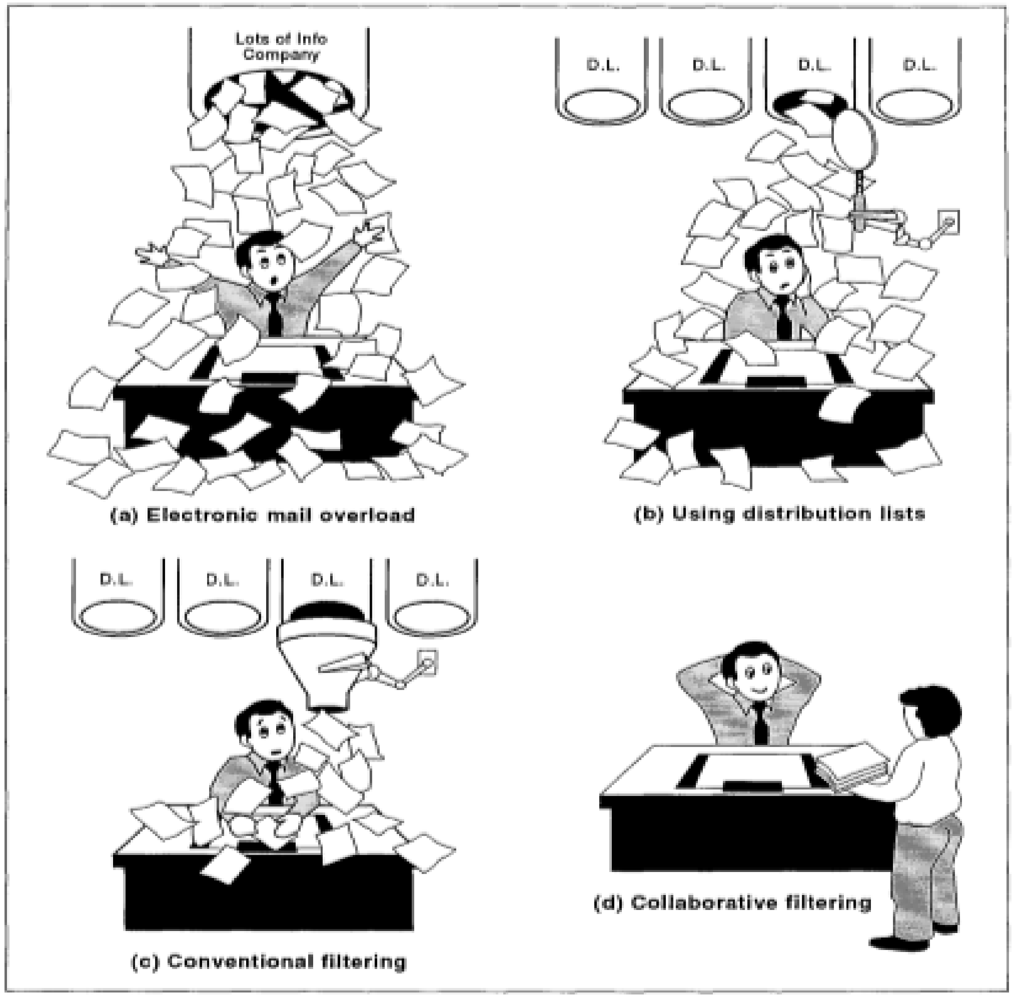

Lots of Info Company
D.L.
D.L.
D.L.
D.L.
(a) Electronic mail overload
(b) Using distribution lists
D.L.
D.L.
D.L.
D.L.
(c) Conventional filtering
(d) Collaborative filtering

Figure 4. Rise of tapestry (a) email document overload (b) distribution lists (c) conventional filtering (d) collaborative filtering [2]

The architecture of the Tapestry system is composed of the following components performing respective functions [2].

- Indexer – Parses documents available online and adds them to the document store. It also parses the documents and generates relevant indexes to help with querying.
- Document store – Stores all the Tapestry documents and their indices.

- Annotation store - Stores annotations or what we currently call “tags” associated with documents.
- Filterer – Repeated task for identifying the related documents based on user filters and adding them to the little box (user queue).
- Little box – A queue for interested documents for every user.
- Remailer – Sends contents of little box to users in periodic interval of time.
- Appraiser – Document personalizer for the ones in the little box. Brings in the concept of prioritization.
- Reader-Browser – Interface to Tapestry system.

The functioning is as discussed before. The Architecture of Tapestry system is as shown below.

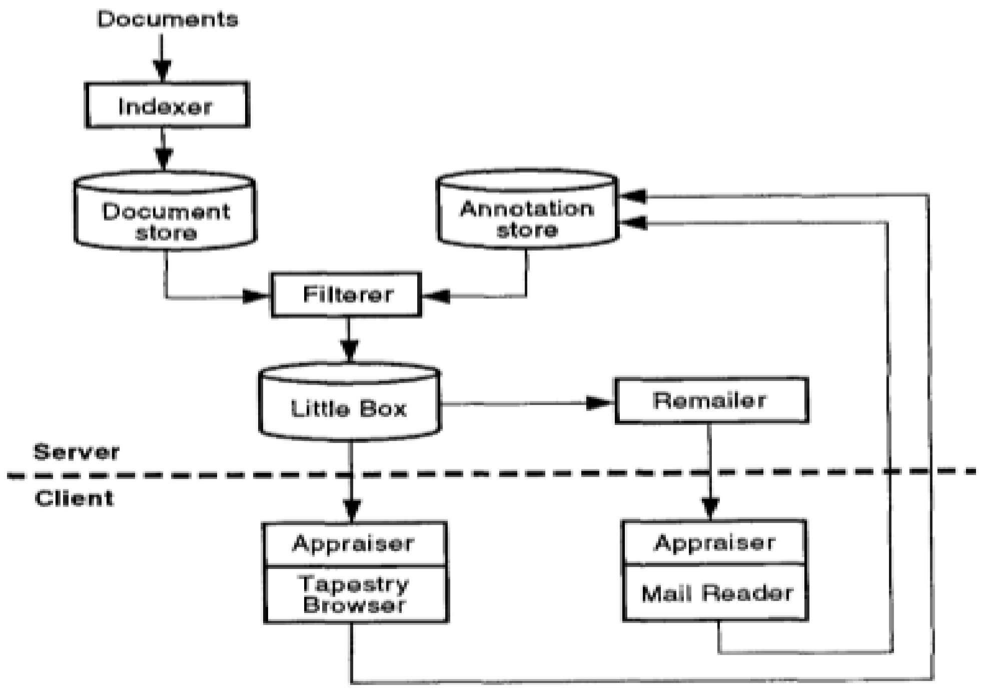


Figure 5. Flow of Documents through Tapestry system [2]

This is regarded as the pioneer of useful information filtering, based on other users’ feedback. This lays down groundwork for the research that follows in the personalized recommender systems field, hence making it one of the very important implementations to look at.

GroupLens [7] was another system based on text service with focus on news. The number of articles at the time had grown rapidly and it was impossible to maintain and filter these documents. Their focus was on an internet discussion forum. GroupLens proposed user-based collaborative filtering, i.e. users with similar tastes. Similarity between users was calculated using Pearson-r correlation. The figure below gives an illustration of their proposed system.

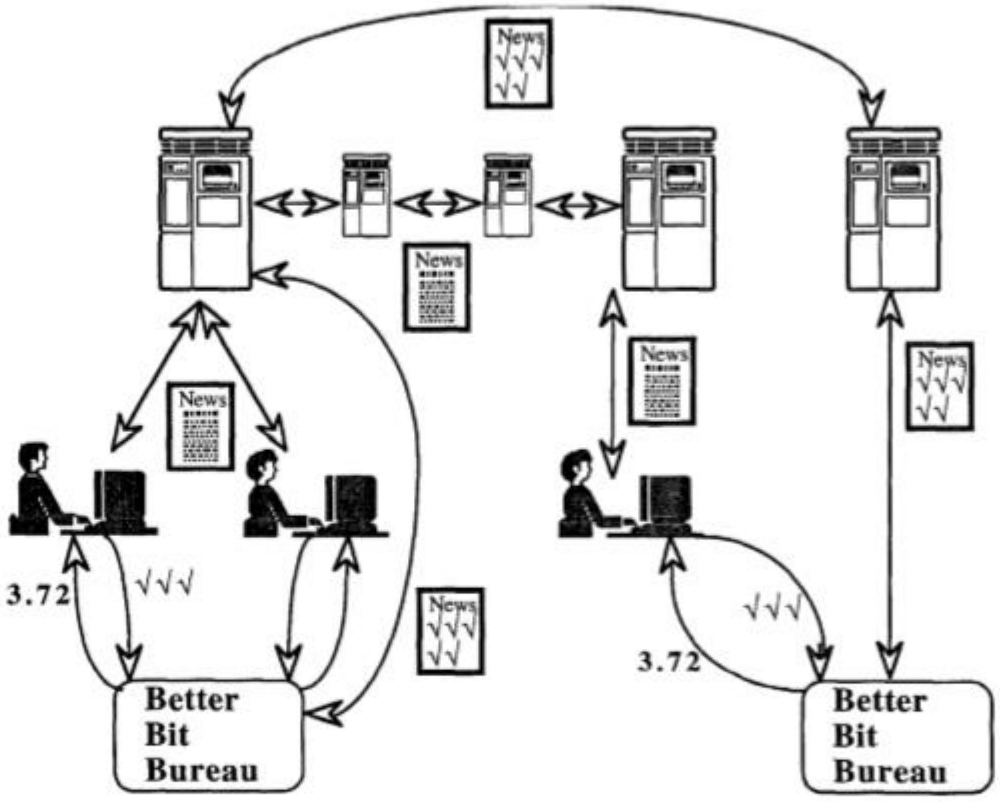


Figure 6. Grouplens Architecture [7]

Now that we have a brief idea on how these systems started, we will look at some of the approaches that are currently being implemented and worked on.

2.2 Recommendation systems today

Recommendation systems play a significant role in majority of the web sites today. Amazon's item-based collaborative filtering (users who bought "X", also bought recommendations) has improved their sales. There are similar improvements to the services and sales when sites like Facebook, Google, Last.fm etc. integrated the recommender systems. To understand recommendation systems today, let us have a look at one of the commercial implementations by Netflix [8]. The architecture of the recommender system used by Netflix is as follows.

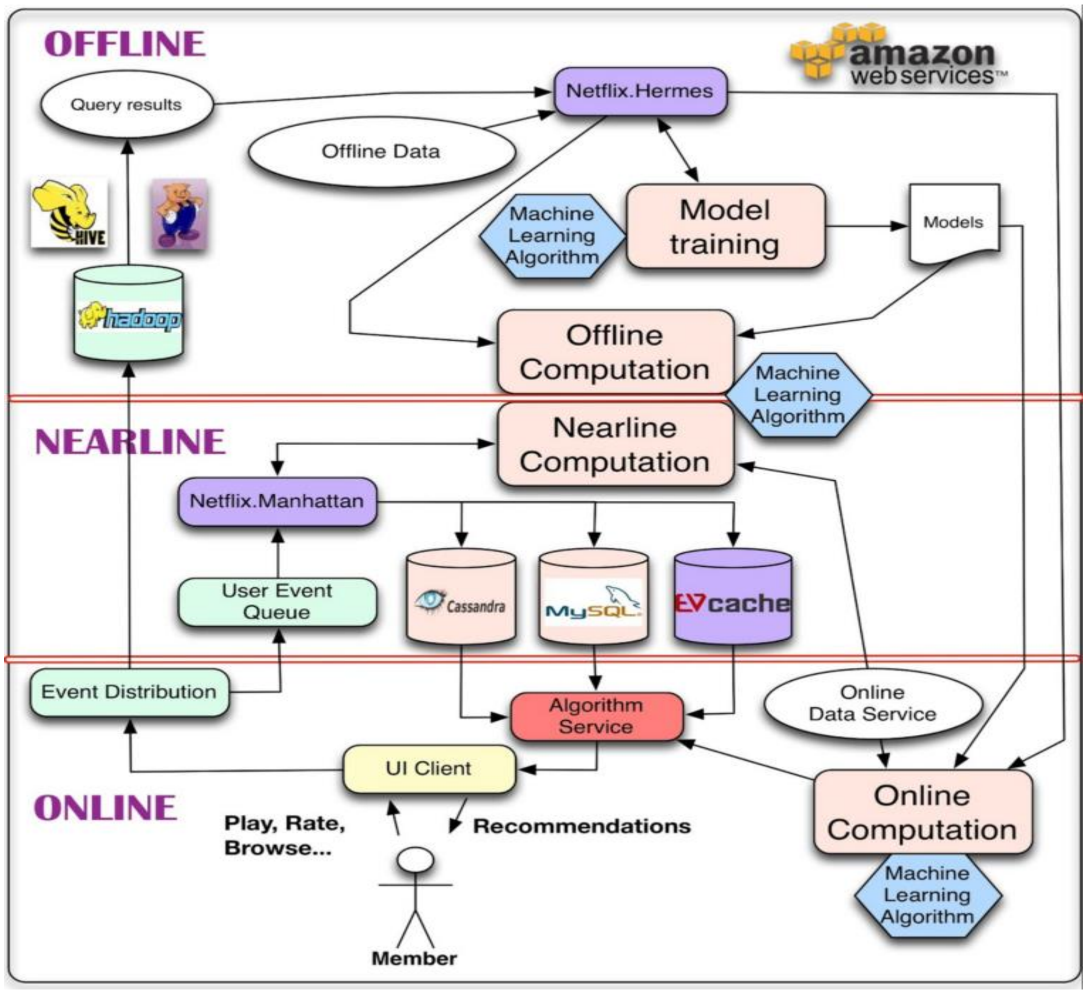


Figure 7. Netflix Recommendation System Architecture [8]

Their architecture mainly focuses on the division of the recommendation tasks to three categories, namely, Offline, Online and Nearline. Offline computation is a process where the recommender model is built on all the available data and is computationally expensive. This does not give results in real time. Online is real-time model that could alter itself to cater to changing views of the users. Nearline is an intermediate model between offline and online, wherein it performs computations like online but there is no real-time response requirement. There is no need to go into the details of every single aspect of the architecture as we are dealing only with the offline data. Most of the recommender system researches try to solve the offline task as they do not actually have access to real-time data or a system. So, let us have a look at the offline task.

