2012

Recommendations for "KIWI" Exploring Different techniques for Recommendations in a Kazakh Online Video Website

Tair Kuanyshev
Dublin Institute of Technology, tair.kuanyshev@dit.ie

Recommendations for “KIWI”

Exploring different techniques for recommendations in a Kazakh online video website

Tair Kuanyshev

A dissertation submitted in partial fulfilment of the requirements of

Dublin Institute of Technology for the degree of

M. Sc. in Computing (Data Analytics)

September 2012

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: _____

Date: 03 September 2012

ABSTRACT

The rapid popularisation of the Internet has increased the volume of data collected by business, industry and society. A need has arisen in new ways of marketing and user targeting for the e-commerce websites in development and revenue increasing tasks. The proper usage of customer's personal information has had a significant impact on business activity and sales, this personal information relates mainly to customer purchases and viewing history. Personal recommendations from retail sale sites such as Amazon and eBay are suggesting products for online shopping, reddit.com suggesting interesting websites, and last.fm, Netflix helping people to find movies and music have shown an increase in company profits

The aim of this project is to build a recommender system for an online website based on an empirical study using real-world data. The real world data was taken from the website Www.kiwi.kz which is an online video sharing website, providing online video, broadcast and radio to users since 2009.

The research problem sought to develop the most suitable recommender system for an online video sharing website and to evaluate the effectiveness of this model on the real world data. This research attempts to develop the recommender system which will suggest possibly interesting video clips for users on an online video sharing website. It also proposes methods and techniques that can be used to implement recommender systems and evaluate its effectiveness on an offline experiment. This project is based on performing an empirical evaluation to establish whether a recommender system can be developed that will predict whether people will like a particular video.

Key words: *Recommender systems, collaborative filtering, content-based filtering, text analysis, similarity metrics, prediction, Python, video, user-item matrix, ratings*

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my supervisor SarahJane Delany for her advice, comments and support during completing my dissertation.

I would like to thank to my wife Kamshat for her support, patience and understanding.

I also would like to express my gratitude to my Study Group colleagues Vincent and Raj for their co-operation and encouragement.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

1.1 Introduction

The rapid popularisation of the Internet has increased the volume of data collected by business, industry and society. Companies are drowning in tonnes of data, and identifying valuable and explicit knowledge from that data remains a key challenge.

A need has arisen in new ways of marketing and user targeting for the e-commerce websites in development and revenue increasing tasks. The proper usage of customer's personal information has had a significant impact on business activity and sales, this personal information relates mainly to customer purchases and viewing history. If this information is used correctly it can be used to increase revenue and customer satisfaction.

According to Akioka S. *et al* (2008), "The quick spread of web services has triggered the flood of information, and requests people to choose a valid set of queries for useful information retrieval in order to extract what they really need". Recommender systems aim to provide a service, which suggests to customers the "items" which they could find interesting or useful for themselves. A proper implemented recommender system could increase both companies business and customer's satisfaction, whereas the bad recommendation system can push away customers. Youtube, is available on the internet, it allows people all over the world to share their personal videos, it is estimated that Youtube is uploading more than 60 hours of video every minute (January, 2012). The success of Youtube has triggered competition among other websites providing similar services.

There are thousands of e-commerce websites and these sites make their money from customers, it is important for e-commerce sites to provide a good quality service and maintain customer satisfaction so that those customers will continue to use the site. Users have also become more

demanding, by expecting fast connection, a wide range of goods and services that are high quality. Good personal recommendations which suggest personal choices have become a means of increasing revenue for e-commerce sites. Personal recommendations from retail sale sites such as Amazon and eBay are suggesting products for online shopping, reddit.com suggesting interesting websites, and last.fm, Netflix helping people to find movies and music have shown an increase in company profits. Recommender systems as part of predictive analysis can be represented as a combination of different fields of collective intelligence and computer science such as machine learning, data mining, statistics, information retrieval and collaborative filtering.

The aim of this project is to build a recommender system for an online website based on an empirical study using real-world data. The real world data was taken from the website Www.kiwi.kz which is an online video sharing website, providing online video, broadcast and radio to users since 2009. Kiwi.kz is one of the most popular websites in Kazakhstan with around 450 000 registered users who can upload and share their videos, Kiwi.kz claims to have 200 000 unique users each day. Kiwi.kz provides opportunity to users to express their attitude to particular videos by "liking" them, add as their favourites and leaving comments. A recommender system suggesting personal video recommendations for users is an up-to-date function for helping to increase user choice which improves user satisfaction and more importantly company revenue.

The research problem sought to develop the most suitable recommender system for an online video sharing website and to evaluate the effectiveness of this model on the real world data provided by the kiwi.kz. This research attempts to develop the recommender system which will suggest possibly interesting video clips for users on an online video sharing website. It also proposes methods and techniques that can be used to implement recommender systems and evaluate its effectiveness on an offline experiment. The technical implementation of the system consists of a combination of machine learning models having for inputs data of user's video watching history. The impact of particular attributes of user's behaviour such as "liking" the

video, and subscribing to the video have been analysed. The desired output is a recommender system, consisting of the most valuable attributes combination, which has shown the highest performance on the real world data available from the website.

This project is based on performing an empirical evaluation to establish whether a recommender system can be developed that will predict whether people will like a particular video. The tasks of the project include:

1. Review and analysis of the existing recommender system algorithms and methods
2. Knowledge representation: identifying the most valuable attributes and constructing the data set from available data for the recommender system
3. Recommender system's design: selecting the most appropriate similarity metrics in collaborative filtering.
4. Evaluating the performance of the different techniques and analysing the trade-off between the computation time and prediction score.

1.2 Research design

In order to arrive at the results and conclusions the research is based on empirical study. The experiment involved the analysis of a given data set and the evaluation of the performance of the data using programming tools such as Python and R. For the project's experiment the data mining methodologies in data preparation, data transformation and data partitioning have been applied. Collaborative filtering techniques and evaluation metrics have been used to identify the most suitable model throughout the experiment.

The project included:

- Data preparation and data transformation tasks such as data analysis, merging data as a data matrix and attribute selection

- System design tasks including data base and programming tools selection, machine learning and collaborative filtering techniques selection, and data partition selection for data training and testing.
- Evaluation metrics selection for results analysis and contribution

1.3 Challenges

There are a number of challenges facing this project:

- The project requires a significant level of programming skills.
- The project requires a high level of computation power in terms of the volume of data to be analysed and the time consumption involved.
- The data was from a website based in Kazakhstan and this meant that the textual data contained different encodings which caused some problems.

1.4 Organisation of the dissertation

The dissertation aims to analyse the existing research in the area and to design and implement an experiment which applied the gained knowledge to the real world data. The dissertation is organised in the following way:

Chapter 2 – Literature review and Background research

This chapter presents an overview of the domain of recommender systems from the technological and business sides. The definitions of information and knowledge are presented. Recommender system functions, used data and knowledge sources, different approaches of designing recommender systems such as collaborative filtering, content-based and hybrid models are analysed. An overview of machine learning techniques and collaborative filtering techniques, the use of these techniques and examples of usage of these techniques in video recommender

systems are given. Collaborative filtering techniques are covered in more detail with an analysis of approaches and similarity metrics used in existing projects. Collaborative techniques are the most popular techniques in video recommender systems.

This chapter also examines text analysis and includes a brief explanation of the domain of text mining, and how textual analysis could decrease the problem of data sparsity and increase accuracy in recommender system being implemented on this project.

Chapter 3 –Experiment design and methodology

This chapter describes the design of the recommender system, the features and organisation of the data from the Kiwi.kz website and explains the experimental methodology. The structure of the used data set which has been created from the analysis of the original data and how the attributes of this data set impacted the performance of recommender system are discussed in this chapter.

The identification of proper similarity metrics and identification of their performances on the original and constructed data set are covered in this chapter. In summary, the main attributes used to create recommender system are identified and discussed.

Chapter 4 – Evaluation and Results

This chapter aims to identify the best approach for building a video recommender system based on real data by evaluating and comparing the performance of different techniques. The evaluation of the model is based on the accuracy of the results achieved. Results based on the experiment are described in this chapter. The best performed method and the structure of the data on which the model was based is discussed. The assumption and challenges during the evaluation processes briefly discussed in the summary.

Chapter 5 – Conclusion

This chapter includes the general overview of the results of the experiment and conclusions of the research. The recommendation for future research in the area and challenges throughout the research are given.

2. LITERATURE REVIEW AND BACKGROUND RESEARCH

2.1 Introduction

This chapter provides the introduction to the recommender system domain, their types and techniques used to implement them.

Everyday people express different opinions about different things on the internet by liking, disliking and so forth. People rely on recommendations from other people either by word of mouth, recommendations in newspapers, TV and so forth. Also people tend to like things that look familiar to other things they like before. If a person likes to watch "Godfather" and "Scarface", it is likely that a person would like to watch another movie that includes acting by Al Pacino. This kind of information can be used to make a prediction such as likes or dislikes. Recommendations are all about predicting patterns of taste, and using them to discover new and desirable things you did not already know about (Owen et al. 2012, p.14).

Recommender Systems (RSs) are software tools and techniques providing suggestions for items that may be used by website users. Miller *et al.* (2004) define recommender systems as "a popular technique for reducing information overload and finding products to purchase". According to Melville and Sindhwani (2010) the goal of recommender system is "to generate meaningful recommendations to a collection of users for items or products that might interest them". The suggestions are related to different decision-making processes, so users are directed toward those items that most met the criteria of user's interest (Burke 2007). Recommender systems have been used in various applications and provide recommendations as to what movies to watch (like Netflix, IMDB, MovieLens), what books to read (like Bookreads, Amazon), what web pages to look at (like Reddit, delic.io.us), or what videos to watch next (Youtube). The "thing" or "item" suggested by a recommender system is depended on area of use of RS.

The design of recommendation engines depends on the domain and the particular characteristics of the data available (Melville and Sindhwani 2010). According to the design, graphical user interface and specific types of "items" RSs use the various types of techniques to generate recommendations. The main aim of RS is to provide recommendations that possibly help users more effectively identify content of interest from a potentially overwhelming set of choices related to their interest.

Increasing amounts of available information on the internet and growing popularisation of e-commerce Websites, especially large online shopping companies such as Amazon and eBay overwhelm customers with tonnes of information. It is important to provide quick and accurate recommendations among the variety of choices to attract the interest of customers and bring benefits for companies.

2.2 Recommender System Function

The use of recommender systems varies according to area of business using it. To identify the aim of using recommender system there should be clear definitions and purposes of recommender system role in company business strategic objectives. For instance, a movie recommender system (such us Netflix, IMDB) suggests to customers which kind of movies to watch, which would increase the interest on the unwatched movies by users and therefore will increase the revenue to the companies. In general, there are different reasons for recommender system use in business.

The five main functions of recommender system engines includes (Ricci *et al.*, 2011, p.5-6):

Increase the number of items sold. One of the most important functions in recommender system is to be able to sell additional set of items compared to those usually sold without any kind of

recommendation. It is achieved by recommending items that a user will find interesting and suitable for them.

Not only commercial organisations can benefit from recommender systems, non-commercial organisations also have similar goals, even if they do not measure it by profit. It is close to this research project, which is for a recommender system for an online video sharing website. As an example there is the Youtube recommender system, which provides video recommendations to increase the time user spent on a website.

Sell more diverse items: Suggesting items for users that might be hard find without precise recommendations. Suggesting more suitable personal recommendation is of more benefit that suggesting the top interested or most popular items. Personal behaviour analysis helps recommender system to identify more interesting items for users among wide range of items.

Increase the user satisfaction: Well performed recommender system could not only satisfy user needs, it could be a edge point in user service satisfaction. More precise personal recommendations increase interest in the project, and user will find a service as interactive system that could find accurate and effectively suggest items that user needs or finds interesting.

Increase user fidelity: As a well-designed recommender system performs well and produces accurate personal recommendations, a user would find a service or website more reliable and be more loyal to a particular website. Recommender systems identify customers and treat them as a valuable visitor. As a result, the more time a user will spend on a website, the more accurate and effective recommendations website will produce to the users.

Better understanding of what the user wants: The importance of information collected and analysed by a recommender system can be used by a company in different other ways. The analysis of the feedback from the given recommendations to a user helps recommender system to more deeply understand the user needs.

Resnick and Varian (1997) classify recommender systems according to their technical characteristics, domain space characteristics and evaluation criteria. The analysis describes the impact of a complexity of recommender system structure influence to the cost of maintaining it.

The role of recommender system is to edge an advantage for businesses in achieving their goals. Different purposes of using recommender system have been mentioned above and different technological methods of implementing and designing them will be discussed in the next sections.

2.3 Data and Knowledge Sources

Data is the main source of knowledge for recommender systems. Recommender systems obtain various kinds of data in order to build recommendations. According to available data different techniques can be used to design RS. For example collaborative filtering techniques can be used with the rich historic data about users past behaviour, whereas content-based filtering techniques are more reliable on a data with good information about "items" and with less information about users. In general recommender system uses data which can be defined as three main objects, which are "items", "users" and transactions (relations between users and items) (Ricci *et al*., 2011, p.8-10). The objects recommended to the users are items. The items can have varying values (positive or negative) an item with a positive value is recommended to other users. The users themselves can vary according to the goals and functions of the recommender systems. Transactions are the recorded data of user interactions within the system, consisting of important information about users personal preferences and behaviour.

The most popular transactional data that recommender system collects are ratings. The collection of ratings is an explicit source of information for designing recommender system. The most popular recommender systems use collaborative filtering techniques, where recommendations are based on the rating.

The ratings are based on the opinions of users to particular items and can either be explicit in the form of scales, such as 1-5 ratings, and in implicit form such as purchases or click-through (Su and Khoshgoftaar 2009). Explicit forms of ratings such as 1 to 5 scale, express opinions of users to particular items in a numeric form, where 1 could be related to not interested/very bad and 5 could be very interesting/the best. The data about user's item purchases usually converts into a user-item matrix, where values represent user's ratings, and missing values are items not rated by users. In Table 2-1, an example of a user-item matrix, where 1 is a rating for videos liked by users, 0 is not strong enough to represent liked videos, and missing cells are unwatched videos.

	Video 1	**Video 2**	**Video 3**	**Video 4**	**Video 5**
User 1	**1**		**0**		**1**
User 2	**1**		**1**	**0**	
User3		**0**	**1**		**0**
User4	**0**	**0**		**1**	**1**

Table 2-1 Example of user-item matrix on binary ratings

2.4 Recommendation Techniques

The core function of recommender systems is to identify the useful items for the user. The system must predict items which a user may find interesting. Prediction may be based on the correlation between user's preferences to the same items, user's purchase history and so on. Recommendation techniques based on knowledge sources divided into four different classes (Burke 2007):

Collaborative: The main source of information in collaborative filtering techniques is user ratings. Recommendations are generated based on the similarity between users rating history. According to Resnick *et al*. (1994) "Collaborative filtering help people make choices based on the opinion of other people. Predicting is based on the heuristic that people who agreed in the

past will probably agree again". Melville and Sindhwani (2010) suggest that collaborative filtering can perform in domains where there is much content associated with items, or where the content is difficult for a computer to analyse, and that CF has the ability to provide serendipitous (lucky) recommendations, which can recommend items that are relevant to the user, but do not necessarily match with content from the user's profile.

Content-based: The system generates recommendations from the features associated with products/items the user has rated highly. The system tries to predict and suggest items that are similar to those which user has liked or purchased before. The main idea of content-based recommender system is to compare preferences and interests of a user profile with the attributes of a content item, to recommend to the user new interesting items. The content-based approach has its roots in the information retrieval (IR) community, and employs many of the same techniques (Balabanovic and Shoham 1997).

Demographic: A demographic recommender provides recommendations based on a demographic profile of the user. Recommendations produced by a recommender system based on combination of information about demographic area of a user and user's rating history, for example a recommendation for a user can be varied depending on the language, country or age of the user. Structured demographic information about users and the characteristics of web pages can be treated as a main source of information in demographic recommender systems.

Pazzani (1999) proposed an alternative approach for demographic recommender systems. Proposed approach minimising the effort of obtaining demographic information and leveraging work on text classification to classify users with Winnow algorithm to learn the characteristics of web pages associated with users.

Knowledge-based: Knowledge-based systems recommend items based on specific domain knowledge about how certain item features meet users' needs and preferences and, ultimately how the item is useful for the user. A system suggests products based on inferences about a

user's needs and preferences. Knowledge-based recommender systems are not dependent on large amount of statistical data about particular rated items or particular users, and systems need only enough knowledge to judge items as similar to each other (Burke 2000).

Additional classes of recommender systems have been proposed by (Ricci *et al.*, 2011, p.13-14):

Community based: This type of system recommends items based on the preferences of the users friends and it is related to the social recommender systems, this type of recommender system acquires information about user's friends' preferences and build models based on user's friend's ratings. Evidence suggests that people tend to rely more on recommendations from their friends than on recommendations from similar but anonymous individuals. The advantages of the community-based approach of dealing with a new user or a new item and improving recommendation accuracy has been analysed by Arazy *et al*., (2009).

Hybrid recommender systems:

This type of recommender system is based on a combination of two or more techniques mentioned above. The advantage of this system is that one type of system is used to fix the disadvantages of the other system, so they can outperform using only one of them.

Cotter and Smyth (2000) proposed an approach which allows collaborative filtering and content-based methods to produce separate ranked lists of recommendations, and then merge their results to produce a final list. Another approach proposed by Claypool *et al.*, (1999) was combination of the two predictions using an adaptive weighted average, where the weight of the collaborative component increases as the number of users accessing an item increases. Balabanovic and Shoham (1997) proposed the hybrid system called Fab using both collaborative and content-based approaches, which has benefits in 1) scaling (to an increasing number of users and an increasing number of items) and 2) automatically identifying emergent communities of interest

in the user population. The hybrid system of collaborative and content-based filtering (Debnath *et al*., 2008) has shown better results when compared to pure content-based method.

2.5 Collaborative Filtering

The term "collaborative filtering" was first introduced by Goldberg *et al.*, (1992). The introduced mail system called *Tapestry*, helped users to identify relevant and interesting mails in tonnes of mails. The main aim of collaborative filtering is to predict a list of items to a particular user (the active user) based on other users votes with similar taste in past (the user database). Resnick *et al*. (1994) defines collaborative filters as a tool which helps people make choices based on the opinion of other people and proposes a collaborative filtering based system called GroupLens, which helps people to find articles which they like in the huge stream of available articles. Sarwar *et al*., (1998) gives definition of collaborative filtering as "system helping address information overload by using the opinions of users in a community to make personal recommendations for documents to each user". Breese *et al*, (1998) use collaborative filtering term as equal to "recommender system", and define it as "use of database of user preferences to predict additional topics or products a new user might like".

Collaborative filtering collects user's feedback to exploit similarities in user expression behaviour between users. Miller *et al*. (2004) argue that the main assumption of collaborative filtering is that two users (user A and B) rate number of items (n items) in similar way, or have similar behaviours (like watching, listening, reading), therefore will rate or act on other items similarly. Therefore, the recommendation to a user is based on the similarities between other users with the high degree of similarity in the past. Recommendations based on collaborative filtering techniques use the data from a database of preferences for items by users. Usually preference database contains a list of users n $\{u_1, u_2, ..., u_n)$ and a list of items m $\{i_1, i_2, ..., i_m)$, and each user u_i has a list of items I_u, which user has rated, or expressed their attitude to

particular item. Collaborative filtering algorithm uses the preference data and produce recommendations based on the similarities between users (user based approach) or between items (item based approach).

Collaborative filtering is the most popular technique used in recommender system, and most popular services such as Amazon, Netflix, eBay and so forth use collaborative filtering techniques in their systems. Collaborative filtering provides three key advantages over content-based filtering: 1) support for filtering items whose content is not easily analysed by automated processes; 2) the ability to filter items based on quality and taste; 3) the ability to provide serendipitous recommendations (Herlocker *et al*, 1999). Rashid *et al*, (2005) argues that the advantage of collaborative filtering is in users opinion based decisions. Collaborative filtering interpret information that are formal, quantitative and observed and analyse it directly through data. However, there are remaining challenges in the collaborative filtering recommender systems. The two fundamental challenges remain in collaborative filtering (Sarwar *et al*, 2001). Firstly, the challenge is improving scalability of collaborative-filtering algorithms.

Existing algorithms work well with tens of thousands of users, but the reality demands fast algorithm for computation for millions of potential neighbours. The fast search requires reducing scalability in neighbour identification, which affects to the result of recommendations. The second challenge is to improve the quality of the recommendations for the users. Recommendations must be trustful and help users to find item which they might like. More additional challenges in collaborative filtering and their characteristics include (Su and Khoshgoftaar 2009):

Data sparsity challenge: Recommendations for collaborative filtering are based on user-item matrix, which is in reality very sparse. The difficulty of providing recommendations to a new user or item due to the lack of information about them is called cold start problem. New items

need to be rated before being recommended and new users need to purchase items to create personal history to allow providing recommendations to them.

Associations: The problem with the same items with different names or entries is refers to the problem of synonymy. Most recommender applications are unable to identify latent associations and treat them as different items.

The challenges of "gray sheep" and "black sheep" refers to the opinions of some group of people whose taste are almost opposite to the most people (black sheep) or do not consistently agree or disagree with any group of people (gray sheep).

The problem of shilling attacks refers to the problem, where user may express purposely their opinions by giving positive recommendations to the products they are related to, and negative recommendations to the products of competitors.

Collaborative filtering according to approaches uses in producing recommendations is divided into two main categories: Memory-Based and Model based algorithms (Melville and Sindhwani 2010; Sarwar *et al*, 2001; Segaran 2007).

2.5.1 Memory-based algorithm

Memory-based algorithms or neighbourhood based algorithm make predictions based on user-item database, where a prediction to the active user is based on a weighted combination of ratings from a group of users similar to the active user on user-item database. Recommender system use prediction techniques to find a set of users, known as neighbours, which have a similar profile history to the target user. The k-nearest neighbour algorithm is the most widely used approach in memory-based collaborative filtering.

The memory-based algorithm consists of 3 steps (Melville and Sindhwani 2010):

1. Assign a weight to all users with respect to similarity with the active user.
2. Select *k* users that have the highest similarity with the active user (also called as the neighbourhood)
3. Compute a prediction from weighted combination of the selected neighbours' ratings.