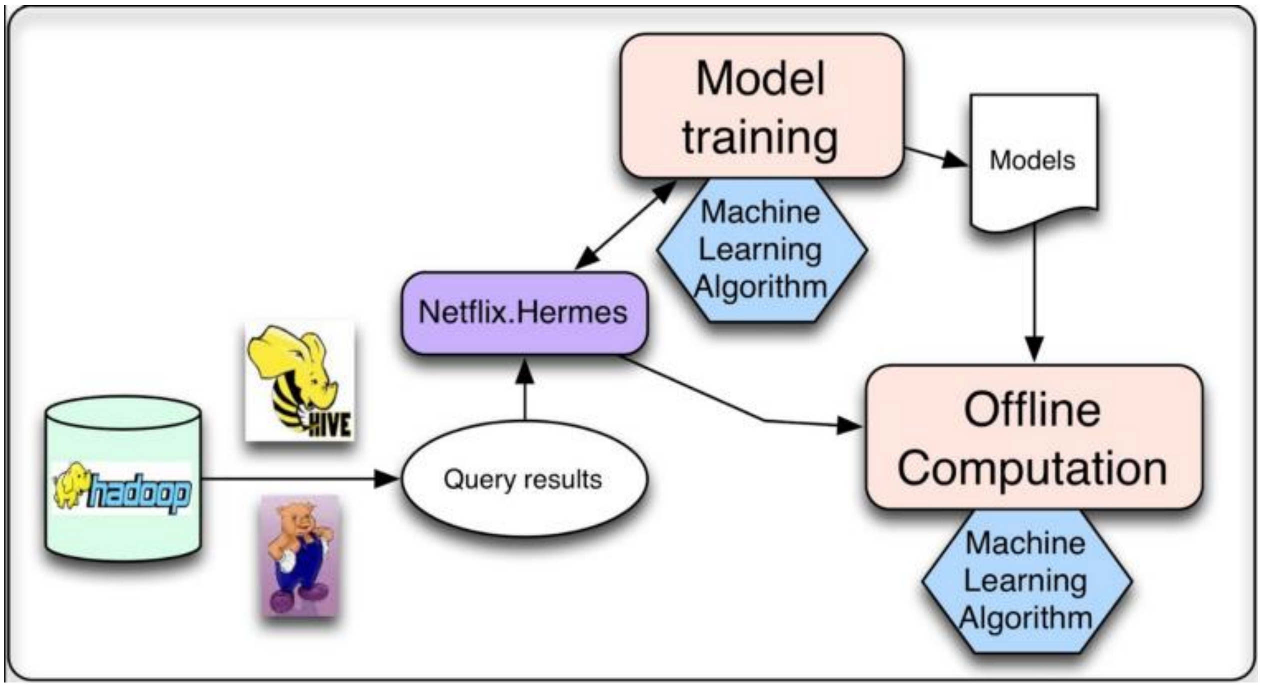


Figure 8. Netflix Recommendation System - Offline Task Architecture [8]

Most of the computations required running personalization machine learning algorithms are carried out offline. This means that their execution is done periodically and they do not have to align with the presentation or results. They divide their offline tasks into two main sub tasks, model training and batch computation of intermediate or final results. In model training, machine learning algorithms are applied on the relevant existing data. This resulting model is stored for later use by the recommendation systems. Batch computation of results is the process in which we use the built model and corresponding input data to estimate results to be used later for online processing or presenting to the user. This provides a brief overview of the recommendation system used by Netflix. It gives a very good understanding on the commercial level application with respect to the required parallelism, storage requirements and handling time and computation complexities.

2.3 State of the art

Given all the research that was triggered by the idea of Tapestry and the Grouplens project, we now have a variety of methods to solve the recommendation problems. Recommendation systems could be divided into the following types based on how they are trying to build the recommendations. The types of recommendation systems are as follows,

- Popularity based recommender systems.
- Content based filtering
- Collaborative filtering
- Hybrid systems

Popularity based system: As Charu C Aggarwal [9] puts forward in his book, popularity based recommendation systems are the most simple implementations that could be used to recommend items to the users. The idea behind this method is very simple. Recommend the most rated items to the users as recommendation. This lacks even the slightest idea of personalization and is the easiest ways to build recommendations. There is not really a way to evaluate these systems.

Content based filtering: As discussed by Parivash et al. [10], this method is mainly reliant on the item description or the user features description. In the case of movielens dataset, each movie is categorized as belonging to a set of genres. If Alice likes a movie “Toy Story”, which belongs to “Animation” and “Children” genres, the system has to identify movies that belong to these genres. The movie similarity is calculated using distance measures like cosine similarity. This is still a good method given the categorization of data on the web.

Collaborative Filtering: This method is based on predicting ratings for unrated movies based on similarities in the ratings between users. E.g. if Alice has rated four movies as 5,5,2,1 and Bob has rated three movies as 5,4,*,1, then Alice and Bob are identified to have similar liking. Then, the unrated third movie’s rating would be estimated to be very close to 2. The major techniques used are to implement this are Matrix Factorization [6, 11] and Autoencoders [12-14].

There are multiple variations of matrix factorization methods that are used. Some of the most common are Singular Value Decomposition (SVD), Non-negative Matrix Factorization and a custom implementation proposed for Netflix Prize challenge [6].

Again, when it comes to Autoencoders, multiple variations like stacked denoising Autoencoders, deep Autoencoders etc. have been undergoing research and are proven to perform better.

Collaborative filtering is known to perform better than the other approaches as there is no manual intervention as to identifying user or item features. The latent features are identified by the system which makes the system still look like a blackbox as these features are not something that we could relate to the ones that we actually see like genres or user's occupation etc. However, these latent features help in improving the performance of the systems.

It is also necessary to understand that recommendation systems could be used to solve either of the two problems. Firstly, to predict the ratings for the unrated items. Secondly, recommending top N items. Some systems use the predicted ratings to carry out the recommendations

Hybrid systems

As it can be seen in the later discussions, all methods have their own advantages and limitations. It is meaningful to use the best of multiple methods and implement them as one system. This could be done by an ensemble of popularity based recommender system and content based system, where the movies with similar genre are recommended, but based on popularity. Also, an ensemble of content based method with collaborative filtering could help in improving the recommendations based on the explicit user profile integration with collaborative filtering. Also, with the cold start problem that collaborative filtering faces, they could be used with popularity based systems to recommend items when there are new users.

2.4 Evaluation

Once we have built the recommender systems, it is important to evaluate how good the system is. There have been many researches in this area, however they are considered to be not very informative given the issues faced by recommender systems. Evaluation could be classified mainly into two categories [15]. Firstly being online evaluation wherein the system is evaluated by verifying the recommendation by actual user. Secondly, we have offline methods where the predictions are evaluated based on the available dataset. In this thesis we consider the offline methods, given that the model is built on offline data and that we do not have a running online system. Some of the well-known offline evaluation metrics are as follows,

RMSE:

For a system that tries to predict the ratings of a user for an unrated item, RMSE is one of the most widely used metrics. Given the list of the movies and the ratings for the movies by the user, the data we work on would be the (user X movie ratings) matrix where each row represents a user, columns represent the movies and each cell value represents the user's rating for that particular movie. For the movies the user has not rated, the value is considered to be zero. RMSE is given by the following formula.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p_i - r_i)^2} \qquad \text{- (1)}$$

Where,

- "n" is the number of movies rated by the user
- "p_i" is the prediction of rating by the user for movie "i"
- "r_i" is the actual rating by the user for movie "i"

RMSE is calculated only on the movies for which the user has actually rated and not all movies. Similar to RMSE, we could also use Mean absolute error or Sum of Squared errors.

Precision and Recall:

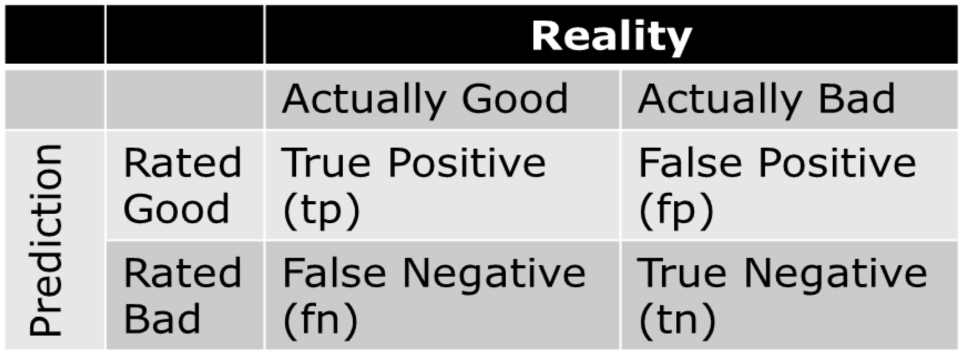
Precision gives the measure of exactness, i.e. determines the fraction of relevant items retrieved out of all items retrieved. In this case, the proportion of recommended movies that are actually good. It is given by the below formula.

$$Precision = \frac{tp}{tp + fp} = \frac{|good\ movies\ recommended|}{|\text{all recommendations}|} \qquad \text{- (2)}$$

Recall gives a measure of completeness, i.e. determines the fraction of relevant items retrieved out of all relevant items. In this case, the proportion of all good movies recommended.

$$Recall = \frac{tp}{tp + fn} = \frac{|good\ movies\ recommended|}{|all\ good\ movies|} \qquad - (3)$$

Where, tp (true positive), tn (true negative), fp (false positive) and fn (false negative) are represented as in the confusion matrix below.



		Reality	
		Actually Good	Actually Bad
Prediction	Rated Good	True Positive (tp)	False Positive (fp)
	Rated Bad	False Negative (fn)	True Negative (tn)

Figure 9. Confusion Matrix

The evaluation metrics suitable for the respective implementation are selected and tested as will be discussed along with the implementation in the later chapter.

Evaluation metrics are not everything. We usually have to go with some hypothesis based on how well it is serving the intended purpose. Let's have a look at why no evaluation metrics actually tells us if the recommender systems are meeting the real-time scenario. Let's say a recommender in a supermarket puts up a sign which says, "Today you should buy bananas and bread". The sales are checked with the recommendation and it is found that this recommender is highly accurate. Another shop just across the street is selling the same amount of bananas and bread. So the recommender at the supermarket is highly accurate but not really good as it did not increase the sales and did not make people reach the whole range of items of the supermarket. Also, it is evident from the human behavior that their tastes keep altering with time. This change makes a huge impact to the recommender system's accuracy as most of the data about the user is irrelevant at the moment. Hence, simple evaluation metrics like RMSE or Precision and Recall could be used for evaluating them on offline data, just to make sure the model is performing well on the available data.

Chapter 3
Theory

In the context of recommender systems, the entities that receive recommendations are termed as users and the elements of interest are termed as items. In this case, the items are the movies from the dataset.

Let us first have a look at the dataset that we are using for this study. The movie ratings (Movielens-1M) dataset made available open-source by the GroupLens project [16] is utilized to build and evaluate the recommender systems. The data set consists of three files, namely movies.dat, users.dat and ratings.dat. The file contents are described as follows,

- Movies.dat
  This file consists of movie's id, movie's name and the genres they belong to
- Users.dat
  This file consists of user's id, gender, age, occupation and their zip-code
- Ratings.dat
  This file consists of user's id, rated movie's id, the rating, timestamp of rating.

On considering all the factors that could be used in building the system, features as required for the type of system are made use of. The data transformation required for the methods will be discussed along with the methods in the following sections.

The theory behind the different types of recommendations will be discussed in the following format,

- Problem Statement
- Sample Data
- Implementation methodology
- Evaluation

3.1 Popularity Based Recommender Systems

3.1.1 Problem Statement

Given a set of users U = {u_1, u_2, ... , u_m} and a set of movies M = { m_1, m_2, ... , m_n } and the ratings by the users for the movies R = { $r_{1,1}$, $r_{1,2}$, ... , $r_{m,n}$}, the system has to identify the top N movies that have received the highest number of ratings.

The first argument about this method is that "Is any movie with the highest number of ratings the most popular?" This assumption could be justified in very simple terms. Let's assume there are 5 movies A, B, C, D and E, and all these movies have ratings from five users. Let's say movies A and B have high ratings (5 on 5) and the other three movies have less ratings (2 or 1 on 5). When other users want to watch movies out of these, it is highly likely that they would either choose from A and B but not C, D and E. Since the number of users who watch the movies A and B are known to increase, it is safe to assume that they both receive high ratings again. This repeats for any other users who want to watch a movie. As it can be seen, the number of ratings for less rated movies is not bound to increase much when compared to the ones rated high. Thus, we can safely assume that the movie which has the highest number of ratings are the most popular movies. Also, it is very highly likely that the users would want to watch the most popular movies.

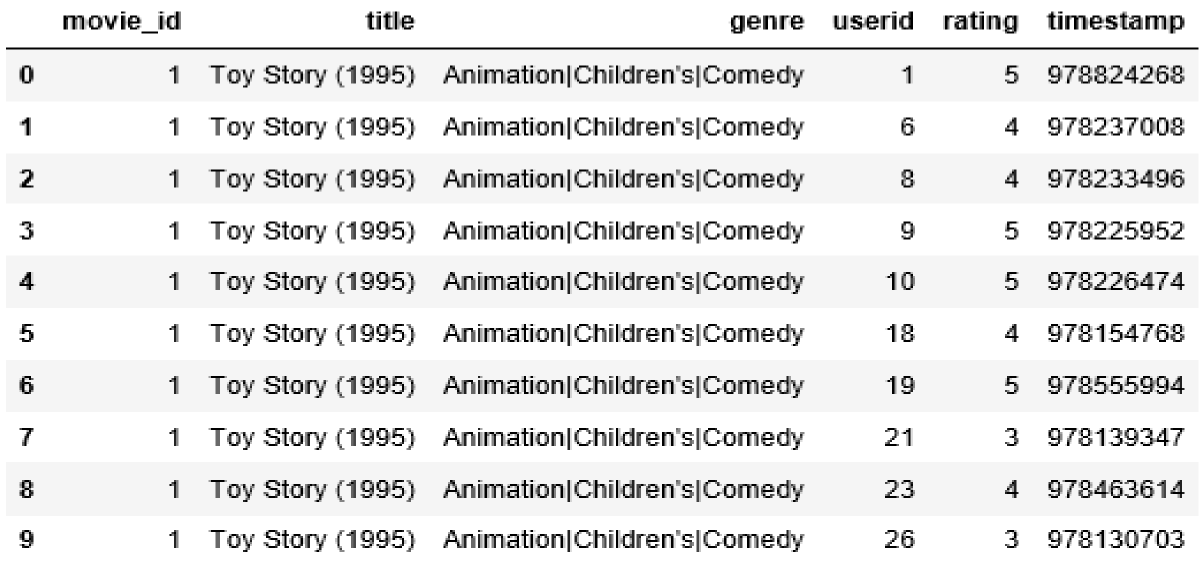
3.1.2 When to use this type of system?

These systems are used when there is no information on the user or item features, or the users are newly joined to the system. Also, these work well with very less computation power and since it is just calculating the number of ratings per item and sorting them, the time required is negligible. Some businesses with no requirement for personalization or with limited resources could use this type of system.