In step one, the measure for the weight *w*(*a,i)* between the active user *a* and user *i* can be any similarity or correlation measures such as Pearson correlation, Euclidean distance or Jaccard distance. Next, the number of users with the highest similarity is chosen. In the step three, prediction computed as the weighted average, and equation (1) can be used.

Breese *et al*., (1998) suggested that the predicted rating of the active user for item is a weighted sum of ratings of the k-nearest neighbours.

$$p_{uj} = \overline{R_a} + k\sum^{n}_{i=1} w(a,i)(\overline{R_{i,j}} - R_i) \qquad (1)$$

Where p_{aj} predicted rating for item *j*, *n* is the number of users with non-zero weights, $R_{i,j}$ a set of ratings corresponding to the rating for user *i* on item *j*. *w*(*a,i)* reflect similarity, correlation or distance between *k* neighbours for each user *i* and the active user *a*.

Recommendations and predictions for the target user are based on the preferences of neighbours by applying different algorithms on the neighbours profile data.

2.5.1.1 Similarity metrics

Similarity between two users or items can be measured by treating each document as a vector of ratings for items. Similarity can be measured by similarity metrics such as Pearson correlation, Euclidean distance or Jaccard index. Pearson correlation is widely used in most existing recommender systems (Linden *et al*., 2003; Resnick *et al*., 1994; (Sarwar *et al*, 2001)) and

according to Segaran (2007) Pearson correlation has better results over Euclidean distance in situations where the data is not well normalised.

Pearson Correlation

Pearson correlation coefficient was first used in collaborative filtering for GroupLens project (Resnick *et al*., 1994), where was defined as the basis for the weights. Pearson correlation is a similarity between two users *u* and *v*, and is measure of how well two sets of data fit on a straight line. The formula of Pearson correlation is:

$$w(u,v) = (\sum_{i \epsilon I}(R_{u,i} - \overline{R}_u)\,(R_{v,i} - \overline{R}_v))/(\sqrt{(\sum_{i \epsilon I}(R_{u,i} - \overline{R}_u)^2)}\sqrt{(\sum_{i \epsilon I}(R_{v,i} - \overline{R}_v)^2)} \quad (2)$$

Where the i ϵ I summations are over the items that both the users *u* and *v* have rated and R_u is the average rating of the co-rated items of the *u*th user (Su and Khoshgoftaar 2009).

For item-based algorithm the formula of Pearson Correlation will be:

$$w(i,j) = (\sum_{u \epsilon U}(R_{u,i} - \overline{R}_i)\,(R_{u,j} - \overline{R}_j))/(\sqrt{(\sum_{u \epsilon U}(R_{u,i} - \overline{R}_i)^2)}\sqrt{(\sum_{u \epsilon U}(R_{i,j} - \overline{R}_j)^2)} \quad (3)$$

where u ϵ U is a set of users who both rated items *i* and *j,* $R_{u,i}$ is the rating of user *u* on item *i,* R_i is the average rating of *i*th item by those users.

Euclidean_distance

To identify neighbours in dimensional space, Euclidean distance (or Pythagorean distance) can be used. Euclidean distance (d_E) defined by equation 4 (Deza 2009):

$$d_{u,v} = ||u - v||^2 = \sqrt{((u_1 - v_1)^2 + \ldots + (u_n - v_n)^2)} \quad (4)$$

where *u* and *v* are two users, and $u_1 \ldots u_n$, $v_1 \ldots v_n$ are ratings of two users to the same items respectively. $d_{u,v}$ gives the numeric value, which represent the distance between two users in dimensional space. The less value of $d_{u,v}$ gives close neighbours, therefore two users possibly have high rate of similarity.

Jaccard index

The Jaccard index, also known as the Jaccard similarity coefficient, is used to measure similarity and diversity of sample sets. The Jaccard coefficient defined as the size of intersection of two sample sets divided by the size of the union of the two samples.

$$\mathbf{J(u,v) = | u \cup v | / | u \cap v |} \quad (5)$$

Where $| u \cup v |$ is a set of videos rated same by user u and v, and $| u \cap v |$ is a set of co-rated videos by user u and v. Dissimilarity between two sample sets is called Jaccard distance, and measured by formula

$$\mathbf{J_d(u,v) = 1 - J(u,v) = (| u \cup v | - | u \cap v |) / | u \cup v |} \quad (6)$$

For example, Jaccard distance between user 1 and 2 from Table 1 will be J(1,2)=0.5, because user 1 and 2 co-rated videos 1 and 3, and gave same rating only for first video. Consequently, jaccard distance between user 1 and 2 is $J_d(1,2)=0.5$.

2.5.1.2 User-based v Item-based approach

Memory based approaches can be divided into user-based approaches and item-based approaches. Both approaches use the same memory based algorithm described above. In the first step of user-based approach the unknown rating is predicted by averaging the weighted known ratings of the test item by similar users, whereas in item-based approach the unknown rating is predicted by identifying similar items.

User-to-User

To predict ratings for videos not yet watched by an active user, a table of k neighbours can be build. The table includes ratings of neighbours for videos the active user has not watched yet and the similarity between the active user and the nearest neighbours. The process of predicting rating includes 2 steps:

1) The ratings of neighbours are multiplied by the similarity, so a neighbour who is similar to the active user will contribute more to the overall score than a neighbour with a low similarity.

2) The sum of the rating multiplied by similarity is divided by the sum of all the similarities for neighbour rated particular video. The result produced by the division gives the predicted rating.

Item-to-Item

To generate prediction based on Item based approach, User-Item matrix can be reversed, and similarity between items first need to be computed. Table 1 need to be reverse as in Table 2-2

	User1	**User2**	**User3**	**User4**
Video1	**1**	**1**		**0**
Video2			**0**	**0**
Video3	**0**	**1**	**1**	
Video4		**0**		**1**
Video5	**1**		**0**	**1**

Table 2-2 Example of user-item matrix for item based approach

To predict rating for users who did watched video yet for video1, the same methodology used for user-based approach can be applied.

Both approaches use the data from user-item matrix to predict unknown ratings by identifying similar users or similar items. Recent research have found the advantages of an item-based approach over a user-based approach (Linden *et al.*, 2003; Sarwar *et al*., 2001.; Herlocker *et al.*, 1999; Breese *et al.*, 1998). Popular e-commerce websites such as Amazon and eBay use the item-based approach as a core method in their recommender systems (Linden *et al*., 2003; Melville and Sindhwani 2010).

2.5.2 Model-based algorithm

A model based approach uses training data to recognise complex patterns and generate a model that is able to make intelligent predictions for the ratings for items that a test user has not rated

before. The model based collaborative filtering algorithms use different machine learning algorithms such as Bayesian network, decision trees, clustering, regression, and rule based approaches (Su and Khoshgoftaar 2009; Mobasher *et al*., 2006; Breese *et al*., 1998).

The Bayesian network model formulates a probabilistic model for collaborative filtering problem (Su and Khoshgoftaar 2009; Breese *et al*., 1998). Clustering models treat for collaborative filtering as a classification problem, as grouping users with a high degree of similarity into clusters, and computing the conditional probability of an estimated rating for the user (Sarwar *et al*., 2001; Breese *et al*., 1998). Rule based approaches use association rule discovery algorithms to identify an association between items purchases at the same time and then generates item recommendation based on the strength of the association between items (Sarwar *et al*., 2001).

Su and Khoshgoftaar (2009) suggest that model based algorithms may not be practical for extremely sparse data. An approach of reducing dimensionality or transforming multiclass data into binary may decrease recommendation performance, and model-building expenses may be high. Therefore, there is a trade-off between prediction performance and scalability for many algorithms. Wang *et al*., (2006) argue that the advantage of the memory-based methods over their model-based alternatives is that less parameters have to be tuned; however, the data sparsity problem is not significantly handled.

2.5.3 Data sparsity

One of the key challenges in the collaborative filtering recommender system is data sparsity. The data sparsity is a measure of how much useful information is available in the data. In this research, data sparsity is measured as the percentage of all possible ratings that exist in the matrix.

***Data sparsity* = total number of ratings / (total number of users * total number of videos** (7)

The alternative measure proposed by (Sarwar et al., 2001) called *sparsity level* measured by equation 8:

***sparsity level* = 1 – (nonzero entries)/(total entries).** (8)

In other words, *spasity level* = 1 – *data sparsity.*

Most recommender systems suffer from the data sparsity problem, and it is still a key challenge for collaborative filtering recommender system (Herlocker*et al*., 1999; Breese *et al*., 1998). Data sparsity in the experiments of previous work in recommender systems has shown the degree between 1% and 10% (Demiriz 2004; Sarwar et al., 2001).

2.6 Content – based approach

Content-based filtering is a widely used approach in designing recommender systems. The algorithm of the system is based on the characteristics of the items, and similar items to the items that were previously liked by the user are recommended. The assumption in content-based filtering is that items with similar features will be rated similarly. An example of movie recommendation as in an introduction can describe the process of content-based recommender system. In order to recommend movies to user *a*, the recommender application searches for the similarities among movies the user has highly rated/watched/favourite in the past. The movies previously unseen and with a high degree of similarity to movies from user profile would be recommended to user. Analysis of the text documents which provide the content and finding regularities in this content are the main issues in content-based algorithms. To evaluate the similarity between items every item is represented by a feature vector or an attribute profile. Machine learning techniques such as Bayesian classifiers, decision trees, and cluster analysis are used as specialised version of classification learners, in which the goal is to learn a function that predicts which class a document belongs to. Other Projects such as Pandora (Howe 2010), Jini

(Joshi and Xu 2003) and Internet Movie Database (Debnath *et al*., 2008) use pure content-based approach in their recommender systems.

One of the examples of a content-based approach used in business is Pandora. Online radio provider Pandora uses content-based approach in their recommender system called The Music Genome Project. In their recommendations, the recommender system uses the properties of a song or artist, and recommends by matching up the user's artist and song likes with other songs that are similar. Even though the method of recommendations employed by Pandora appears to be successful, there is a key challenge for Pandora in classifying songs in their database and building their musical taxonomy. The major issue with the method is scalability. The author suggests few ways to solve the problems such as speeding up the manual classifications, automate the classifications and to determine a trusted way for more people to contribute ratings. But the most promising strategy according to is allowing users to contribute classifications in line with a more collaborative filtering strategy. These issues are the key to moving forward to maintain their recommendation quality while scaling their music library (Howe 2010).

Mooney and Roy (1999) introduced content-based book recommender system called LIBRA. The database of books information from Amazon.com was used in the experiment. Libra uses a naive Bayesian text classifier according to its better results than other methods. The basic results of the experiment were quite encouraging. In order to analyse the results, results tested on the data with combination of collaborative filtering techniques. The experiment results have shown that using collaborative content demonstrated significant advantages according to one or more metrics. The author suggests that information obtained from collaborative methods can be used to improve content-based recommending.

The advantages of content-based techniques over collaborative-filtering include (Ricci *et al*., 2011, p.78-80).

User-independence: Content-based recommenders use only the user's own ratings, whereas collaborative filtering build their recommendations based on the data from other users to calculate similarities, estimate ratings and recommend items.

Transparency: The way a recommendation appears recommendation in the recommendation list can be explained easily in content-based filtering by listing features or descriptions of the recommended items. Those explicit features tend to promote trust in a recommender system, whereas collaborative systems are black boxes, where an explanation of an appeared recommendation is based on the taste of similar user.