It is worth noting that this could be used in one of the problems with collaborative filtering called the cold start problem by creating a hybrid of collaborative filtering and popularity based method. Cold start problem will be discussed in the collaborative filtering section that follows.

3.1.3 Sample Data

The sample data is as follows. The processed data consists of movie id, user id, rating and the remaining features that are not used in this method.



	movie_id	title	genre	userid	rating	timestamp
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268
1	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008
2	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496
3	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952
4	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474
5	1	Toy Story (1995)	Animation|Children's|Comedy	18	4	978154768
6	1	Toy Story (1995)	Animation|Children's|Comedy	19	5	978555994
7	1	Toy Story (1995)	Animation|Children's|Comedy	21	3	978139347
8	1	Toy Story (1995)	Animation|Children's|Comedy	23	4	978463614
9	1	Toy Story (1995)	Animation|Children's|Comedy	26	3	978130703

Figure 10. Sample data for Popularity based method

3.1.4 Implementation

The implementation of this method consists of the following steps.

- Initially, the data is processed to convert to the format shown above.
- The count of ratings for each movie are calculated.
- This list of movies is then sorted on the basis of the number of ratings.
- Top N of the above generated list are recommended to the users.

3.1.5 Evaluation

This model does not require any evaluation metrics given that it recommends the same items to all users. This may not be the most useful model, but it is quite surprising to know that this model is used by a lot of websites.

3.2 Content Based Filtering

Content based filtering is carried out using the feature description manually created by humans for every single item.

3.2.1 Problem Statement

Given a set of movies M = { m_1, m_2, ... , m_n } and their features (in this case the genres they belong to), a matrix with rows being the movies and the columns being the individual genres with each cell value representing if the movie belongs to the genre by storing a 1 else 0. Given the matrix, it is required to identify top N movies that very closely relate to a selected movie. This could be carried out by manually getting user profile and then generating recommendation, but the idea behind the implementations are the same. The system has to identify items that are very similar to the selected item or user preference for the item properties.

A simple example:

Let us assume we are given a set of movies and the genres they belong to as follows.

	Genre1	Genre2	Genre3	Genre4
Movie1	1	1	0	0
Movie2	1	0	0	1
Movie3	1	1	0	0
Movie4	1	0	0	1
Movie5	1	1	0	0

Now that we know what movies belong to what genre, given that a user likes movies with the following genres [Genre1, Genre2], the system has to recommend the movies Movie1, Movie2 and Movie3.

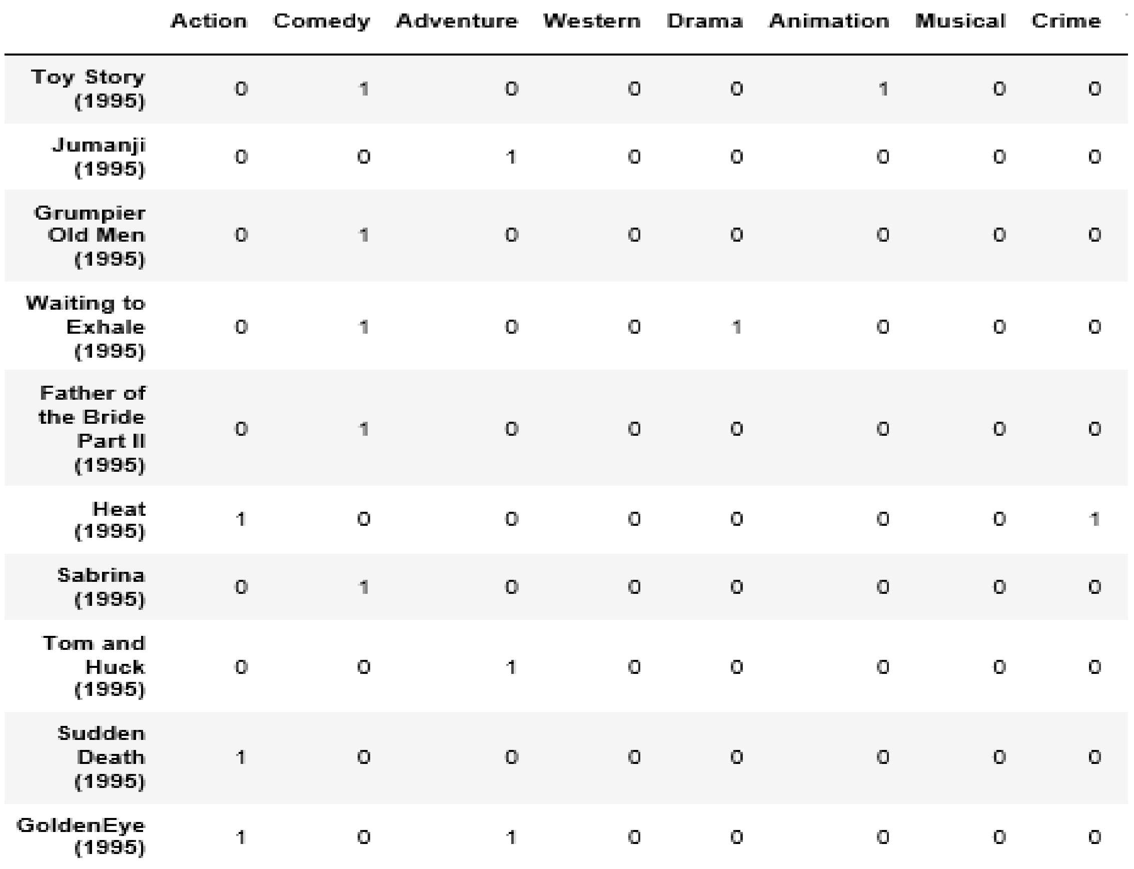
The implementation covering the recommendations based on the items features only and not the user profile is cover as part of this thesis.

3.2.2 When to use this type of system?

This system could be used when the features of all items are known and the user profile is known. This is not very expensive computationally or in terms of time consumption. Given the categorization in web today, these systems perform really well in coming up with recommendations.

3.2.3 Sample Data

The sample data is as follows. The processed data is a matrix of (Movies X Genres).



	Action	Comedy	Adventure	Western	Drama	Animation	Musical	Crime
Toy Story (1995)	0	1	0	0	0	1	0	0
Jumanji (1995)	0	0	1	0	0	0	0	0
Grumpier Old Men (1995)	0	1	0	0	0	0	0	0
Waiting to Exhale (1995)	0	1	0	0	1	0	0	0
Father of the Bride Part II (1995)	0	1	0	0	0	0	0	0
Heat (1995)	1	0	0	0	0	0	0	1
Sabrina (1995)	0	1	0	0	0	0	0	0
Tom and Huck (1995)	0	0	1	0	0	0	0	0
Sudden Death (1995)	1	0	0	0	0	0	0	0
GoldenEye (1995)	1	0	1	0	0	0	0	0

Figure 11. Sample Data Subset for Content Based Filtering

3.2.4 Implementation

The implementation methodology consists of the following steps.

- The data is processed and the matrix as in the sample data is generated
- This matrix is then used in building a nearest neighbour model which is a lazy learning method.
- Once we give the item features as input, it comes up with k similar items as recommendations

K-nearest neighbor learning

K-nearest neighbor method is one of the simplest and the most widely used methods used with a distance function. The algorithm does not build the model but it memorizes the data and tries to identify the items with the most similar features.

In our case, the model is fed with the (Movie X Genre) matrix, where each row is a sample. The algorithm that is used is out-of-the-box implementation from Python's sci-kit-learn which uses the selected distance measure and gives out the k nearest items. This is something very similar to Vector Space Model that is used in document retrieval, with the difference being that the columns here are not words in the documents but the genres of the movies. Although this algorithm is mainly used for classification as shown below, it could also be used to identify the similar items as discussed before.

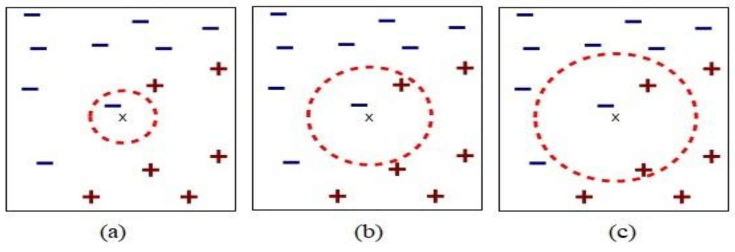


Figure 12. K-Nearest Neighbor algorithm for classification
(a) 1-nearest neighbor (2) 2-nearest neighbor (3) 3-nearest neighbor

3.2.5 Evaluation

The system could be evaluated by calculating average Minkowski distance between the recommended movies and the input movie based on their genre. Minkowski distance between X = (x_1, x_2, ... ,x_n) and Y = (y_1, y_2, ... , y_n) ϵ $\mathbb{R}^n$ of order p is given by,

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \qquad - (4)$$

3.3 Collaborative Filtering

The underlying idea of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person.

3.3.1 Problem Statement

The problem statement for Collaborative filtering could be defined as follows. The problem can be modeled by the triplet (U, I, R), where,

- U is the set of users
- I is the set of items (movies) in our case
- R is the set of ratings for movies by the users. (values ranging from 1-5 for rated movies and 0 for unrated movies)

A realization of (U, I, R) denoted by (u, i, r) means that user u rated item i with value r. The goal of this system is to estimate R from (U,I) such that the RMSE is minimum.

Intuition

Let us consider the below example to understand how collaborative filtering works.

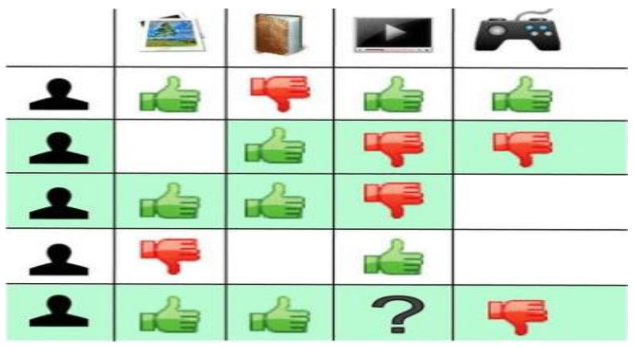


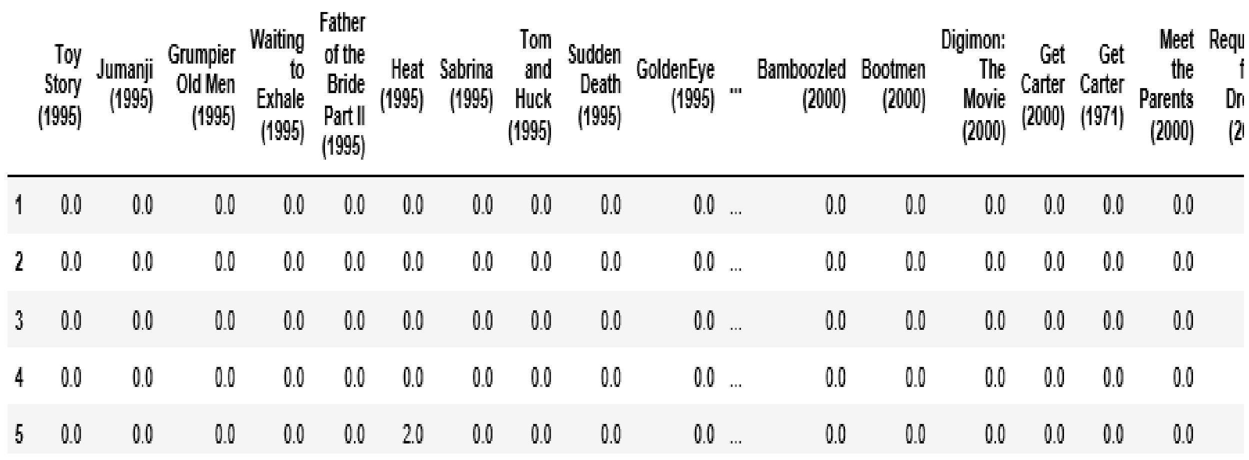
Figure 13. Collaborative Filtering Intuition

There are five users who have shared their opinion on the activities like photography, reading, watching movies and playing video games. Collaborative filtering's basic ideology is that users with similar tastes would like similar items. Now, if we are trying to identify if the fifth user likes watching movies, we will have to identify users similar to him. It can be seen that users 2 and 3 seem to be very similar to user 5. Now that we know similar users, looking at their response to watching movies, it can be said that the user 5 is highly unlikely to be interested in the activity.

3.3.2 When to use this type of system?

This type of system is known to be the best performing for predicting the ratings. This performs well in all conditions as they are not reliant on any explicit feature descriptions. Also, personalization is the built in feature of these systems. However, this cannot be used when there is not much data about the user or the item ratings i.e. new user or new item. This is called the cold start problem.

3.3.3 Sample Data



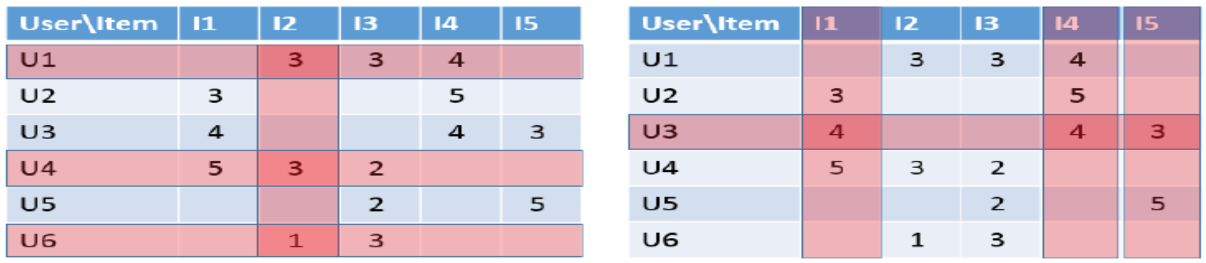
	Toy Story (1995)	Jumanji (1995)	Grumpier Old Men (1995)	Waiting to Exhale (1995)	Father of the Bride Part II (1995)	Heat (1995)	Sabrina (1995)	Tom and Huck (1995)	Sudden Death (1995)	GoldenEye (1995)	...	Bamboozled (2000)	Bootmen (2000)	Digimon: The Movie (2000)	Get Carter (2000)	Get Carter (1971)	Meet the Parents (2000)	Requ f Dr (2
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
5	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	

Figure 14. Sample Data for Collaborative in general

Collaborative filtering methods can be classified into two main categories, namely, Memory based method and Model based method.

3.3.4 Memory based method

This method uses user-item rating matrix to compute the similarity between users or items. Then the unknown ratings are estimated. Following which, the predicted ratings could be used for top N item recommendations. Due to the simplicity and effectiveness, this method was used in many commercial applications. Typical examples of this approach are neighborhood-based CF to provide item-based/user-based top-N recommendations. This method could further be divided into two categories based on user or item similarity. This is discussed in the sections that follow.