New item: Ability to recommend new items which have not yet rated by other user is another advantage of content-based systems. Therefore, content-based filtering systems do not suffer from a first-rater problem. Collaborative filtering systems require items to be rated by a number of users before recommending it to others.

The disadvantages of content-based techniques over collaborative-filtering include (Ricci *et al*., 2011, p.80-82).

Limited-content analysis: Information about item features could not be enough to diversify the items according to the interest of the user. If the analysed content does not contain enough information to discriminate items the user likes from what the user does not like, the system cannot provide reliable suggestions.

Over-specialisation: The content-based method does not find unexpected items. The system analyses the items and compares them to the item from user history, therefore recommends items similar to the items user previously liked, and produces recommendations with a limited degree of novelty. This drawback also called a lack of serendipity.

Hiralall (2011) suggests that the advantage of the content-based approach is that approach does not need knowledge about the domain. Broadly speaking, as this approach uses item features and

compares it with other items, it does not matter what the items is to give recommendations. Attribute representation is very important in content-based approach, and an approach works well if the items can be properly represented as a set of features.

There are a number of limitations of content-based systems (Shardanand and Maes, 1995; Balabanovic and Shoham, 1997). First of all, not all information is capable of useful feature extraction. Feature extraction in fields such as music, movie cannot extract all the valuable information using current technology. Inability of the system to evaluate aesthetic qualities of information is also influence to the feature extraction task. Items must be of some machine parsable form (e.g. text), or attributes must have been assigned to the items by hand. Secondly, there is a problem of over-specialisation. The system is unable to predict the new item which is different to the items have been purchased before. And finally there is a problem of eliciting user feedback. The user's own ratings are only the source of influencing for future performance, as quantity will reduce the performance will reduce in the same way.

However Iaquinta *et al*., (2008) argues that the content-based approach assumption (the user is interested in what is similar to what she/he has already bought/searched/visited) is wrong. The design and implementation of a content-based recommender system that includes serendipitous heuristic in order to mitigate the over-specialisation problem with surprising suggestions proposed in their work.

Content-based filtering method is widely used recommendation technique. It has advantages and disadvantages, and can be applied for a certain type of projects.

2.7 Text analysis

85% of business information exists in the form of text (Hotho *et al.,* 2005). An ability to analyse text and discover knowledge in the text could increase an understanding of obtained data and a

proper use of the information within the organisation. Text mining refers to the process of deriving high-quality information from text, and focused on discovering hidden knowledge and interesting patterns from unstructured textual data. Solka (2008) defines text data mining as "concerned with data mining methodologies applied to textual sources". Text mining is related to areas such as information extraction and inherits their methods and techniques in the process of extracting patterns from texts. Information extraction techniques are widely used for tasks such as document matching, ranking, and clustering. The process of text analysis can be divided into 3 steps: structuring input text data, deriving patterns in the data, and evaluation and interpretation of the produced data. There are different text mining tasks such as document classification, text clustering, document summarisation and sentiment analysis. Data mining methods, such as classification and clustering are widely used in text mining to structure the collection of documents and identify hidden knowledge. The challenge in text mining is linguistic knowledge base and representation of this knowledge. Most of applications need more structured linguistic base for precisely knowledge extraction.

Pre-processing techniques for text analysis task includes making decision for each word or phrase in order to give single label to entire document collection or sentence. Luhn (1958) suggested that the significance of a particular word can be measured by its frequency in a document. Feature extraction included stemming and removing stop words.

Solka (2008) defines stop words as "common words that do not add meaningful content to the document" and stemming as "the process of removing suffixes and prefixes, leaving the root or stem of the word". The pre-processing step is to prepare text in a format which can be analysed by text mining methods in order to produce a result.

Sentiment analysis or opinion mining is the identification and extracting subjective information from source materials in natural language processing and text analytics applications. The aim of sentiment analysis is to identify the attitude of the person with respect to the opinion in a

document, sentence or an entity. Expressed attitude or opinion of a person can be positive, negative or neutral (Feldman and Sanger 2006). There are two most popular approaches in sentiment analysis – lexicon and supervised machine learning (classification). To evaluate the textual expression a list of polar words and phrases such “like”, “dislike”, “good”, “bad”, and so forth are required. Lexicons or dictionaries contain a list of polar words which recognise text as “negative” or “positive”. Sentiment classification task labels document expressions or sentences as overall positive or overall negative (Pang and Lee 2008). One of the simplest ways of textual representation is “bag of words” approach. In this approach, text is represented as a collection of unordered words, where grammar and words order is not considered. The bag of words approach can be used in sentiment classification, where each word represents some numeric value, and overall value of the text will be sum of this values. To transform to classification problem, bag of words approach will use a dictionary, a list of prepared words representing positive and negative expression, and each word in text will be matched with dictionary. For example, there are two expressions, and a dictionary with positive and negative words.

Positive words: like, nice, awesome

Negative words: bad, worse, awful

1. I like this game, it is nice.

   [“I”:0, “like”:1, “this”:0, “game”:0, “it”:0, “is”:0, “nice”:1] = 1+1 = 2
2. It was awful.

   [“It”:0, “was”:0, “awful”:-1] = -1

Each word in the two examples sentences is compared against the dictionaries, and if they match a positive or negative score is returned. All other words not in dictionary were assumed to be stop words, and have value 0 as not giving any emotional value. The final value for each sentence is a sum of scores of each word returned from a dictionary. The overall score is the

sentiment for the sentence, where positive score (all scores higher than 0) indicates positive opinion and negative score (all scores less than 0) indicates negative opinion.

Text analysis aims to identify hidden knowledge in textual information. Analysis of user comments for a particular item can give additional information about user preferences for these items and can be used to increase explicit data used in the recommender systems.

2.8 Conclusions

This chapter has discussed the domain of recommender systems, their functions and types. The definitions and purposes of recommender system roles in business have been analysed. The main functions of recommender system have been mentioned and described. The importance of obtained data and knowledge sources in order to build recommender system has been discussed. A role of information types of data such as explicit and implicit data used to build recommender model have been discussed.

The purposes and objectives of the recommendation techniques such as collaborative filtering, content-based, demographic, knowledge-based, community-based, and hybrid have been compared and analysed. It is clear from this research that the recommender system can have a positive effect on revenue for an online company, however it is important as demonstrated here that the choice of appropriate recommender technique is important to generate recommendations that satisfy user needs.

The most popular recommendation technique, collaborative filtering has been analysed more broadly. Model based and memory based approaches of collaborative filtering have been defined. The high performance of user based and item based approaches have been proved by success of e-commerce websites such as Amazon and Netflix. The process of predicting ratings based on different similarity metrics such as Pearson correlation, Euclidean distance and Jaccard

coefficient has been clarified. The key challenges of building recommender system based on collaborative filtering have also been mentioned. The definition of data sparsity has been discussed. This research has shown advantages of using collaborative filtering method on real examples, the process of creating recommendations based on different approaches and similarity metrics.

Examples of recommender systems based on content-based technique have been discussed above. The process of building recommendations, the types of data used to build it, advantages and disadvantages of the approach have been analysed. Comparison between collaborative filtering and content-based filtering techniques revealed advantages and disadvantages of each technique according to the purpose of recommender system and available data. This section indicates the key challenges in content-based recommender system, and discusses how content-based approach can be used in hybrid systems to improve the performance of recommender system.

Text analysis is a part of data mining, aimed at revealing unseen patterns in unstructured data. The process and techniques used to reveal knowledge from textual information have been described. It is clear from this research that a possibility to obtain additional information from textual data can increase the knowledge of the provided data.

Based on the research in this chapter the following recommendations have been made concerning the experiment design:

1) Collaborative filtering technique approach is to be used in the experiment. Information about user's "likes" and "favourites" will be used as explicit user ratings.
2) Similarity metrics based on Euclidean distance and Jaccard distance will be used in the experiment. Pearson correlation coefficient does not suit to this project, according to the data type (likes and favourites) available in the dataset, which is binary.

3) The usage of text mining to analyse the comments of users on particular video to suggest whether they liked or disliked it, can provide more information for recommendation. The combination of data about user's likes, favourites and their comments on specific videos can reduce data sparsity and give more information on which to determine the user's rating for an item.

3. EXPERIMENT DESIGN AND METHODOLOGY

3.1 Introduction

In this project the most popular approaches to recommender systems have been analysed in the literature review chapter. Among these approaches collaborative filtering, content-based filtering and hybrid systems have shown the higher performance compared to other approaches. Many of successful commercial websites such as Amazon, eBay, MovieFinder and so forth use one of these approaches or a combination of them (Linden *et al*., 2003; Schafer *et al*., 1999).

The benefits of a collaborative filtering over content-based filtering, and the challenges of content-based approaches were mentioned in chapter 2.5. The hybrid recommender systems have shown good results in existing research, but it should be mentioned that these types of recommender systems require much more computational power, and there is a trade-off between accuracy and performance of the recommender system. According to the advantages of the collaborative filtering approach and taking into consideration the structure of the data for this research project the collaborative filtering approach has been chosen as the most appropriate above others.

The purpose of this section is to describe structure of the data used in the project and the experimental methodology. The data set section includes information about the data set built from the original data and used in the experiment. The provided data gives information on the user's opinion of videos which a rating for the video will be extracted and used in the recommender system.

3.2 Data set

The kiwi.kz website does not have a rating system for their videos. Each video on the website consist title, tags and the name of the user uploaded video. Registered users on the website can

express their attitude to the videos by clicking buttons “like” and/or “add as favourite”. The number of “likes” for a video is displayed under the video, and shows how many times it was liked by users. The videos added as “favourite” stored in user’s profile, and can be easily find by user. Also users can leave comments on the video, which gives more information on the opinion of the user. Structure of comments is constructed in a way, where comments for videos is displayed on the left side of a page, and replies for comments between users displayed in a form of tree, one under another. The original data provided by the company consists of 7 different files about users, videos and their user histories. The data provided by the company consisted explicit and implicit types of data for a 2 month of period.

The data consisted of information about 998 users, 105412 videos, 39081 watched videos by users, 11557 comments of users, 2312 videos added as favourite, and 384 liked videos. The information from the data allows to identify what watched videos users had liked, favourite or entered comments. In order to build recommender system the original data was transformed and new files were constructed. 3 different files based on user likes, favourites and comments were constructed. The first file includes information about user’s attitude to particular video by indicating whether it was only watched or watched and liked. First file stores information about users who expressed their attitude to videos by liking it at least once. It means user who only watched videos does not included to the file.

The second file was constructed for users who expressed their attitude to particular videos by adding them as favourite. Again, only users who added videos as favourite at least one time were included. The third file included information about users who entered comments for particular videos. Information about users who left their comments under other user comment was assumed as discussion between users, and was removed. Only comments left under video was considered in the third file and consisted of 6742 comments. The full names, attributes and transformation process of each file in the data are provided in the Appendix B.