User\Item	I1	I2	I3	I4	I5
U1		3	3	4	
U2	3			5	
U3	4			4	3
U4	5	3	2		
U5			2		5
U6		1	3		

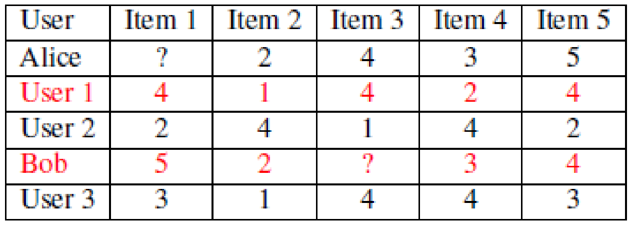
User\Item	I1	I2	I3	I4	I5
U1		3	3	4	
U2	3			5	
U3	4			4	3
U4	5	3	2		
U5			2		5
U6		1	3		

Figure 15. User-based (a) and Item-based (b) collaborative filtering

3.3.4.1 User based collaborative filtering

We have a (User X Movie) rating matrix, and we are required to predict the rating of an unrated item i by user u. In user-based collaborative filtering, firstly we identify neighbors of the user u. We can use similarity measures like cosine similarity or minkowski distance measure. By calculating similarity between “u” and all other users, we can select top k similar users to u. To predict the rating $M_{u,i}$, we use ratings of these k similar users for the item i. We then calculate weighted average of their ratings. This is as depicted in the Figure 12 (a).

Let us have a look at a simple example.



User	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	?	2	4	3	5
User 1	4	1	4	2	4
User 2	2	4	1	4	2
Bob	5	2	?	3	4
User 3	3	1	4	4	3

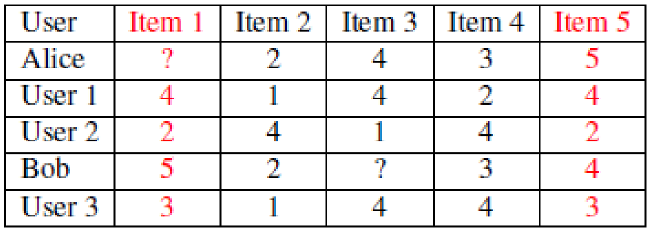
Figure 16. User Based Collaborative Filtering Example

In the table above, let us try to estimate how much Bob would like Item 3. So the “k” most similar users to Bob are selected and their ratings of Item 3 are noted. Based on these ratings, it is possible to predict Bob’s rating for Item 3. To keep things simple for the purpose of explanation, let us look for “k=1” similar user. The table shows that User 1’s ratings are very close to Bob’s. The Items Bob rated high were rated high by User 1 and the ones rated low were again rated low by User 1. These two users could be considered as the closest in this case. User 1 has rated Item 3 as 4. Hence it could be estimated that Bob would rate the item very close to 4. In actual scenario, the value for “k” is usually large and the ratings are calculated using weighted average.

3.3.4.2 Item based collaborative filtering

This method is very similar to User based collaborative filtering. To determine the rating of item i by user u, we identify the neighbors of the item i, instead of user u. By calculating similarity between i and all other items, we can select k most similar items to i, just as we did with users in user-based method. Then we can predict the rating $M_{u,i}$ using weighted average. This is as shown in figure 12 (b)

Let us have a look at a very simple example.



User	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	?	2	4	3	5
User 1	4	1	4	2	4
User 2	2	4	1	4	2
Bob	5	2	?	3	4
User 3	3	1	4	4	3

Figure 17. Item Based Collaborative Filtering Example

In the above table, we will try to estimate how much Alice may like Item 1. Firstly, we have to choose top "k" items similar to Item 1 and ratings for these items by all users are noted. Based on this information, the rating by Alice could be estimated. For simplicity purpose, let us consider "k=1" similar items. From the table, it can be seen that, Item 1 is similar to Item 5 based on ratings by all users. Then, for Alice, it can be estimated that the rating for Item 1 is very close to 5. In actual scenario, the value for "k" is considerable large and just like User based method, weighted average is calculated to estimate the rating.

Challenges faced by collaborative filtering method

- Cold Start problem
- Data sparsity
- Scalability

Cold Start problem

The inability of the system to perform the desired operation due to insufficient data in the case of recommendation systems is called cold start problem. This could be tackled by using popularity based method or content based filtering along with collaborative filtering.

Data Sparsity

The number of items available on most of the famous web sites is huge. Even the most active consumer would have been able to rate or review very few items. Considering this fact, it can be said that even the most popular items would have very less number of ratings.

Scalability

Most of the web sites have enormous number of users and items. With millions of users and items it is computationally very expensive, thus when the number of users and items keep increasing, it is very hard to compute recommendations. We will have a look at this issue in Matrix factorization implementation and look at how the proposed algorithm [6] fails to estimate the rating on the actual dataset but performs exceptionally well on a very small subset of the dataset.

3.3.5 Model Based method

In this method, estimation models are developed using machine learning algorithms to predict users' ratings for unrated items. Many algorithms could be used for this method. Some examples are Bayesian networks, clustering models, Matrix factorization etc.

As part of this thesis, memory based collaborative filtering was not implemented as they follow the idea behind content based filtering (i.e. based on neighborhood approach). Only model based approaches (i.e. Matrix Factorization and Autoencoder method) were carried out. Matrix factorization and Autoencoder methods will be discussed in the section that follows.

3.3.5.1 Implementation

Many neighborhood based approaches could be used to solve this problem. However, the scope of this thesis is on Matrix Factorization and Auto-Encoder methods.

3.3.5.2 Matrix Factorization

Matrix factorization is one of the most widely used techniques for collaborative filtering problems. The idea behind matrix factorization techniques is very simple. We try to approximately estimate the ratings Matrix using two low rank matrices P and Q. This is something similar to identifying the factors of a large number (e.g. Factors of 200000 are 400 and 500)

$$R \approx PQ \qquad - (5)$$

Where, P is an (N X K) and Q is a (K X M) matrix (for N users and M movies). This factorization gives low rank representation of users and items. It means that, P could represent the hidden features of the users and Q could represent the hidden features of the items or movies.

As stated by Gábor Takács et al. [6] "Given problem statement, the unrated ratings of R cannot be represented by zero. For this case, the approximation task can be defined as follows. Let p_{uk} denote the elements of $P \in \mathbb{R}^{NxK}$, and q_{ki} the elements of $Q \in \mathbb{R}^{KxM}$. Let p_u^T denote the transpose of the u-th row of P, and q_i, the i-th column of Q. Then:

$$\hat{r}_{ui} = \sum_{k=1}^{K} p_{uk} q_{ki} = \mathbf{p}_u^T \mathbf{q}_i \qquad \text{- (6)}$$

$$e_{ui} = r_{ui} - \hat{r}_{ui} \quad \text{for } (u,i) \in \mathcal{T} \qquad \text{- (7)}$$

$$\text{SSE} = \sum_{(u,i)\in\mathcal{T}} e_{ui}^2 = \sum_{(u,i)\in\mathcal{T}} \left(r_{ui} - \sum_{k=1}^{K} p_{uk} q_{ki} \right)^2 \qquad \text{- (8)}$$

$$\text{RMSE} = \sqrt{\text{SSE}/|\mathcal{T}|} \qquad \text{- (9)}$$

$$(\mathbf{P}^*, \mathbf{Q}^*) = \underset{(\mathbf{P},\mathbf{Q})}{\arg\min}\, \text{SSE} = \underset{(\mathbf{P},\mathbf{Q})}{\arg\min}\, \text{RMSE} \qquad \text{- (10)}$$

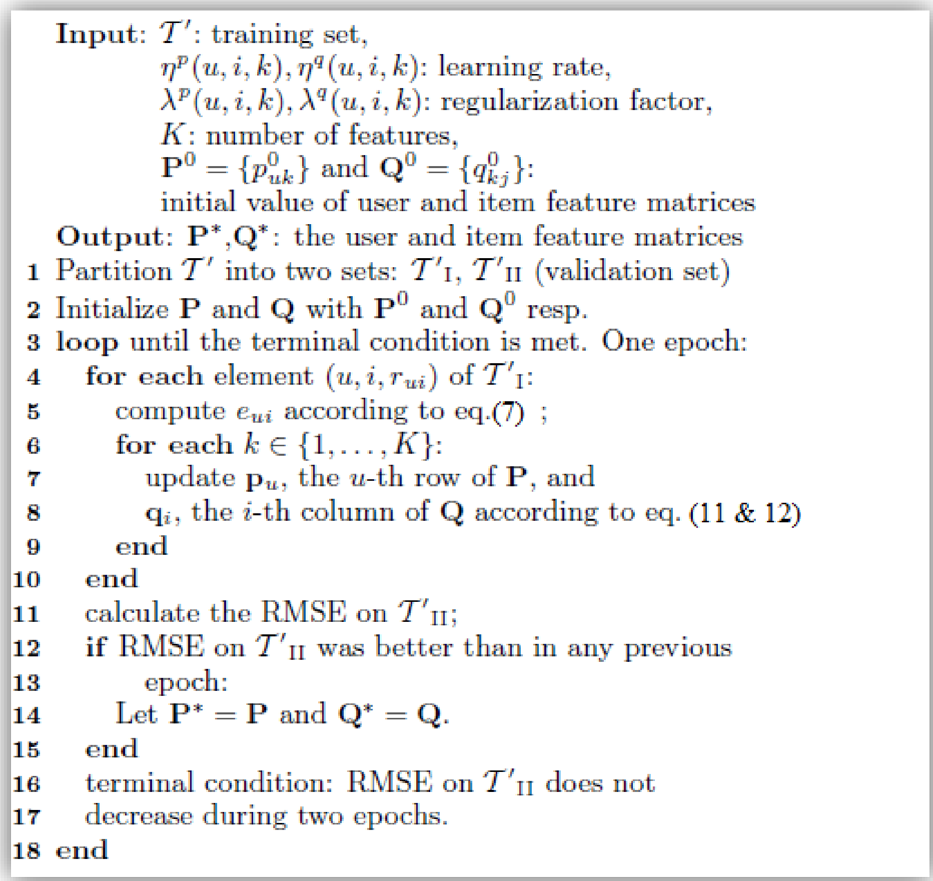
Here, $\hat{r}_{ui}$ denotes how the u^th user rates the i^th item, according to the model, e_{ui} denotes the training error measured at the (u,i)^th rating, SSE denotes the sum of squared training errors. It should be noted that the RMSE or SSE are minimized only on the known ratings."

Their presented MF has 4 important parameters η (learning rate), λ (regularization factor) and the initial values of P and Q, denoted by P^0 and Q^0 respectively. The weights are updated using the below equation.

$$p'_{uk} = p_{uk} + \eta^p(u,i,k) \cdot (e_{ui} \cdot q_{ki} - \lambda^p(u,i,k) \cdot p_{uk}) \qquad \text{- (11)}$$

$$q'_{ki} = q_{ki} + \eta^q(u,i,k) \cdot (e_{ui} \cdot p_{uk} - \lambda^q(u,i,k) \cdot q_{ki}) \qquad \text{- (12)}$$

The algorithm is very simple, in which the gradient descent approach is used to reduce the error. Initially P and Q are initialized to random matrices of the given order. Then, every iteration, for each user, the error is reduced using learning rate and normalization factor as any other gradient descent algorithm. The exception being error reduction is only on the known ratings. The algorithm presented by Gábor Takács et al. [6] is as follows,



```
Input: $\mathcal{T}'$: training set,
       $\eta^p(u,i,k), \eta^q(u,i,k)$: learning rate,
       $\lambda^p(u,i,k), \lambda^q(u,i,k)$: regularization factor,
       $K$: number of features,
       $\mathbf{P}^0 = \{p^0_{uk}\}$ and $\mathbf{Q}^0 = \{q^0_{kj}\}$:
       initial value of user and item feature matrices
Output: $\mathbf{P}^*$,$\mathbf{Q}^*$: the user and item feature matrices
Partition $\mathcal{T}'$ into two sets: $\mathcal{T}'_{\text{I}}$, $\mathcal{T}'_{\text{II}}$ (validation set)
Initialize $\mathbf{P}$ and $\mathbf{Q}$ with $\mathbf{P}^0$ and $\mathbf{Q}^0$ resp.
loop until the terminal condition is met. One epoch:
  for each element $(u, i, r_{ui})$ of $\mathcal{T}'_{\text{I}}$:
    compute $e_{ui}$ according to eq.(7) ;
    for each $k \in \{1, \dots, K\}$:
      update $\mathbf{p}_u$, the $u$-th row of $\mathbf{P}$, and
      $\mathbf{q}_i$, the $i$-th column of $\mathbf{Q}$ according to eq. (11 & 12)
    end
  end
  calculate the RMSE on $\mathcal{T}'_{\text{II}}$;
  if RMSE on $\mathcal{T}'_{\text{II}}$ was better than in any previous
      epoch:
    Let $\mathbf{P}^* = \mathbf{P}$ and $\mathbf{Q}^* = \mathbf{Q}$.
  end
  terminal condition: RMSE on $\mathcal{T}'_{\text{II}}$ does not
  decrease during two epochs.
end
```

Figure 18. Matrix Factorization Algorithm [6]

3.3.5.3 Matrix Factorization - Evaluation

The performance of the model is evaluated using RMSE measure on the known ratings. The dataset is first divided into train and test sets. Then the ratings matrices for train and test are created. The training set matrix is used to build the model, i.e. to estimate the low rank matrices. Then the estimated R matrix is checked with the Test set matrix and the RMSE is calculated.

3.3.5.4 AutoEncoders

Auto Encoders are a special type of deep neural networks mainly used for unsupervised learning. The main goal of an autoencoder is to learn a representation (encoding) for given input data. The architecture comprises of encoding layers and decoding layers. The output layer consists of the same number of nodes as the input layer. The job of Autoencoders is to reproduce the input values at the output, thereby learning latent features in the data. The encoding layers and decoding layers could be comprised of multiple layers as required for the application. A typical architecture is as follows.