The data set represents the attitudes of users to particular videos in different ways. Users can like them, add as their favourite or comment on it. A key assumption in the experiment lies in the classification of the user's opinion based on the historical data. The attitude of users to videos can be classified as "positive", "negative" or "neutral". Analysis of the data set allows us to classify the data into three classes. These classes will be used to build user-item matrices which will include ratings for the videos for specific users. Different recommender systems will be built on the different matrices to see which data provides the best information on user opinions to use for providing the best recommendations. Four different user-item matrices were built, these are explained below:

Information about videos liked by users can be classified into two classes. Videos only watched by users cannot reveal "positive" or "negative" opinion for videos, and can be classified as "neutral". Users who "liked" videos express their opinion to videos in a positive way; therefore can be classified as "positive". Based on this classification task, user-item matrix based on videos liked by users can be built. The ratings in the user-item matrix for videos watched and liked by users represent "positive" attitude and have a value of "1".

Videos only watched by users in the user-item matrix represent "neutral" attitude and have a value of by "0". Blank spaces in the matrix represent that a video has not watched by users, therefore there is no opinion and rating for these videos. The detail about the user-item matrix where rating is based only on the likes is provided in Tables 3-1.

Number of user who liked video	86
Number of videos	7628
Number of ratings	9862
Data Sparsity	1,5%
Ratings values used	0, 1

Table 3-1 User-Item matrix, where rating is user "likes"

The user-item matrix includes 86 users who watched 7628 videos and liked some of them. The total number of ratings in the user-item matrix is 9862 (“0” for neutral and “1” for positive). Data sparsity in the user-item matrix is 1,5%, and indicates that data is sparse.

Information about videos added as favourite can be classified in the same way as videos liked by users. The attitude of users who watched and added videos as favourites can be classified as “positive”, and attitude to videos watched but not rated as favourite can be classified as “neutral”. The user-item matrix built from file about videos added as favourite videos have includes ratings “0” and “1” which represent “neutral” and “positive” attitude for videos respectively. Blank spaces in user-item matrix give information about videos not watched by users. Table 3-2 provides details about user-item matrix where ratings are “favourites”.

Number of users	369
Number of videos	10712
Number of ratings	16281
Data Sparsity	0,4%
Ratings values used	0, 1

Table 3-2 User-Item matrix, where rating is user favourites

The total number of users, videos and ratings in user favourites file is higher than in user likes file. The data sparsity in user favourites file is 0.4%, which indicates high level of sparsity in the user-item matrix. Data sparsity in user likes file is 1,5% and higher than in user favourites file due to the less total number of users, videos and ratings.

The further user-item matrix is built from a combination of information about videos which were “liked” and “favourite” by users. The aim of combination of these two attributes is to decrease the imbalance of ratings. Ratings in the both user-item matrixes for “likes” and “favourites” are very imbalanced. Only 4% of ratings in likes and 14% of ratings in favourites have rating value of “1”. Users who liked and/or favourited videos express their attitude to videos as “positive”. However, the nature of videos added as favourite is different from liked videos. The videos

added as favourite are stored in user profile, and it is suggested that people tend to add videos ass favourite videos to keep them and to be able to find them easily when they need them. Therefore, it is suggested that the "positive" ratings from user-item matrix for favourite videos have a higher value than the "positive" ratings from liked videos, and have given rating of 2. The new user-item matrix has ratings, where "0" represents "neutral" attitude for only watched videos, "1" represents videos only liked by users, "2" represents videos added as favourite and rating "3" is given for videos added as likes and favourites. Details of the user-item matrix constructed using information from both user likes and favourites are shown in Table 3-3.

Number of users	388
Number of videos	11549
Number of ratings	17877
Data Sparsity	0.35%
Ratings values used	0, 1, 2, 3

Table 3-3 User-Item matrix, where rating is combination of user likes and favourites

The number of users, videos and ratings is increased comparing to only user likes and favourites. However the data in the constructed user-item matrix is more sparse, and has value 0,35%, which is less than in the user – item matrix of user likes (1,5%) and user favourites (0,4%). The new table has been built by using MySQL and full code is provided in Appendix C.

A further user-item matrix was built from data that used a combination of available explicit types of data (user's likes and favourites) and implicit data type derived from comments. Semantic analysis on comment file has been done to generate a numeric score for the video which represents the user's opinion as expressed in the comment. The comments for videos can represent "positive", "negative" or "neutral" attitude of users. All scores higher than "0" represent "positive" attitude and scores less than "0" represent "negative" attitude for videos. The score with value "0" express "neutral" attitude for videos.

The file including user comments has been used to identify attitude for videos. The semantic analysis was done with the R statistical tool and was done as follows:

Two vocabularies or list of words with positive sentiment and negative sentiment respectively were extracted manually from the comments. The lists contain words in 3 different languages (Kazakh, English and Russian) which are written in two different alphabetic writing systems (Cyrillic and Latin) according to the nature of comments. The full list of positive and negative words contains 365 positive and 116 negative words and is included in Appendix C.

The sentences in the comments were parsed and tokenised into words to identify positive or negative opinion for a whole comment. The total number of comments to videos in the data was 6742. Each word from the comment was checked against the positive and negative lists and scores were accumulated, a positive score for a positive word and a negative score for a negative word. Initial transformation has given numeric values representing user opinion from -7 to 3.

Class	Positive			Neutral	Negative					
Score	3	2	1	0	-1	-2	-3	-4	-5	-7
Number of ratings	7	66	936	4890	722	95	17	6	1	2

Table 3-4 Initial sentiment scores of comments

To reduce the effect of outliers, all positive resulting scores were given an overall sentiment score of 1, whereas all negative resulting scores were given an overall sentiment score of -1.

Class	Positive	Neutral	Negative
Ratings values used	1	0	-1
Number of ratings	1009	4890	843

Table 3-5 Overall sentiment scores of comments

The last user-item matrix is built from a combination of user-item matrix of likes with favourites and with user-item matrix from comments file, where ratings are scores representing attitude of users to videos in comments.

Ratings in the new user-item matrix were created by comparing user-item pair in both matrixes. If a video from likes and favourites has also been commented on by the same user, the new rating in user-item matrix is became a sum of the ratings from the two matrixes. To remove negative values in the user-item matrix, all ratings in the user-item matrix are incremented by 1.

Class	Negative	Neutral	Positive		
Ratings	0	1	2	3	4
Number	758	19301	1222	2034	152
Percentage	3,2%	82,2%	5,2%	8,7%	0,7%

Table 3-6 Ratings distribution of combined user-item matrix

The generated user-item matrix has ratings, where "0" represents "negative" attitude for watched videos, "1" represents "neutral" attitude for watched videos and ratings with values "2", "3", "4" represent "positive" attitude for watched videos. The full code of the transformation processes and lists of positive and negative words are described in Appendix C and the process of transformation comments into score is described in Appendix B.

3.3 Experimental Methodology

The objective of this project is to build different recommender systems using real world data and to evaluate the accuracy of the predicted recommendations. Each experiment will use a specific user-item matrix as described above to see whether good recommendations can be predicted and to see which data can produce the best recommendations.

The Python software programming language has been chosen for this experiment as one of the most effective tools. Python is a convenient tool used for implementing recommender systems.

(Segaran 2007; Celma 2010; Caraciolo *et al*., 2011) contain examples of existing recommender systems based on Python.

The data from the user-item matrices has been divided into a test set and s training set. 20% of the data has been used as a test set and 80% of the data has been used as a training set in each experiment. 80% of the data in the training set used to predict ratings for the ratings in the test set. Evaluation of predicted ratings was based on accuracy of predicted ratings comparing with actual ratings in the test set. Information about the number of ratings used in the test data used in each experiment is provided in Table 3-7.

User-item matrix, where ratings are:	The number of ratings in the test set
LIKES	1973
FAVOURITES	3257
LIKES AND FAVOURITES	3579
LIKES, FAVOURITES AND COMMENTS	4756

Table 3-7 Number of ratings used in the test data

The experimental process for each experiment is follows:

1. Load user-item matrix into Python programming tool

User-item matrix including user ID, video ID and ratings are upload to software for further analysis. The software stores information about user preferences and produce recommendations based on the information defined by user in steps 2, 3, and 4.

2. Choosing similarity metric

The similarity metric needed to be defined for further identification of similar users or items. The first step in memory-based prediction algorithms is to weight all users with respect to similarity with the active user. Similarity weighted measures such as the Jaccard coefficient and Euclidean distance have been chosen to evaluate similarities between similar users to build recommendations. Pearson correlation coefficient has not been used in this experiment due to the ratings in the user-item matrices which are binary.

Pearson correlation coefficient measures the strength of the association between the two variables and gives high results on interval data, whereas the data used in the experiment is binary.

3. Define memory based collaborative filtering approach

Similarity metric chosen in step two will be used to calculate similarity between users or items. Memory based approaches such as user based and item based approaches need to be chosen in the third step. In order to suggest items for test user or item the measure of the weight between test user or item and neighbours is calculated. Similarity between users based on metric chosen in the second step is calculated in the user based approach and similarity between items is calculated on item based approach.

4. Choose the test user and calculate top N neighbours using the selected similarity measure

A test user needs to be chosen in the step 4. Predicted rating for item is depended from the number of neighbours used to calculate weighted sum of ratings. The similarity metric chosen in step 2 is used to measure the weight between the test user and the neighbours. Number of neighbours with the highest similarity is needs to be defined in order to calculate predicted ratings.

5. Generate recommendations based on top N neighbours.

Top N neighbours with the highest similarity with the test user are used to compute a prediction for videos the test user has not watched yet. The predicted rating of the test user for item is a weighted sum of ratings of the top N neighbours (Equation 1 in 2.5).

6. Evaluate the performance of recommender system.

Perform an evaluation of how recommender system predicts ratings for the test user. The evaluation metric used in evaluation process is described more broadly in section 3.6.

3.4 Evaluation metric

User's satisfaction with recommendation depends on how well and effectively collaborative filtering methods work on the data. The evaluation of algorithms will be based on an evaluation metric. To identify how well recommender system predicts recommended videos, accuracy has been chosen as the evaluation criterion. Accuracy assessment is an important step in the classification process. The aim is to quantitatively identify how effectively classes were grouped in the experiment under investigation. The accuracy is the degree to which repeated measurements under unchanged conditions show the same results (Feldman and Sanger 2006). Accuracy measures the percentage of recommendations which was correctly predicted.

Accuracy = Number of correct recommendations / (Total number of recommendations) (8)

Average class accuracy has shown less sensitivity to highly imbalanced ratings than overall ratings (Ismail and Jusoff 2008). Average class accuracy provides more precise accuracies than overall accuracy in cases, where the amount of information in one class is much higher than in others. The average class accuracy is the average of the accuracies for each class. For example, for "positive", "negative" and "neutral" classes, the average accuracy is an average of total accuracy of each class.

Accuracy (Positive) = Number of correct positive recommendations/Total number of positive recommendations (9)

Accuracy (Neutral) = Number of correct neutral recommendations/Total number of neutral recommendations (10)

Accuracy (Negative) = Number of correct negative recommendations/Total number of neutral recommendations (11)

The accuracies of each class first need to be computed and then average class accuracy is found by division sum of each of the class accuracies to the sum of classes.

Average class accuracy = (Accuracy (Positive) +Accuracy (Neutral)+ Accuracy(Negative))/Sum of classes (12)

Fitzgerald *et al.* (1994) claims that the average class accuracy has been shown to be statistically more sophisticated measure of classifier agreement and therefore gives better interclass discrimination than overall accuracy.

3.5 Conclusion

This chapter has discussed the structure of the design of the experiment, the data used in the experiment, and the methodology of building and evaluation of recommendations.