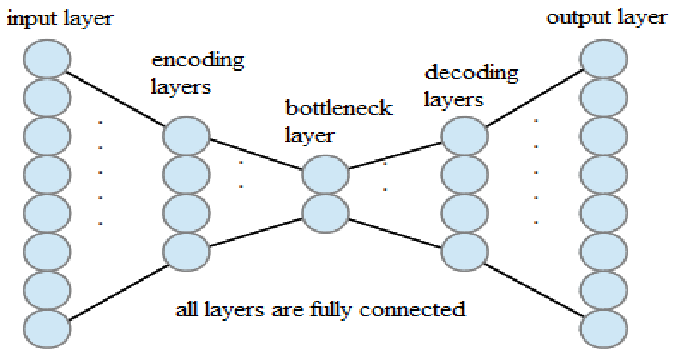


Figure 19. AutoEncoders – Typical Architecture

To explain mathematics behind autoencoders in a very simple terms, let us consider the architecturally simplest model with one hidden layer (i.e. only the bottleneck layer). It is the simplest extension of multilayer perceptron network with same number of input and output nodes and a hidden layer. Each layer has an additional bias node except for the output layer and each layer is fully connected to the next layer with weights and activation function. The purpose is ot reconstruct the input. Autoencoders consist of encoder part followed by the decoder part, which can be defined as transitions ϕ and ψ respectively such that:

$$\phi : \mathcal{X} \rightarrow \mathcal{F} \qquad \text{- (13)}$$

$$\psi : \mathcal{F} \rightarrow \mathcal{X} \qquad \text{- (14)}$$

$$\phi, \psi = \arg\min_{\phi,\psi} \|X - (\psi \circ \phi)X\|^2 \qquad \text{- (15)}$$

In simple case, encoding layer maps the input x (input) to z (encoding). This is done as follows

$$z = \sigma (Wx + b) \qquad - (16)$$

where, σ is the activation function like sigmoid or rectified linear unit, W is the weight matrix and b is the bias.

In the decoding layer, it maps z (encoded) to x`(reconstructed input). This is done as follows.

$$x` = \sigma`(W`z + b`) \qquad - (17)$$

where, σ`, W`, b` are the activation function, weight matrix and bias for the decoder. The weights and bias are optimized using backpropagation algorithms, in which the forward step tries to map input to output and the backward step propagates the error in the estimation thus updating the weights and the biases as necessary. A combination of these two steps are considered as one epoch.

Given the property of autoencoders, they could be used similar to matrix factorization to extract the latent features of users and movies. The latent features could be identified to be extracted at the encoding and the decoding layers. The architecture above shows a bottleneck layer, which is one of the main components used in the case of dimensionality reduction. This helps in encoding images or audio signals without much loss.

The architecture used as part of this thesis for collaborative filtering consists of three hidden layers with the first layer acting as encoding layer and the second layer as bottleneck layer and the third layer as the decoding layer. The main goal is to find the weights of encoding and decoding layers to reproduce the input.

The input to this algorithm is the ratings matrix, where the model tries to reconstruct the ratings given by the users. The usual train and test split of the data is carried out before the training to check the performance of the model. Autoencoders use backpropagation algorithm and update the weights to optimize the loss. The loss considered here to optimize is the RMSE error on the reconstruction of the input for the rated movies only.

3.3.5.5 Autoencoders Evaluation

The built model is evaluated using RMSE on the known ratings from the test set. i.e. The model is evaluated on how well the input was reconstructed only on the basis of known ratings.

Predicted ratings in collaborative filtering methods in general are then used to come up with recommendations. The movies are sorted in the descending order of the predicted ratings by the user. Then the movies that are already rated by the user are discarded. Following which, the top N movies are recommended to the user. When implemented in a real-time system, these could be evaluated by the users online and the results of which could be used to improve the models.

The next chapter talks about some analysis on the used movielens dataset and goes through the implementation details along with the results.

Chapter 4

Implementation and Results

With a brief understanding of the methods implemented in this thesis, let us have a look at what the dataset consists of and the meaningful insights in the data first before going into the implementation and results details of the methods.

4.1 Data Analysis

The dataset used is the movielens dataset with approximately 1 million ratings (in the range 1-5) by a total of 6040 users for 3883 movies. The data would be used as a matrix of (Users X Movies) with cell values being user's rating for the movie and 0 for unrated. The sparsity of the data (the number of cells unrated in the {user X movie-ratings} matrix) was found to be 95.74%. i.e. We only have 4.26% of cells filled.

The movies in the dataset are categorized into 18 genres. The dataset provides information as to what genres the movies belong to. A simple word-cloud representation of genres is as follows.

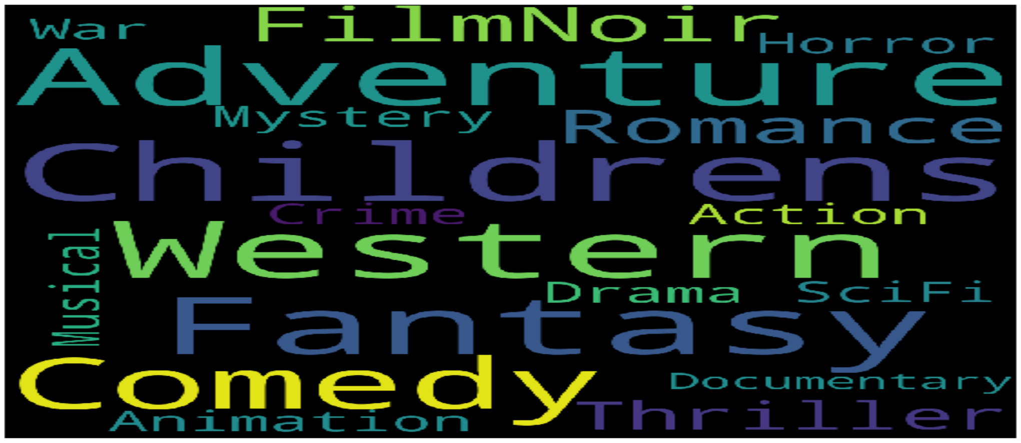


Figure 20. Movie Genre Word Cloud representation

There are 3883 movies in the dataset. A simple word-cloud representation is as follows.

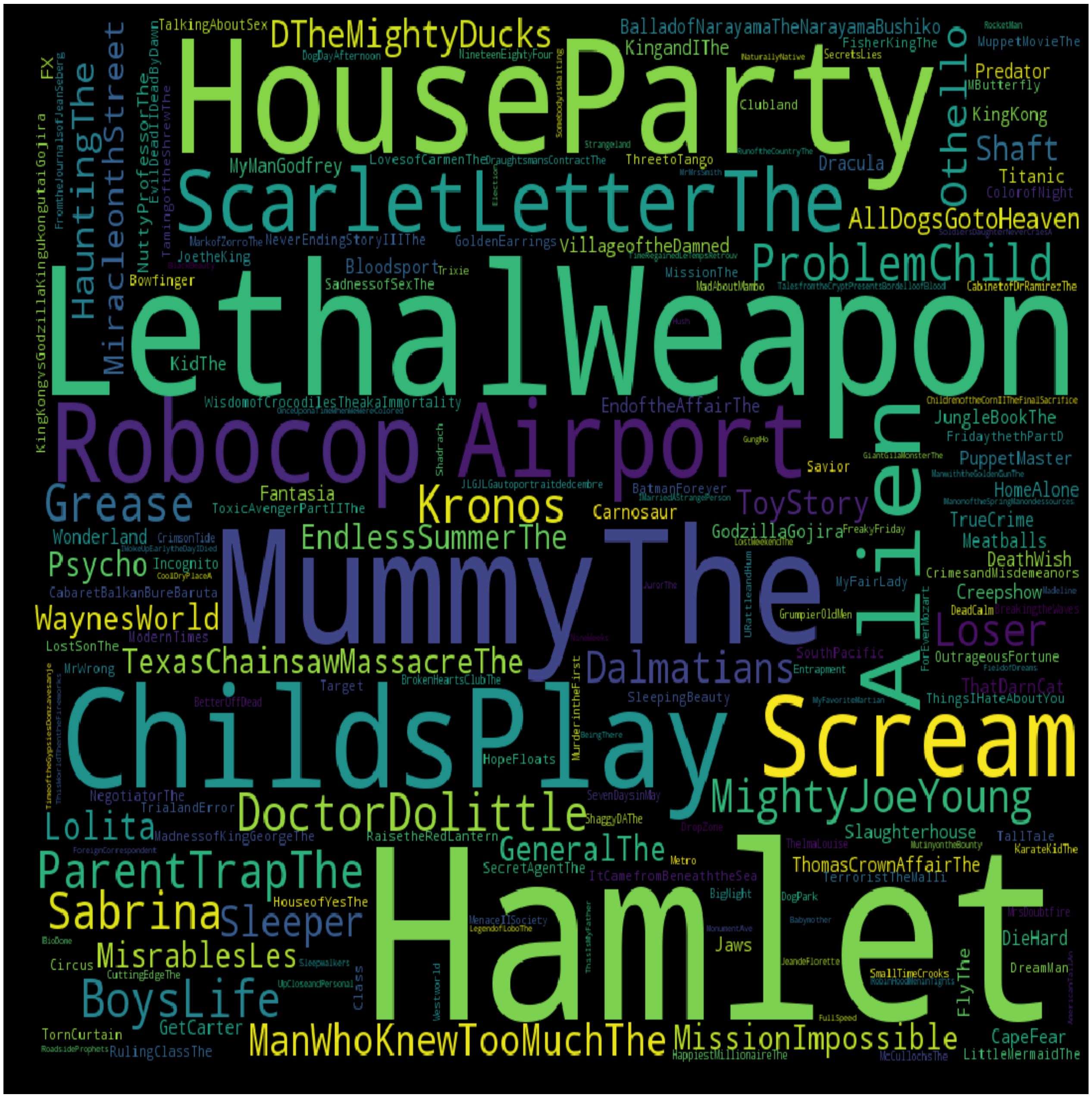


Figure 21. Movies - Word Cloud

Let us now have a look at the number of movies per genre. Comedy and Drama have a reasonably higher number of movies in the dataset.

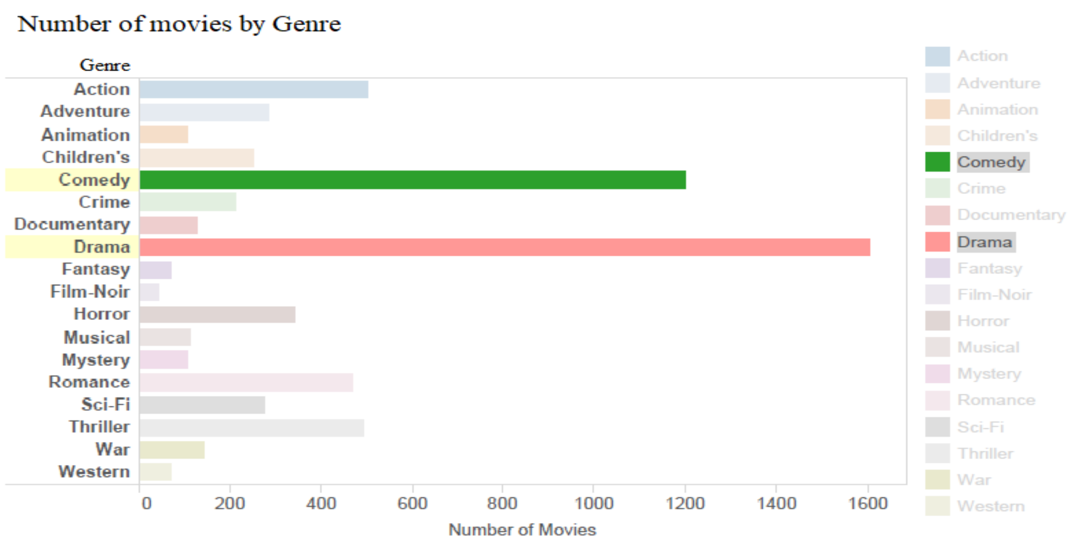


Figure 22. Number of Movies by Genre [17]

The movie titles also have the year they were released in. A graph showing the number of movies per genre yearly is shown below. It can be observed that, there are relatively higher number of movies that were released in the 1990's,

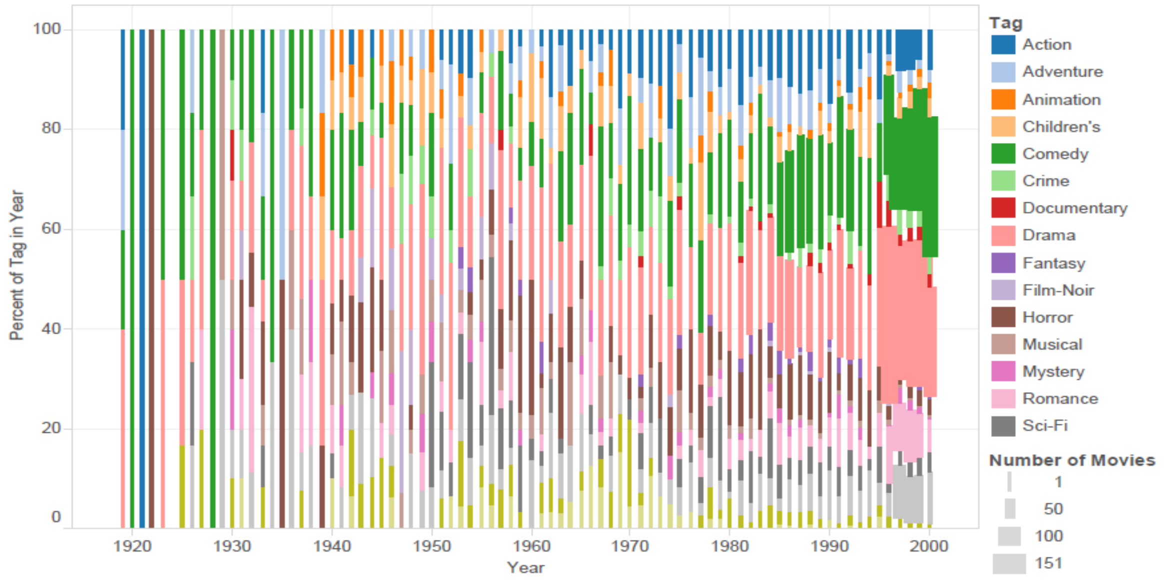


Figure 23. Genres by Year [17]

Also, from the data it was seen that, the average ratings for the movies yearly has a negative slope as shown below, which means that the users are more picky about the movies they watch.

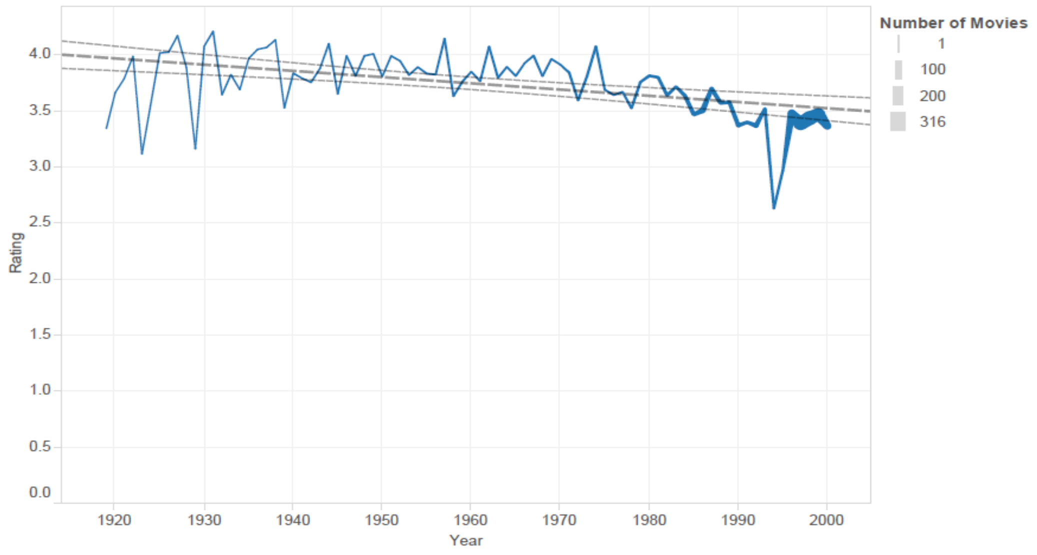


Figure 24. Average ratings by year [17]

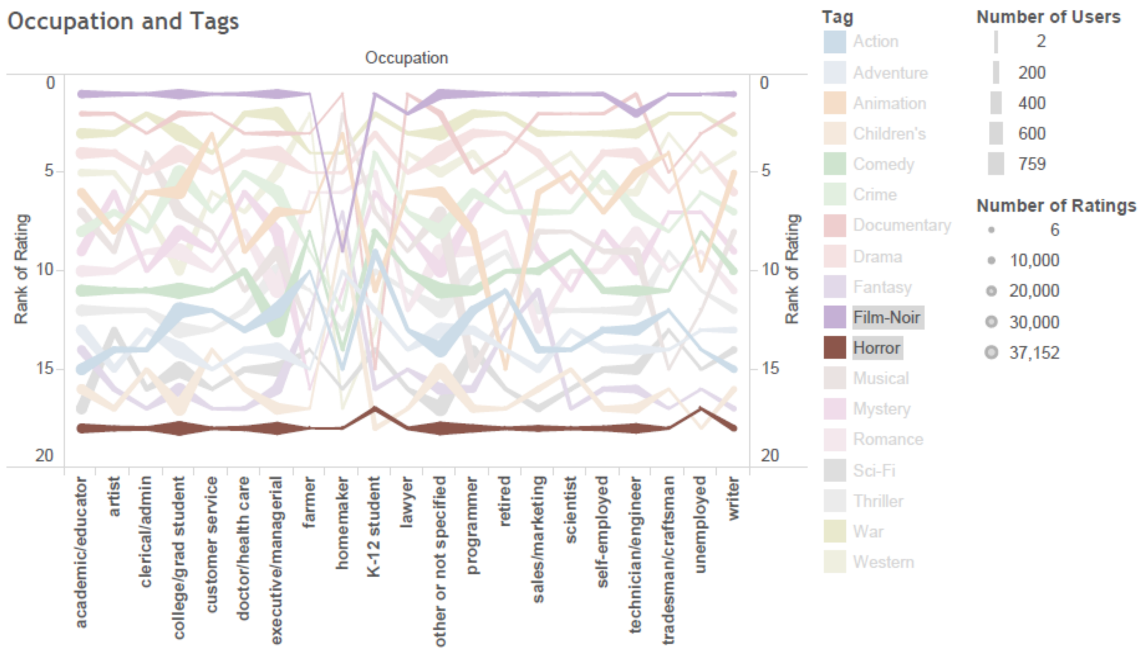


Figure 25. Occupation vs Genre [17]

From the figure above, it can be seen that horror is the least preferred and Film-noir is the most preferred genre across all occupations of the users.

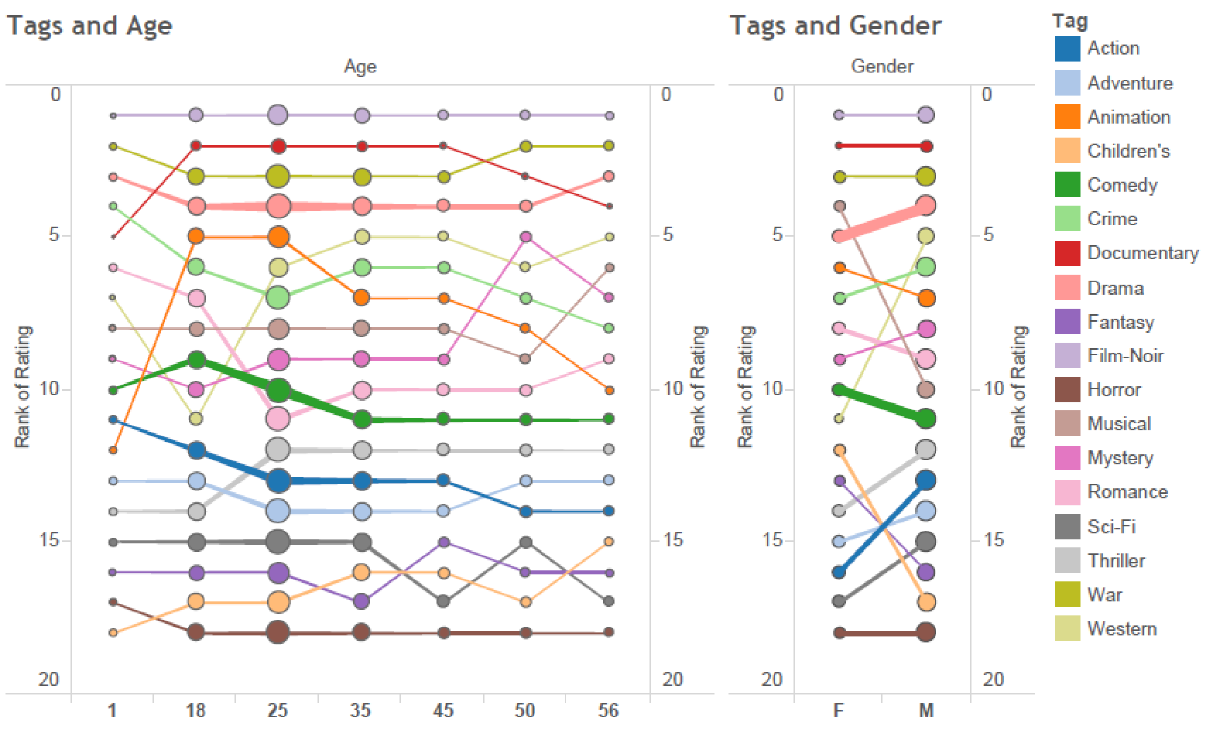


Figure 26. Age vs Genre with gender information [17]

From the above figure, it can be seen that the liking for action movies goes down with age. Also, action is liked most by males when compared to females and comedy, romance are liked by females more than males.

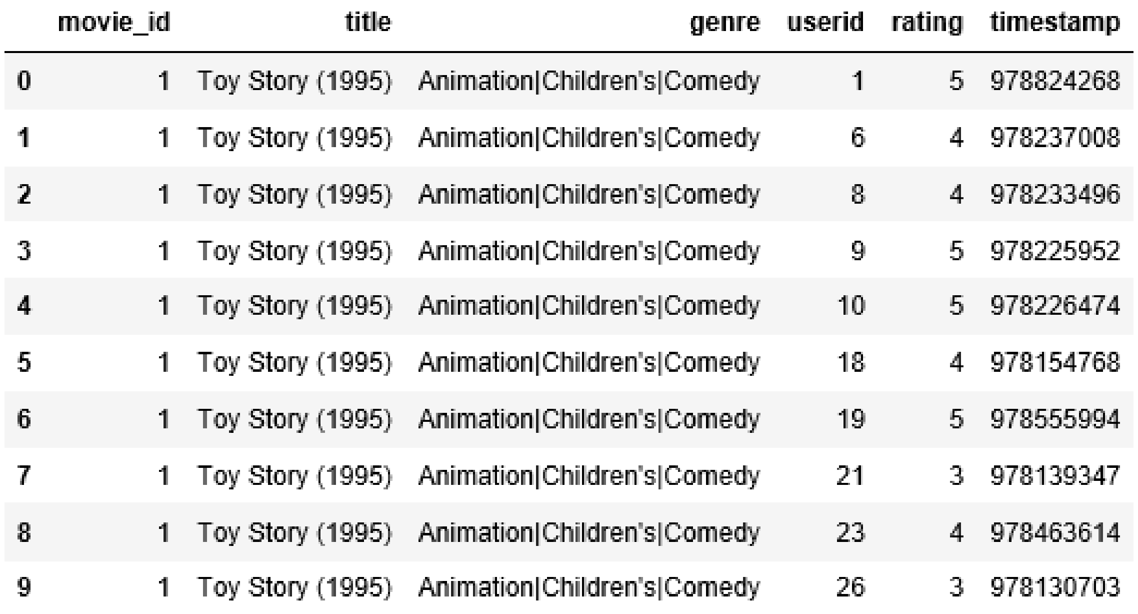
These findings are not directly used in building the recommender systems by the implementations followed in the thesis. The question that arises next is "Why look at all these properties when we are not using it in the models?". The answer to this question could be put forward as follows. In collaborative filtering using model based approach, we are trying to estimate the hidden or latent features. We do not use any of the features explicitly provided for the users (like age or occupation) and for items/movies (like genre or the year the movies were released in), it is possible that the latent features that we are trying to estimate could be very close to these properties in the data, even if not the exact features.

4.2 Implementation details and results

This section describes implementation details and the results observed for Popularity based recommender systems, content based recommender systems, collaborative filtering based recommender systems using matrix factorization and autoencoders. The implementation was carried out using Python 3.5.2. Pandas library was used for most of the data handling and manipulation. Numpy library was used for the mathematical requirements. Keras, on Tensorflow backend was used for autoencoder implementation.

4.2.1 Popularity based recommender system

The user ratings data was loaded into Pandas dataframe and merged with movies data to get the input data as follows.

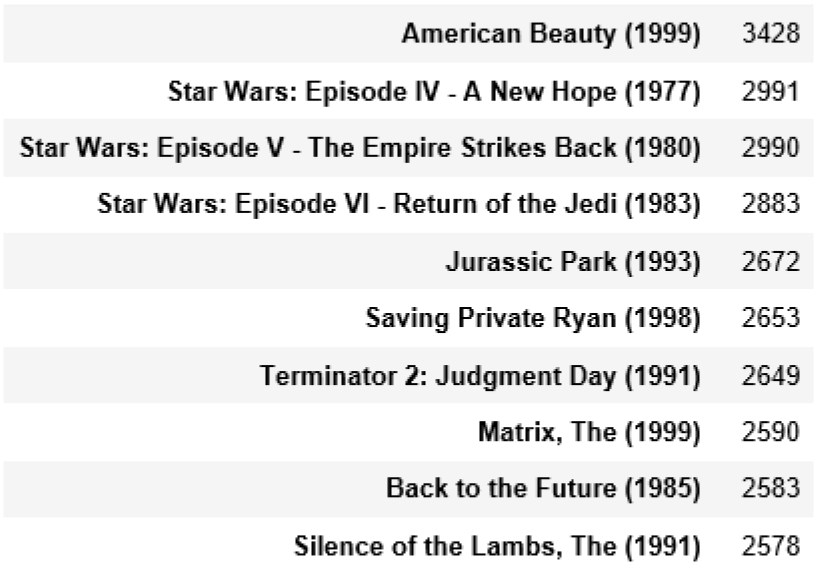


	movie_id	title	genre	userid	rating	timestamp
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268
1	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008
2	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496
3	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952
4	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474
5	1	Toy Story (1995)	Animation|Children's|Comedy	18	4	978154768
6	1	Toy Story (1995)	Animation|Children's|Comedy	19	5	978555994
7	1	Toy Story (1995)	Animation|Children's|Comedy	21	3	978139347
8	1	Toy Story (1995)	Animation|Children's|Comedy	23	4	978463614
9	1	Toy Story (1995)	Animation|Children's|Comedy	26	3	978130703

Figure 27. Input data - Popularity based method

Then, the number of ratings for each movie was calculated by grouping the data on the movie title. With the data now transformed to movies with respective number of ratings, the next task was to arrange them in the descending order of the number of ratings and recommend top 10 movies as recommendations.

The resulting recommendations are as listed below,



American Beauty (1999)	3428
Star Wars: Episode IV - A New Hope (1977)	2991
Star Wars: Episode V - The Empire Strikes Back (1980)	2990
Star Wars: Episode VI - Return of the Jedi (1983)	2883
Jurassic Park (1993)	2672
Saving Private Ryan (1998)	2653
Terminator 2: Judgment Day (1991)	2649
Matrix, The (1999)	2590
Back to the Future (1985)	2583
Silence of the Lambs, The (1991)	2578

Figure 28. Recommendations generated by Popularity based method

It can be seen that, the list consists of some of the most popular movies of all time. This method recommends the same list of the movies for every single user. It does make sense to recommend these movies if there is no requirement for personalization.

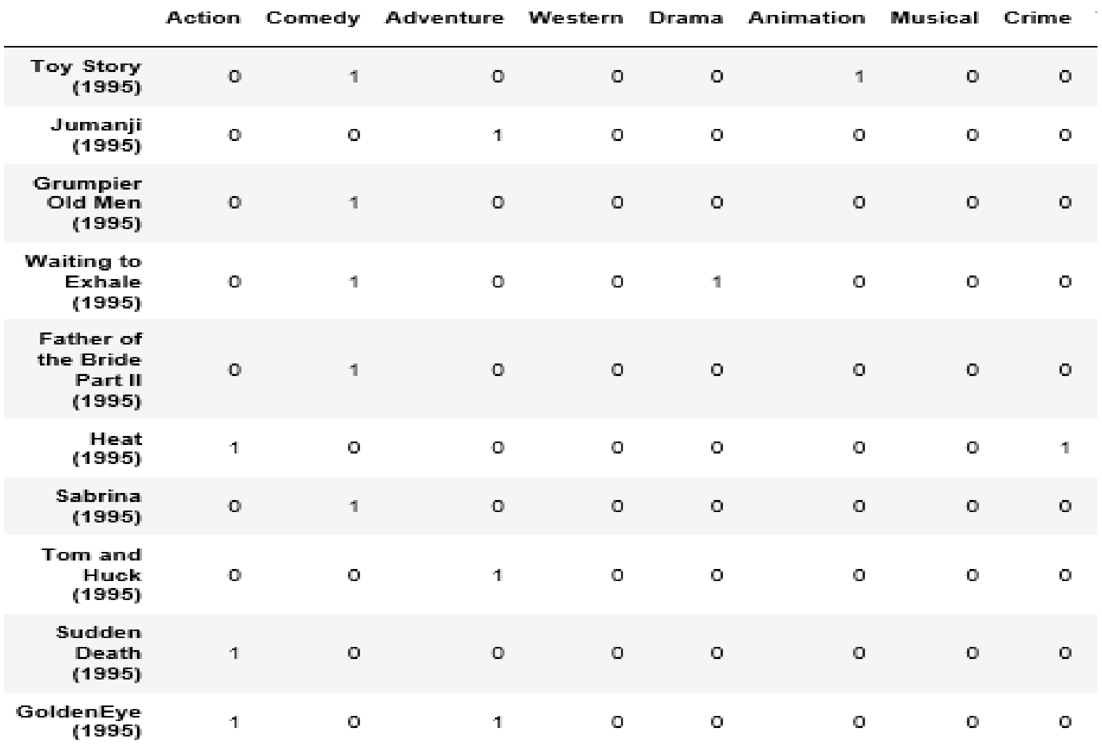
There is no evaluation metric that could be used for this method.