Information about the original data, the amount of information in the data and transformed data used in the experiment has been discussed. Details of the data with rating based on likes and favourites are provided. The problem of data sparsity and imbalance in the ratings distribution is discussed.

Analysis of creating more complex user-item matrices built from combination of available explicit and implicit data and possibility to handle data sparsity and imbalance in ratings distribution problems has been discussed.

Explanations of selection criteria for similarity metrics used in the experiment have been written.

Definitions of accuracy and average class accuracy used in the evaluation have been explained.

Experimental methodology has been analysed. The process of building recommendation, where prediction rating is based on the weighted sum of the ratings from the neighbours with the highest similarity is defined. The importance of evaluation algorithm also been discussed.

4. EVALUATION AND RESULTS

4.1 Introduction

The objective of this chapter is to implement several tests on the real data to identify the most suitable collaborative filtering approach and examine the possibility of increasing performance of the recommender system by combining the user ratings data from both explicit sources (such as likes and favourites) and implicit sources (user comments). This chapter presents experimental results of applying collaborative filtering techniques on the different user-item matrices for generating predictions. Three experiments have been done in order to evaluate the performance of recommender system based on accuracy.

The first experiment involved two separate user matrices – the one built from ratings based on user likes and the other built from ratings based on favourites. The aim of the experiment is to identify the best performing collaborative filtering approach. This experiment choosing between memory - based approaches such as user based and item based approaches, and similarity measures such as Euclidean distance and Jaccard coefficient. Based on the highest average class accuracy in the experiment results with user likes and favourites, the best approach was then used in experiment two and three.

The second experiment includes testing the best approach from the first experiment on the user-item matrix build from a combination of likes and favourites ratings. The aim of combination of those two user-item matrices is to reduce the imbalance in the distribution of the ratings.

The aim of the third experiment is to identify how a combination of explicit and implicit types of data can influence to the performance of recommender systems. The third experiment involved running a test on the data build from combination of explicit and implicit type of data. The ratings from user-item matrix built from combination of user-item matrices of likes and favourites used as explicit form of ratings.

The semantic score derived from user comments on videos has been used as implicit form of rating in the experiment. A text analysis of comments, where positive and negative words in each comment counted has been used to derive semantic score for each comment. The sum of scores represents the attitude of a user to a video, and used as implicit data in the experiment.

Average class accuracy has been used as the main assessment parameter in the experiment. The distribution of the ratings in the data is much imbalanced, therefore average class accuracy is chosen than overall accuracy.

Parameters of the experiment were determined before the experiment. The data is divided into a training set and a test set. 80% of the data was used as training set and 20% of the data was used as test set. Accuracy of predicted ratings from training data on the ratings from the test data is measured to identify the performance of the experiment.

Experimental platform: All experiments were implemented using Python and run on PC with Intel Core Duo processor having 4Gb of RAM and a speed of 1.7GHz.

4.2. Experiment 1

The first experiment includes building recommendations based on two different user-item matrices in order to identify the best collaborative filtering approach. Experiments using data on user likes and favourites have been performed using user based and item based approaches. Euclidean distance and Jaccard coefficient have been used to compute similarity between users and items in memory based approach.

4.2.1 Experiment based on user likes

The first experiment uses the data where user express their attitude to videos by liking it. The experiment includes test of two different collaborative filtering techniques (Euclidean distance

and Jaccard coefficient) based on two memory – based approaches (user based and item based approach).

The distribution of ratings, where the ratings are built from user likes:

Class	Neutral	Positive
Rating	0	1
Total number	9484	378
Percentage	96%	4%

Table 4-1 The distribution of ratings, where the ratings are built from user likes

Data sparsity in user likes file is 1.5% and the distribution of ratings is much imbalanced, where 4% of the total ratings were "positive" and 96% were "neutral" ratings. The data was divided randomly into a training set and a test set, where 80% of the ratings used as a training set and 20% of the ratings used as a test set. The predicted rating where compared with actual ratings and accuracy of prediction has been used to identify the performance of used techniques. The actual ratings in the user-item matrix have ratings "0" and "1" representing "neutral" and "positive" attitude respectively. The predicted ratings are lie in the scale between 0 and 1. The sliding-scale in Figure 1 represents the distribution of predicting rating value.

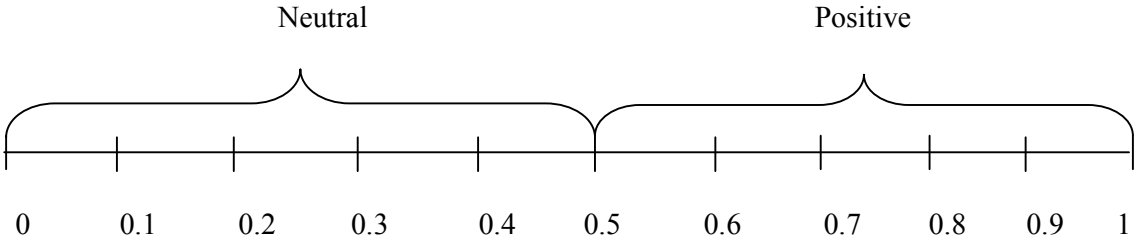


Figure 4-1 The sliding-scale representing the distribution of predicting rating values for the first experiment

The threshold has been used to identify the attitude of a user. Threshold for predicted ratings is used to find whether a user find a video "positive" or "neutral". Threshold of 0.5 has been

chosen in evaluation process. Predicted ratings equal to or greater from 0.5 were attributed as "positive" attitude, and predicted ratings less than 0.5 were attributed as "neutral" attitude.

Average class accuracy and overall accuracy have been calculated for predicted ratings. In the experiment, average class accuracy is an average of accuracy of predicted "positive" and "neutral" class accuracies. Overall accuracy is an accuracy of a whole test data. Due to the very high imbalanced ratings distribution, where only 4% of ratings are positive, average class accuracy has been chosen as more reliable evaluation assessment over overall accuracy.

The evaluation based on user based and item based approaches with a threshold 0.5 compared two similarity metrics Euclidean distance and Jaccard coefficient. Details of the evaluation results are in Table 4-2.

Memory-based approach	Accuracy	Euclidean distance	Jaccard coefficient
User-based approach	Accuracy(Positive)	14,02%	11,72%
	Accuracy(Neutral)	99,97%	97,02%
	Average class accuracy	**56,99%**	54.37%
	Overall accuracy	94,86%	93,72%
Item-based approach	Accuracy(Positive)	8,74%	6,96%
	Accuracy(Neutral)	99,65%	97,76%
	Average class accuracy	54,19%	52,36%
	Overall accuracy	94,49%	93,02%

Table 4-2 Details of the user likes experiment evaluation results

The average class accuracy of user based approach based on Euclidean distance is 56,99%, which was calculated from the accuracy of predicting positive ratings (14,02%) and the accuracy of predicting neutral ratings (99,97%). Overall accuracy is higher than average class accuracy,

but it does not reflect the accuracy of positive recommendations in a fair way. The high result of overall accuracy is influenced by the accuracy of "neutral" class accuracy.

The average class accuracy for collaborative filtering techniques based on Euclidean distance has shown slightly higher results than collaborative filtering techniques based on Jaccard coefficient in both user based and item based approaches. The user based approach has shown higher results based on two different similarity metrics in both overall and average class accuracy metrics. This is surprising considering that item-based approaches typically outperform user – based approaches for companies such as Amazon and eBay (Linden *et al.*, 2003; Melville and Sindhwani 2010). This can possibly be due to the nature of video data which is very sparse and the ratings in the data are binary. Most of collaborative filtering recommender systems in companies such as Amazon, eBay, Netflix and so on provide ratings on a scale from 1(disliked) to 5(liked) to assess the quality of interaction between users and items (Melville and Sindhwani 2010; Linden *et al.*, 2003).

4.2.2 Experiment based on user favourites

The ratings built from the user favourites are used in the experiment. The methodology and process of experiment is the same as the first experiment where ratings are built from the user likes. Both user-item matrices have binary ratings, representing "neutral" or "positive" classes.

Class	Neutral	Positive
Rating	0	1
Total number	14074	2207
Percentage	86%	14%

Table 4-3 The distribution of ratings, where the ratings are built from user favourites

Comparing to user – item matrix of user likes, the user-item matrix based on favourites is sparser. However, the imbalance in rating distribution in favourites file is less. 14% of ratings in favourites are positive whereas only 4% of ratings in likes are positive.

The data was divided into 20% of a test set and 80% of a training set, where the number of ratings in the test set was 3257.

Evaluation of predicted ratings based on the Euclidean distance and Jaccard coefficient similarity is provided in Table 4-4.

User-based approach	Accuracy	Euclidean distance	Jaccard coefficient
	Accuracy(Positive)	11,03%	9,85%
	Accuracy(Neutral)	98,89%	97,24%
	Average class accuracy	**54.96%**	53,54%
	Overall accuracy	85,24%	84,87%
Item-based approach	Accuracy(Positive)	5,95%	5,01%
	Accuracy(Neutral)	99,25%	98,89%
	Average class accuracy	52,60%	51,95%
	Overall accuracy	85,11%	83,76%

Table 4-4 Details of the user favourites experiment evaluation results

Collaborative filtering technique using user based approach with Euclidean distance and Jaccard coefficient has shown slightly higher results than item based approach using same similarity. Accuracy of predicted ratings based on Euclidean distance has shown better results in both user based and item based approaches.

User based approach has shown higher result than item based approach in predicting "positive" class ratings, 11,03% and 5,95% respectively. The user based collaborative filtering with Euclidean distance as similarity metrics has shown the highest result than other approaches, with average class accuracy 54,96%.

Comparison between performance of recommender system based user-item matrixes where ratings are likes and favourites has shown that average class accuracy and overall accuracy of predicted ratings in the user-item matrix of user likes is higher than in the user-item matrix of user favourites. Average class accuracy of the user-item matrix from user likes based on user based approach with Euclidean distance as similarity metrics has shown almost 2% better results than the same approach in user – item matrix of user favourites. Considering that the number of user and videos in the user – item matrix of user favourites is higher and therefore data sparsity is higher than in user likes, it is understandable that performance of recommendations based on user likes is higher.

The results from experiment 1 have shown that user based approach of collaborative filtering using Euclidean distance performs better than other approaches. Based on the received results, user based collaborative filtering technique using Euclidean distance as similarity metric has been chosen as a main approach for the second and third experiments.

4.3 Experiment 2

The objective of the second experiment is to test how a combination of two available explicit rating types can impact on the performance of predicted recommendations. User likes and favourites file were combined to build one user – item matrix with more diverse ratings. Based on the results from 1 experiment, user based approach with Euclidean distance as similarity metric has been used to build and evaluate recommendations.

Constructed user – item matrix attempts to decrease imbalance of ratings distribution in order to improve the quality of recommendations. According to the nature of the data from user favourites, all rating s from this file have rating value 2. If a user liked and added as favourite the same video, the new rating for this video will be a sum of the likes and favourites ratings which is equal to value 3. All ratings equal or higher than 1 represent "positive" attitude for videos from users. Just watched videos have rating 0 and express neutral attitude to the videos.

The distribution of ratings, where the ratings are built from combination of user likes and user favourites is provided in Table 4-5.