4.2.2 Content Based Method

The movie data was transformed to a matrix representing (movie X genre), with cells set to 1 if it belonged to that genre, 0 otherwise. Again, Pandas was used for data manipulation. For training and testing purpose, the matrix was split into train and test sets. The model was built using NearestNeighbor implementation of sci-kit learn. The similarity measure used was Minkowski distance implementation in python. Once the model was built, it was evaluated using Minkowski distance between the returned 10 nearest neighbors for the test set. However, since the data is available and no new data has to be fitted to predict the recommendations, it is safe to build the model using all the data. Since it is a neighborhood based approach and given the requirement of recommendation system, it does not make much of difference even when new movies are added.

In this implementation, recommending movies based only on a test movie genre is carried out. No specific user profile is provided or identified as the scope of the implementation was to understand to come up with movies similar to a given movie.

The input data is as below,



	Action	Comedy	Adventure	Western	Drama	Animation	Musical	Crime
Toy Story (1995)	0	1	0	0	0	1	0	0
Jumanji (1995)	0	0	1	0	0	0	0	0
Grumpier Old Men (1995)	0	1	0	0	0	0	0	0
Waiting to Exhale (1995)	0	1	0	0	1	0	0	0
Father of the Bride Part II (1995)	0	1	0	0	0	0	0	0
Heat (1995)	1	0	0	0	0	0	0	1
Sabrina (1995)	0	1	0	0	0	0	0	0
Tom and Huck (1995)	0	0	1	0	0	0	0	0
Sudden Death (1995)	1	0	0	0	0	0	0	0
GoldenEye (1995)	1	0	1	0	0	0	0	0

Figure 29. Input data for content based recommendation system

The average Minkowski distance for the test set was found to be 0, which means that all the recommended movies belonged to the same genres as that of the respective test points. This can be justified by the fact that, there is a huge number of movies and hence getting results with same combination of genres is not that difficult.

A simple test for identifying top 10 movies similar to “Toy Story (1995)” that belongs to the genres “Animated”, “Comedy” and “Children’s” resulted in the following list of movies.

MOVIES SIMILAR TO TOY STORY (1995)

TOY STORY (1995)
ADVENTURES OF ROCKY AND BULLWINKLE, THE (2000)
RUGRATS MOVIE, THE (1998)
CHICKEN RUN (2000)
AMERICAN TAIL: FIEVEL GOES WEST, AN (1991)
TOY STORY 2 (1999)
BUG'S LIFE, A (1998)
ALADDIN AND THE KING OF THIEVES (1996)
AMERICAN TAIL, AN (1986)
SALUDOS AMIGOS (1943)

It can be seen that, the first returned result is the movie itself. This could be handled before recommending the movies to users by dropping the movie. It can also be noted that all the returned movies belong to the genres “Animated”, “Comedy” and “Children’s”.

4.2.3 Collaborative Filtering – Matrix Factorization

The algorithm proposed by Gábor Takács et al. [6] was implemented in python. The algorithm was not a successful implementation given the data sparsity. It was observed that one update of P and Q low rank matrices took almost around 20 minutes, with very less change in the error with the subsequent steps. However, when the same was used for a very small size matrix, the convergence happened in a very short time. The reasons why this algorithm did not perform well will be discussed in the following chapter.

This algorithm was used on a very small matrix to verify the working. The input data was a matrix of order (5 x 6). The feature count to be extracted was set to 2 (i.e. the resulting low rank matrices P and Q would be of the order (5 x 2) and (2 x 6) respectively).

The actual rating matrix is,

$$\text{Sample Rating Matrix} = \begin{pmatrix} 5 & 4 & 4 & 1 & 1 & 3 \\ 1 & 2 & 2 & 4 & 4 & 5 \\ 5 & 4 & 4 & 1 & 1 & 3 \\ 1 & 2 & 2 & 4 & 4 & 5 \\ 5 & 4 & 4 & 1 & 1 & 3 \end{pmatrix}$$

This matrix was then split to train and test set as follows,

$$\text{Train} = \begin{pmatrix} 5 & 4 & 4 & 1 & 1 & 3 \\ 0 & 2 & 0 & 4 & 4 & 5 \\ 5 & 0 & 4 & 1 & 0 & 3 \\ 1 & 2 & 2 & 4 & 4 & 5 \\ 5 & 4 & 4 & 1 & 1 & 3 \end{pmatrix}$$

$$\text{Test} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

It can be seen that user 1,3 and 5 are similar & 2 and 4 are similar. Error only on the known ratings was optimized as discussed before. The resulting matrix estimate obtained by multiplying P and Q after 20 iterations (epochs), with learning rate "alpha" set to 0.05 and regularization term "beta" set to 0.05, was found to be,

$$\begin{pmatrix} 4.95198685 & 3.93906257 & 3.94280393 & 0.99544217 & 1.00066046 & 2.97882023 \\ 1.0262859 & 2.01434011 & 2.01625225 & 3.98094941 & 3.9797994 & 4.97177633 \\ 4.95576499 & 3.94284328 & 3.94658823 & 0.99864476 & 1.0038656 & 2.98391131 \\ 1.02475992 & 2.0129613 & 2.01487214 & 3.98012287 & 3.97897156 & 4.97025877 \\ 4.96228798 & 3.94731315 & 3.95106234 & 0.99769101 & 1.00292006 & 2.98522226 \end{pmatrix}$$

It can be observed that, the ratings predicted and the actual ratings are very close. The RMSE was found to be 0.03 on an average.

A graph of training RMSE vs Epochs and the test set RMSE is as shown below.

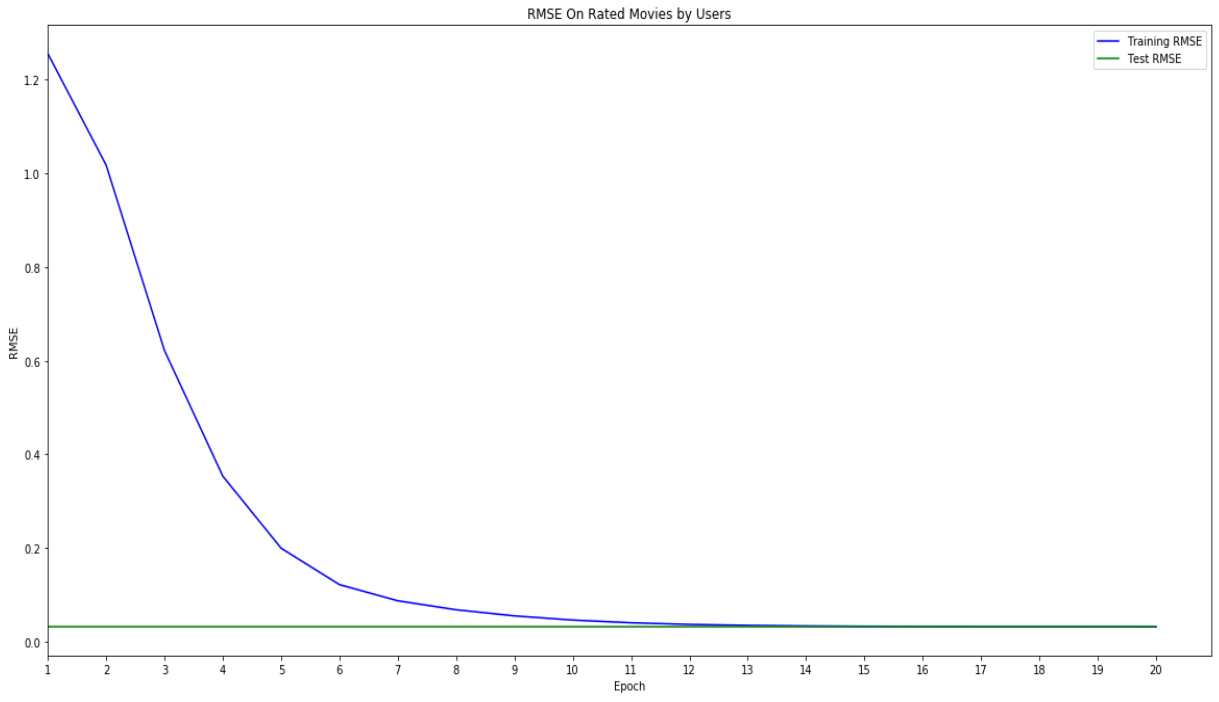
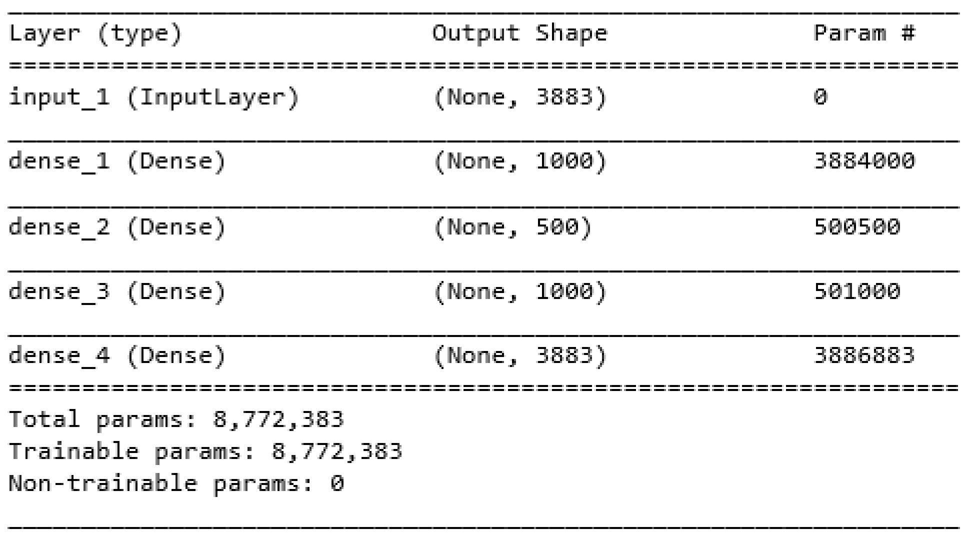


Figure 30. Training RMSE for Matrix Factorization

4.2.4 Collaborative Filtering – Autoencoder

Autoencoder with 3883 nodes in input layer and output layer, 1000 nodes in hidden encoder layer and 1000 nodes in the hidden decoder layer, with 500 nodes in the bottleneck layer were built using Keras neural network library in python, using Tensorflow backend. The activation function used in each layer was “rectified linear unit”. The training algorithm used for optimizing the error was “RMSProp”, a type of backpropagation algorithm with learning rate set to 0.001. Different optimization algorithms were tested and this one was chosen on the basis of performance and convergence time. The number of nodes in the different layers were chosen by trial and error method. A range of combinations were tested before deciding the number of nodes and layers. The parameters considered to decide the layers and nodes were the RMSE of the final model built, the time taken for computation.

The model summary is as shown below.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 3883)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              3884000
_________________________________________________________________
dense_2 (Dense)              (None, 500)               500500
_________________________________________________________________
dense_3 (Dense)              (None, 1000)              501000
_________________________________________________________________
dense_4 (Dense)              (None, 3883)              3886883
=================================================================
Total params: 8,772,383
Trainable params: 8,772,383
Non-trainable params: 0

_________________________________________________________________
```

Figure 31. Autoencoder model summary

The parameters in the model summary could be explained as follows.

Input layer has 3883 nodes (for the number of movies) and a bias node (not shown in the summary). These 3884 nodes are connected to all the 1000 nodes in layer 2 (dense_1). Which means that, the number of parameter for tuning would be (3883 movie nodes + 1 bias node * 1000) = 3884000. Then we have a bias node in this layer. These 1001 nodes are connected to 500 nodes in layer 2 (dense_2), the bottleneck layer. The number parameters here would be (1001 * 500) = 500500. Again, adding a bias to this layer and then connecting to layer 3 (dense_3) we have (501 * 100) = 501000 parameters and finally for the output layer, we have (1001 * 3883) = 3886883 parameters.

The algorithm converged to a stable state in less than 100 iterations with early stopping based on validation error. The RMSE of train and validation set is as shown below. The test RMSE was found to be 1.14. This is close to some of the presented works in the extended methods of autoencoders like denoising autoencoders, stacked denoising autoencoders etc.

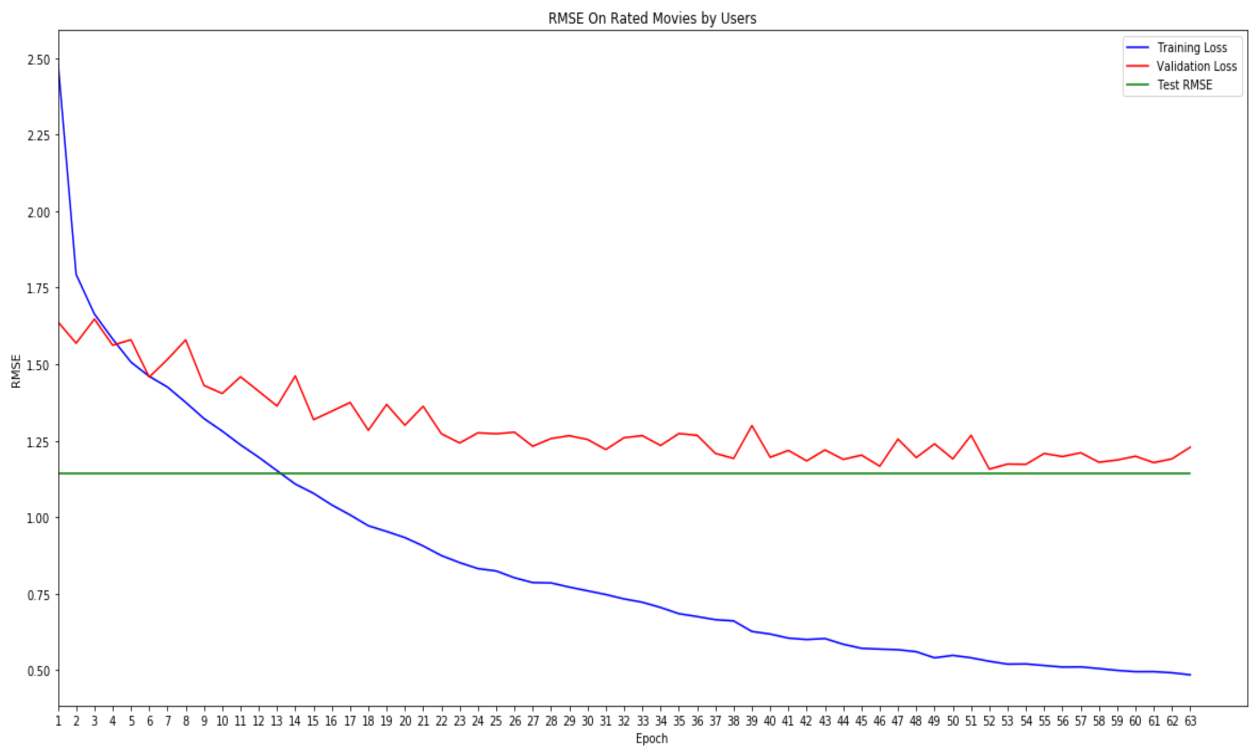
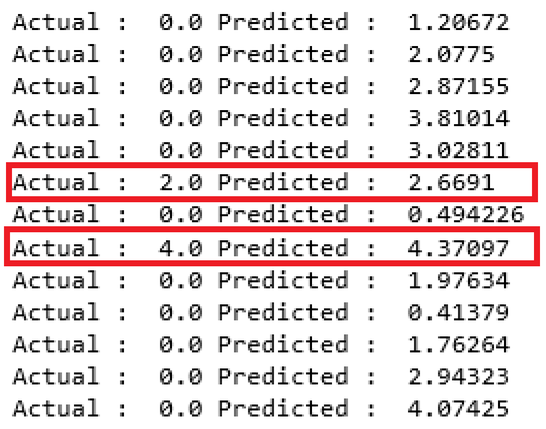


Figure 32. AutoEncoder RMSE

A screenshot of the predicted ratings on a subset of movies is as follows.