Class	Neutral	Positive		
Rating	0	1	2	3
Total number	15340	331	2160	46
Percentage	85,8%	2%	12%	0,2%

Table 4-5 The distribution of ratings, where the ratings are built from combination of user likes and user favourites

The number of users, videos and ratings is increased by combination of user likes and favourites user-item matrices. The imbalance of the ratings distribution in the generated user – item matrix is almost the same with the user-item matrix of user favourites. The scale of ratings is increased from 0 to 3, but the imbalance of positive and neutral ratings are still around 86% for neutral and 14% for positive ratings.

In order to predict ratings 80% of the data was used as a training set and 20% of the data was used as a test set. The number of ratings in the test set was 3576. All predicted ratings were classified into "neutral" and "positive" classes.

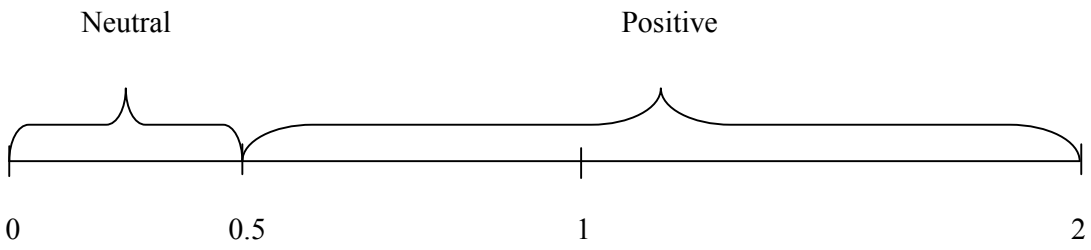


Figure 4-2 The sliding-scale representing the distribution of predicting rating values for the second experiment

The sliding-scale in Figure 2 represents the distribution of predicting rating values. The predicted ratings less than 0.5 are attributed as "neutral", and all the ratings equal or higher than 0.5 are attributed as "positive" ratings. Accuracy of each class is computed by identification how many of predicted "positive" or "neutral" ratings are in the actual ratings respectively.

	User-based approach
Accuracy(Positive)	12,43%
Accuracy(Neutral)	98,22%
Average class accuracy	55,32%
Overall accuracy	86,23%

Table 4-6 Details of accuracy of predicted ratings based on user based approach with Euclidean distance as similarity metric in combined user-item matrix

Accuracy of "positive" class ratings is 12, 43% and it is higher than in favourites user – item matrix, but is less than in user – item matrix of user likes. Average class accuracy of the combined user – item matrix is also higher than in user favourites and less than in user likes, with a value 55, 32%, whereas average class accuracy in favourites and likes is 54, 96% and 56, 99% respectively. The overall accuracy is also shows higher results than overall accuracy in user favourites and less than overall accuracy in user likes.

The interesting thing in the second experiment is that even if the data became much sparser than in favourites file, the accuracy of prediction is higher. The possible answer could be that the distribution of ratings is more diverse, and scale 0-3 was introduced.

4.4 Experiment 3

The last experiment involved the combination of explicit data type from user likes and favourites and implicit type of data derived from user comments. Text analysis has been done on user comments on particular video, to identify whether user has "positive", "negative" or "neutral" attitude to the video. Sentiment analysis using lexicon has been used on comments file, and the score representing attitude of a user to a video has been derived. This score is based on a count of the positive and negative words in the comment, with +1 for each positive word and -1 for each negative word. The initial transformation is given in Table 4-7, where a score represents the attitude of a user to a video.

Class	Positive			Neutral	Negative					
Score	3	2	1	0	-1	-2	-3	-4	-5	-7
Number of ratings	7	66	936	4890	722	95	17	6	1	2

Table 4-7 Initial sentiment scores of comments

The score with a value "0" represents "neutral" opinion for a video by user. The scores with a value higher or equal to "1" represent "positive" opinion for a video, whereas scores with a value less or equal to "-1" represent "negative" opinion for a video by a user. To avoid impact of outliers and scores with relatively high scores, all scores higher than one classified as "positive" and given score "1", and all scores less than "-1" classified as "negative" and has given score "-1". Table 4-8 shows comments ratings with transformed sentiment scores of -1, 0, and 1.

Class	Positive	Neutral	Negative
Ratings	1	0	-1
Number of ratings	1009	4890	843
Percentage	14,96%	72,53%	12,51%

Table 4-8 Overall sentiment scores of comments

The combination of comments user-item matrix with a combined user-item matrices of user likes and favourites created by merging the data from tables 14 and 17, to avoid negative values in the data, each rating was incremented by 1 to build a rating with 0-4 scale.

Number of user	576
Number of videos	14041
Number of ratings	23467
Data sparsity	0.25%
Ratings values used	0, 1, 2, 3, 4

Table 4-9 User-item matrix where ratings are represented by combination of likes, favourites and sentiment scores of comments

Ratings in user-item matrix are slightly less imbalanced than in user-item matrices with likes, favourites and combination of them. The number of neutral ratings is decreased to 82, 2%, whereas overall of positive ratings is slightly increased up to 14,6%. The new class of negative ratings were introduced with a total of 3, 2% of all ratings. However, data is much sparser than in previous user-item matrices.

Class	Negative	Neutral	Positive		
Ratings	0	1	2	3	4
Number	758	19301	1222	2034	152
Percentage	3,2%	82,2%	5,2%	8,7%	0,7%

Table 4-10. The distribution of ratings in the user-item where ratings are combination of likes, favourites and sentiment scores of comments

At the end the user-item matrix includes explicit data about preferences for videos from their likes for videos, adding videos as their favourite and analysing comments as to whether comments whether user liked or disliked it. In Table 19, all ratings equal or over 2 represent possibility that user tend to like the videos with rating higher than 2. Ratings with value of 1 shows neutrality, and 0 value negative attitude for videos.

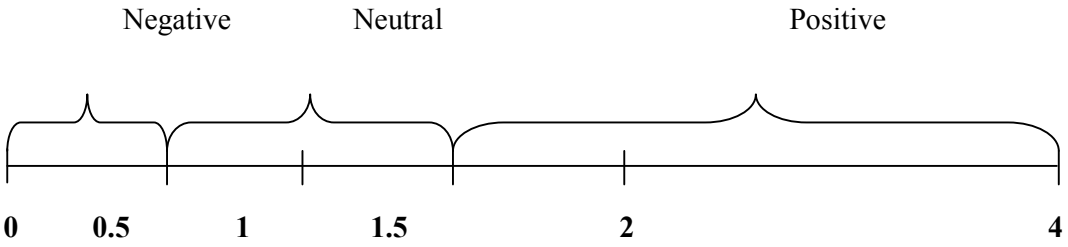


Figure 4-3 The sliding-scale representing the distribution of predicting rating values for the third experiment

Evaluation of performance of user based approach based on Euclidean distance has been done. Threshold 0.5 for predicted ratings value has been used to classify predicted ratings. Average class accuracy evaluated on a test data, which consisted of 20% of the data, and had 4756 ratings. The results of evaluation are in Table 4-11.

	User-based approach
Accuracy(Positive)	18,84%
Accuracy(Neutral)	100%
Accuracy(Negative)	58,11%
Average class accuracy	58,98%
Overall accuracy	87,79%

Table 4-11. Details of predicted ratings with Euclidean distance as similarity metric

The average class accuracy is an average of a accuracy of 3 classes: positive, neutral and negative, and has a value of 58, 60%. Accuracy of predicted positive class has a value of 18, 26%, accuracy of "negative" class has a value of 57, 65% and accuracy for neutral class is 100%.

4.5 Conclusion

This chapter has examined different collaborative filtering techniques on the different user-item matrices from the real data. Number of experiments was run to identify the best approach of collaborative filtering and to determine which combination of available data can produce higher results. Average class accuracy has been chosen as main evaluation criteria due to high imbalance in the distribution of ratings.

One of the main challenges in producing accurate predictions was data sparsity. The data provided by the company consisted explicit and implicit types of data for a 2 month of period.

	DATA SPARSITY	RATINGS
LIKES	1.5%	0,1
FAVOURITES	0.4%	0,1
COMBINATION	0.35%	0,1,2,3
COMMENTS	0.25%	0,1,2,3,4

Table 4-12 Data sparsity of files used in the experiment

Distribution of ratings in the likes and favourites files also has been much imbalanced. The combination of likes and favourites files in the second experiment, and introducing implicit data by combining sentiment scores from comments file in the third experiment attempted to increase rating distribution and improve the range in ratings scale. With increasing number of users and videos, the scale of rating were increased, which revealed interesting results in evaluation of predicted ratings between 4 files.

	Accuracy(Positive)	Accuracy(Neutral)	Accuracy(Negative)	Average class Accuracy	Overall accuracy
LIKES	14,02%	99,97%	-	56,99%	**94,86%**
FAVOURITES	11,03%	98,89%	-	54.96%	85,24%

COMBINATION	12,43%	98,22%	-	55,32%	86,23%
COMMENTS	**18,84%**	**100%**	58,11%	**58,98%**	87,79%

Table 4-13 The results of evaluation of all 3 experiments

The overall accuracy in all experiment was promising, where accuracy was over 85% in all three experiments. It should be mentioned, that the high overall accuracy was reached because of imbalanced ratings distribution, where data consist of mainly with "neutral" class of ratings.

In this case, average class accuracy reflects more reliable evaluation metrics in the experiment. The average class accuracy of all experiment was around 54-59% which is not high. The score of average class accuracy is influenced by positive ratings accuracy, which was low in all the experiments. The possible explanation of low results in positive ratings accuracy could be:

1) Data sparsity. The real data used in the experiment was very sparse (around and less than 1%) which is much less than in existing research projects.
2) Binary data, explicit forms of data such as likes and favourites was binary. The ratings scale of the data was 0-1, where 0 rating were given for watched videos, which represented "neutral" attitude for videos.
3) Imbalance in the data. Distribution of the ratings was much imbalanced. Only 4% of ratings in likes were "1" and 14% of ratings in favourites were "1". Even with the combination of them, and adding sentiment scores in the file decreased imbalance, it is still high.
4) The accuracy of positive class ratings is low in likes, favourites and combined file, but slightly increased in the third experiment. Accuracy of predicted positive ratings in the third experiment, where data was constructed from explicit types of data (likes and favourites) and implicit type of data (derived sentiment scores from user comments) was higher than in previous experiments, at 18,84%. The accuracy of positive ratings in the third experiment was higher for 4, 82% than in likes, 7, 81% than in favourites and 6, 41% than in combination files even if the data was much sparser. The idea of introducing implicit type of

data shows great promise promising taking into consideration that accuracy is increased despite the fact that the data became much sparser.

5. CONCLUSION

5.1 Introduction

The final chapter of the project summarises the conclusions, main findings, contributions to the body of knowledge and highlights the key differences with past literature. The experiment and evaluation results are discussed. Finally, the project challenges and recommendations for future work are highlighted.

5.2 Project review

Popularisation of the Internet and the increasing volume of data used by companies pushed e-commerce websites to find new ways of marketing and targeting their customers in order increase their revenue. One of the most popular solutions used in order to increase profit for online companies is the recommender system. Recommender systems of companies such as Amazon, Netflix, eBay and so on provide personal recommendations suggesting items which users could find interesting for themselves.