```
Actual :  0.0 Predicted :  1.20672
Actual :  0.0 Predicted :  2.0775
Actual :  0.0 Predicted :  2.87155
Actual :  0.0 Predicted :  3.81014
Actual :  0.0 Predicted :  3.02811
Actual :  2.0 Predicted :  2.6691
Actual :  0.0 Predicted :  0.494226
Actual :  4.0 Predicted :  4.37097
Actual :  0.0 Predicted :  1.97634
Actual :  0.0 Predicted :  0.41379
Actual :  0.0 Predicted :  1.76264
Actual :  0.0 Predicted :  2.94323
Actual :  0.0 Predicted :  4.07425
```

Figure 33. Autoencoder Rating prediction for a subset of movies for a user

It can be seen that, for the actual rating of 2, the model predicted 2.66 and for a rating of 4, the model predicted 4.37. Also, ratings for the movies that were not rated by the user can be seen in the figure.

Chapter 5

Discussion

Answering the questions from the initial stage of the thesis,

- What are the different ways of Building Recommendation systems?
  As seen from the discussions and implementations, on the basis of requirement, these systems could be implemented in many ways. They are popularity based method, content based filtering method, collaborative filtering method and hybrid methods.
- What type of system suits best for what kind of requirement?
  We could just recommend the most popular items using popularity based method. If we have the features for all the items and users, it is better to use content based filtering method. If we do not have explicit feature description and personalization is a key factor, it is better to use collaborative filtering method. A hybrid of these methods could be used to further improve the performance.
- How expensive are these approaches?
  Popularity based method is the least expensive method. Content based method is expensive in terms of storage of the model given that it is memory based, lazy learning method. Collaborative filtering – memory based methods are not very expensive, similar to content based filtering method. Collaborative filtering – matrix factorization is the most expensive method from the implementations carried out given the data density and sparsity. Collaborative filtering – autoencoders are less expensive when compared to matrix factorization and provides better results.
- How do we evaluate the effectiveness of the system?
  The offline methods are usually evaluated on the RMSE of the known ratings. Mean squared error or sum of squared error could also be used. Precision and recall are other metrics to measure the effectiveness, however, the results are not very accurate given the fact that “N” in top N recommendations may not actually appear in the top N list given the fact that if his liking matches to someone else’s exactly, then there is a very high possibility

of the movies liked by them to appear in the recommendations if the user has not rated many movies. Also, rating is relative. Some user may rate 5 on 5 for an item he likes, whereas someone else may rate it as 4.

- Is the evaluation of the systems really meaningful?

  The evaluation of the system for offline data does not really give a clear picture as to how good the recommender system is in real time.

Let us now decode some of the observations from the implementations carried out as part of this thesis.

5.1 Popularity Based Recommendation systems

This is the simplest method that any web site could use to recommend items to the users, completely based on popularity. There is not much to decode from the implementation observation for this method. We will have a look at some of the advantages and limitations of this systems.

Advantages

- Simplest possible implementation
- Requires very less computation power
- Time required to come up with these recommendations is very less
- Given the above two advantages, it is cheaper to implement

Limitations

- No personalization
- Same recommendations to every user

5.2 Content based filtering method

As seen from the results of implementing this system, it is clear that if we have the feature description of all the items and the user preferences or their profile, it is very easy to come up with recommendations. Also, it was seen from the results that the distance of the nearest n items was very close to the given item. Let us have a look at the advantages and limitations of this method.

Advantages

- Simple neighbourhood based approach.
- Recommendations are very close to the preferred items.
- Computational and time requirements are very less.

Limitations

- This system is completely dependent on explicit feature description of items and users
- The recommendations could be something very similar to the preferred item, but something that no user has every bought or something that isn't really good or liked by other users
- Recommendations are reliant features only

5.3 Collaborative filtering – Matrix Factorization

From the results it was seen that this algorithm was able to be implemented successfully only on a very small dataset. Just to give some basic overview, the system used was an Intel 6th gen I5 CPU with 8 GB of RAM, with dedicated memory for running the algorithm. It was observed that, for the user-rating matrix from movielens with the dimension (6040 X 3883), and the number of ratings in total being 1000209, i.e. with data sparsity of 95.76%, it was computationally expensive. The complexity could be given by O(n*m*k), where n is the number of users, m is the number of movies and k is the number of latent features that we are trying to extract. One iteration of low rank matrix update took an average of 25 minutes, with very less reductions in the subsequent iterations. Hence it was not able to estimate the low rank matrices effectively. This could be improved by using parallel distributed computation (High performance computing methodologies). However, since the goal was to understand the idea behind the algorithm, this algorithm was run with a small matrix as discussed in the implementation and results section. It was seen that the algorithm was able to converge in very few steps.

We will have a look at the advantages and limitations of collaborative filtering methods in general after a small discussion on the observations of the autoencoder method.

5.4 Collaborative filtering – AutoEncoders

This method is yet another simple algorithm. As seen from the results, it is evident that, the algorithm reaches a stable convergence point. We could just use the predicted ratings to generate the recommendations for the user. Since the goal of this method is to reduce the RMSE of rated movies by the users, it performs really well to meet that requirement. However, the results seem to be misleading for some users as the algorithm tries to predict the ratings as something very close to the average of most of the ratings by the user. It is not the most effective method either to implement with no dependence on using another method with it to improve the recommendations. It is seen from most of the researches presented that the RMSE obtained from this method and extended methods on this idea are close to 0.9 to 1 for movielens dataset, whereas the RMSE from the implementation as part of this thesis is found to be 1.18. It is notable that the time taken is hardly a few minutes for the algorithm to converge to a stable point.

Now, let us look at the advantages and limitations of collaborative filtering method.

Advantages

- Does not rely on explicit feature description of users or the items
- Personalization is the key difference when compared to every other implementation
- All recommendations are based on collaborating multiple user-item relationships
- Recommendation not restricted to just the user profile

Limitations

- Suffers from the following challenges
- Cold start problem – unable to recommend items to new users and new items won't make it to the recommendations.
- Scalability – unable to scale well with huge number of users and items.
- Data sparsity – Movielens dataset being used has only around 4% rating matrix filled by actual ratings. This makes it very hard for the system to estimate the ratings.

Collaborative filtering suffers from one major issue. This method can be used to modify the recommendations by simple user manipulation of data. Let us have a look at a very simple example.

Let’s assume a fraudulent marketing team of a movie M wants to make sure all users are recommended their movie. All the team has to do is, create fake users and rate their movie the highest and rate all other movies lesser. This does affect the recommendations based on collaborative filtering method. Even though we can argue that it establishes user-user and item relationships, it is possible to launch this type of attack on collaborative filtering methods. It is not easy to identify suck attacks and measures have to be taken to control the fake user creation by using verification methods. There are other possible methods to make the system to behave the way we want it to. Hence it is necessary to look at this aspect of the recommendation systems than just trying to improve the rating predictions or top n item ranking.

Chapter 6

Conclusions

This chapter summarizes the thesis by talking about the extent to which the implementation met the set goal for thesis. Then, possible future work is presented.

6.1 Conclusion

The aim of the thesis was to understand and implement some of the approaches of building recommendation systems. Tapestry and Grouplens' Usenet projects provided a better insight on the recommendation systems and their evolution. Netflix's system architecture provided an overview of where some of the operations for recommendation systems are carried out, giving a clear outline as to where the project's main focus area has to be given the dataset. As seen from the implementation and results, four different approaches to build these systems were carried out. Popularity based recommendation systems, Content based recommendation system based only on the movie genre, collaborative filtering using matrix factorization and autoencoders were implemented.

Popularity based method, Content Based method and Autoencoder method were a success with respect to the selected movielens dataset. However, the matrix factorization method failed to reach a stable enough state to predict accurate user ratings for the dataset. Fortunately, it did provide an insight on how to implement the algorithm tweaked to meet the requirements of Recommendation systems, which worked well with a small matrix.

Popularity based method gave reasonably good results (it suggested some of the best movies of all time). Content based method too could come up with very good recommendations (it could suggest the most relevant movies given a movie based on their genre). Matrix factorization method, which is considered one of the best methods and awarded the Netflix Prize challenge's first place, struggled due to scalability issues. Autoencoders performed well, reaching an RMSE of 1.18 which is reasonably good compared to other implementations trying to solve the same problem. It is safe to say that the intended goals of the thesis were met successfully.

Some of the setbacks of these methods were seen in the discussion section. It is evident that none of these methods are capable of effectively meeting the business requirements alone (unless it is a popularity based method that the company wants). It is always better to use an ensemble of these methods to attain better performance.

Given the importance of Recommendation systems in today's internet, it is very important to learn about how they work to improvise their performance. This thesis could be used as a starting point for a new-comer in this field, as the thesis was started with no experience in the particular systems and some of the basic aspects of recommendation systems like categorization of recommendation systems, evaluation metrics (and why none of them could be considered as the best) and basics of implementing state-of-the-art techniques are talked about. Overall, it was a very good learning experience, successfully implementing the systems, understanding the challenges faced by the systems and the challenges faced when implementing the systems.

6.2 Future Work

As any other system, there is always scope for betterment of the implementation. The built systems could be used with each other to improve the recommendations. Hybrid systems are the way to go and commercially used approaches to solve this problem. A combination of popularity based system and collaborative filtering could help in solving the cold start problem. A combination of popularity based system and content based filtering could be used to recommend items of not just the same type but the items of same type that are currently popular. Matrix factorization method could be made scalable by using High performance computing. Autoencoders could be extended to build recommendation systems using stacked autoencoders, denoising autoencoders, stacked denoising autoencoders etc. A hybrid of content based filtering and collaborative filtering is found to improve the recommendation system. These implementations are currently in practice commercially. Even though the possibility of reducing the error even by 0.1 when predicting ratings as in the case of autoencoder method is very small, it still makes sense to work on this given the importance of recommendation systems. Combining features like social circle (facebook friends), location details etc. could help in improving these systems in providing better recommendations. The key thing to remember is that it is not just about improving accuracy or meeting some preset evaluation metrics but it is to build recommendations that serve the intended purpose and with less computation power and time consumption.

References

[1] Ò. Celma and P. Lamere, "If you like the beatles you might like: a tutorial on music recommendation," ed, 2008, pp. 1157-1158.

[2] D. Goldberg, D. Nichols, B. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61-70, 1992.

[3] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl, "GroupLens: applying collaborative filtering to Usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77-87, 1997.

[4] J. Schafer, J. Konstan, and J. Riedl, "Recommender systems in e-commerce," ed, 1999, pp. 158-166.

[5] J. Schafer, J. Konstan, and J. Riedl, "E-Commerce Recommendation Applications," *Data Mining and Knowledge Discovery,* vol. 5, no. 1, pp. 115-153, 2001.

[6] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Matrix factorization and neighbor based algorithms for the netflix prize problem," ed, 2008, pp. 267-274.

[7] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews," ed, 1994, pp. 175-186.

[8] X. Amatriain and J. Basilico. (2013). *System Architectures for Personalization and Recommendation*. Available: https://medium.com/netflix-techblog/system-architectures-for-personalization-and-recommendation-e081aa94b5d8

[9] C. C. Aggarwal, *Recommender Systems The Textbook*. Cham: Cham : Springer International Publishing : Imprint: Springer, 2016.

[10] P. Pirasteh, J. J. Jung, and D. Hwang, "Item-based collaborative filtering with attribute correlation: A case study on movie recommendation," vol. 8398, ed, 2014, pp. 245-252.

[11] E. Kharitonov, "Empirical Study of Matrix Factorization Methods for Collaborative Filtering," in *Pattern Recognition and Machine Intelligence: 4th International Conference, PReMI 2011, Moscow, Russia, June 27 - July 1, 2011. Proceedings*, S. O. Kuznetsov, D. P. Mandal, M. K. Kundu, and S. K. Pal, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 358-363.

[12] Y. Ouyang, W. Liu, W. Rong, and Z. Xiong, "Autoencoder-Based Collaborative Filtering," in *Neural Information Processing: 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part III*, C. K. Loo, K. S. Yap, K. W. Wong, A. T. Beng Jin, and K. Huang, Eds. Cham: Springer International Publishing, 2014, pp. 284-291.

[13] S. Sedhain, A. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders Meet Collaborative Filtering," ed, 2015, pp. 111-112.

[14] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Systems With Applications*, vol. 69, pp. 29-39, 2017.

[15] J. Beel, S. Langer, M. Genzmehr, B. Gipp, C. Breitinger, and A. Nürnberger, "Research paper recommender system evaluation: a quantitative literature survey," ed, 2013, pp. 15-22.

[16] F. Harper and J. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, pp. 1-19, 2016.

[17] A. Hübner. (2015). *The Movielens 1M Dataset*. Available: https://public.tableau.com/profile/andreas.h.bner#!/vizhome/ml-1m/Movielens1MStory