The aim of the project was to build a recommender system for an online video sharing website which would recommend video clips to users of the website could find interesting, and evaluate the effectiveness of the system on sample data provided by the online video sharing website Kiwi.kz. The data provided by the company included information about videos which user liked, added as favourite and left comments on it. The website is based in Kazakhstan, therefore users left comments on three different languages (Kazakh, Russian and English) and two different transcripts (Latin and Cyrillic) due to the languages used in the country.

Domain of recommender systems, including examples of existing recommender systems, the functions and the data used was analysed in order to identify the most appropriate approach for recommender system. Different recommendation techniques such as collaborative filtering, content-based filtering and so on have been discussed. The advantages and disadvantages of the two main recommender system techniques, collaborative filtering and content-based filtering were compared more broadly. The algorithms used in the collaborative filtering techniques including user based and item based algorithms, and the process building recommendations were defined in detail.

Recommendations are based on the predicted ratings. To predict a rating for an item, similarity metrics based on Euclidean distance, Jaccard coefficient and Pearson correlation have analysed. Definitions of similarity metrics and equations used to predict ratings provided. The main challenges in collaborative filtering were explained and discussed. The domain of text analysis was discussed in the last section of the chapter. Sentiment classification of textual data which can be used to extract implicit ratings from the textual comments was analysed

Based on the knowledge gained from the literature review chapter, the similarity metrics used in the experiment have been chosen. Use of textual information in the data in order to improve the quality of recommendation was considered for the final experiment. Extracting implicit ratings from textual comments and combination with explicit ratings from user likes and favourites suggested.

The experiment design and methodology chapter described the data, experimental methodology and evaluation metrics used in the experiment. The sections addressing data sets included information about original data provided by the company, attributes of the data and transformation processes used to build user-item matrices for the experiment.

User-item matrices where ratings based on user likes, favourites, and combination of them were built from the data. Semantic analysis of comments has undertaken in order to build user-item matrix for users where ratings are sentiment scores of comments. Introducing data from implicit sources (comments) and combination of it with explicit sources of data (likes and favourites) has been done. The user-item matrix was based on explicit and implicit data, where ratings are a combination of the likes, favourites and sentiment scores of comments. The issues with data sparsity and ratings distribution imbalance in user-item matrices analysed and details of user-matrices were provided.

The data in the experiments divided into a test set and training set to evaluate accuracy of predicted ratings. The process of building recommendations has been described. Definitions of accuracy metrics such as overall accuracy and average class accuracy have been provided. In this chapter details of the data used in the experiment were provided, and methodology of the experiment was discussed.

The goal of the experiments was to identify the most suitable collaborative filtering technique based on the real data and examine possibility of increasing performance of recommender system by a combination of user-item matrices based on likes, favourites and sentiment scores of comments. The first experiment was based on two user-item matrices of user likes and favourites, where the best performed collaborative filtering was chosen to be use in the second and third experiment. The second experiment tried to examine performance comparing to the first experiment based on collaborative filtering techniques chosen from first experiment on the user-item matrix built from combination of user likes and favourites. The third experiment involved user-item matrix built from explicit and implicit data sources in order to identify how combination of different sources of data can influence to the performance of the recommender system.

Based on the results received several interesting trends were revealed from the experiment. Overall accuracy in the experiment was poor, which could be caused by the lack of data and data sparsity problem. The combination of available data tried to handle this problem. The combination of available explicit types of data (likes and favourites) had slightly better accuracy based on favourites alone, despite the fact that the data sparsity was worse. The combination of explicit (likes and favourites) and implicit (comments) ratings performed best of all even though the data was very sparse.

Evaluation have shown that the user-based approach performed better than the item-based approach in all cases, which was somewhat surprising comparing to the existing recommender systems in the projects such as Amazon, Netflix, where the item-based approach is performed better than user-based approach. The Euclidean distance as a similarity metric has shown slightly higher results than Jaccard coefficient. These finding possible reflects to the nature of the likes and favourites files, which are binary. Based on the results from the experiment, user-based approach where Euclidean distance is similarity metrics has been used in the next experiments.

The second part of the experiment was focused on the possibility of increasing performance of recommender systems by combining explicit and implicit sources of data. User based approach where Euclidean distance used as similarity metric has been applied on two user-item matrices where ratings were combination of user likes and favourites and combination of user likes, favourites and sentiment scores of comments. Evaluations based on average class accuracy have been compared between results of the all experiment. The results have shown interesting trends:

1) The recommender system predicts a low level of positive ratings in both user likes and favourites, 14, 02% and 11, 03% respectively. The average class accuracy shows results around 56, 99% and 54, 96% for user likes and favourites, comparing to 94, 86% and 85, 24% in overall accuracy. This level of accuracy is dependent on the amount of data used

in the experiment, where ratings are significantly imbalanced, and the number of positive ratings is much less than neutral ratings.

2) By a combination of user – item matrices of user likes and favourites, number of users and videos increased and the data became much sparser in the user – item matrix with combined ratings. However, the average class accuracy and accuracy of positive class ratings slightly increased than in user-item matrix of user favourites. Possible explanation of the increased results could be that the imbalance of ratings is less than in user-item matrix of user likes, and ratings have more diverse scale of ratings 0-3 rather than previous binary ratings 0-1.
3) Introducing implicit sources of data has increased the accuracy of the recommender system in both positive ratings and average class ratings. Data sparsity in the user-item matrix was 0, 25% which is less than previous user-item matrices in user likes, favourites and combination of them, with values 1,5%, 0,4% and 0,35% respectively. A more diverse scale of ratings, and negative ratings class was introduced in the user-item matrix. Accuracy of positive ratings was higher than in user likes, favourites and combination of them respectively, despite. The idea of bringing implicit data into explicit data has shown promising results due to increased accuracy in predicting positive ratings.

5.3 Personal reflection

The aim for this project was to gain a deep knowledge of the domain of recommender systems, implement recommender systems using real data and evaluate the effectiveness of the recommender system. It required a good understanding of the processes involved in making recommendations and the appropriate methods used in the recommender system. Existing recommender systems are dependent on the nature of the data they are built on, therefore this project depended on a good understanding of the real world data provided by the online website. There were a number of personal and technical challenges in this project.

- This type of project required a significant level of programming skills in order to write code for recommender systems and the evaluation part. Frameworks based on Python programming languages are written for existing projects, but they was not suitable for the data used in the experiment. User likes and favourites used as explicit ratings in the experiment, which have scale of rating 0-1. Existing projects are designed for typical scale of ratings 1-5. Writing a code for a recommender system was one of the main challenges in the project.
- The amount of data used in the experiment was significantly large which required high computational power. Evaluation processes took hours on a local machine which influenced the timetable of the project.
- The lack of experience in the experiment processes had an impact on the proper use of time in the project. Using the part of the data without valuable information, including excessive information about videos not used in predicting recommendations and selecting the wrong thresholds for evaluation cost time, but valuable experience was gained for future research.
- The problem of understanding and transformation processes of textual information in different languages with different encodings was another challenge. The piece of code available was not enough to handle the encoding problems and the list of positive and negative words was only for the English language. The process of creating the list of positive and negative words required analysing number of existing comments, which are very diversified and challenged in classification of them.

5.4 Future work and promising directions

This project provides scope for further investigations and future work. In particular previously highlighted directions include the investigation on the possibility of combining available explicit

binary types of the data. The possibility of evaluation the data with less data sparsity and ratings combined from binary ratings offer an interesting opportunity for future work.

One of the most promising directions in the recommender system is the combination of explicit and implicit source of information. Adding implicit sentiment scores to explicit ratings increased accuracy in the experiment. Further investigation including more deep text analysis and a combination of sentiment score with more diverse scale of ratings could lead to a broader range of perspectives and advantages for performance of recommender systems.

BIBLIOGRAPHY

Arazy O., Kumar N., Shapira B. 2009. *Improving Social Recommender Systems.* IT Professionals, Volume 11, pp. 38-44, 2009

Bell R. M., Koren Y. 2007. *Improved Neighborhood-based Collaborative Filtering*. KDD Cup and Workshop, 2007

Burke R. 2000. *Knowledge-based Recommender systems.* Encyclopedia of library and information systems, 2000

Burke R. 2007. *Hybrid web recommender systems.* In: The AdaptiveWeb, pp. 377–408. Springer, Berlin / Heidelberg 2007

Balabanovic M., Shoham Y. 1997. *Fab: Content-Based, Collaborative Recommendation.* Communications of the ACM, Volume 40, pp. 66-72, 1997.

Breese J., Heckerman D., Kadie C. 1998. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering.* Proceedings of the 14th conference on Uncertainty in artificial intelligence, pp. 43-52, 1998.

Cotter P., Smyth B. 2000. *PTV: Intelligent Personalised TV Guides.* IAAI-00 Proceedings, 2000

Claypool M., Gokhale A., Miranda T. 1999. *Combining Content-Based and Collaborative Filters is an Online Newspaper.* In Proceedings of ACM SIGIR Workshop on Recommender Systems, 1999

Caraciolo M., Caspirro R., Melo B. 2011. *A Python Framework for Building Recommendation Engines.* Scipy, Austin TX, 2011

Das D., Martins A. 2007. *A Survey on Automatic Text Summarazation.* Literature Survey for the Language and Statistics II, 2007

Davidson J., Liebald B., Liu J., Nandy P., van Vleet T. 2010. *The YouTube Video Recommendation System.* Proceedings of the 4th ACM conference on Recommender Systems, pp. 293-296, 2010

Deza M. M., Deza E. 2009. *Encyclopedia of Distances.* Springer, 2009

Debnath S., Ganguly N., Mitra P. 2008. *Feature Weighting in Content Based Recommendation System Using Social Network Analysis.* Proceedings of the 17th international conference on World Wide Web, pp. 1041-1042, 2008

Goldberg D., Nichols D., Oki B., Terry D. 1992. *Using collaborative filtering to weave an information Tapestry.* Communications of the ACM – Special Issue on information filtering, Volume 35, Issue 12, pp. 61-70, 1992

Demiriz A. 2004. *Enhancing Product Recommender Systems on Sparse Binary Data.* Data Mining and Knowledge Discovery, Springer, 2004

Ismail M., H., Jusoff K. 2008. *Satellite Data Classification Accuracy Assessment Based from Reference Dataset.* International Journal of Computer and Information Engineering 2:6 2008

Feldman R., Sanger J. 2007. *The Text Mining Handbook.* Cambridge University Press, 2007

Fitzgerald,R.W and Lees,B.G. 1994. *Assessing the classification accuracy of multiscore remote sensing data.* Elsevier, Volume 47, Issue 3, 1994

Hotho A., Nurnberger A., Paas G. 2005. *A brief survey of Text Mining.* LDV Forum, 2005

Herlocker J., Konstan J., Borchers A., Riedl J. 1999. *An Algorithmic Framework for Performing Collaborative Filtering.* Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 230-237, ACM 1999

Howe M. 2007. *Pandora's Music Recommender.* UW Press, 2007

Hiralall M. 2011. *Recommender systems for e-shops.* Business Mathematics and Informatics paper, Vrije Universiteit, 2011

Joshi A., Xu L. 2000. *A Jini based framework for a Component Recommender Systems.* Sixteenth IMACS World Congress, 2000

Iaquinta L., de Gemmis M., Lops P., Semeraro G., Filannino M., Molino P. 2008. *Introducing Serendipity in a Content-based Recommender System.* 8th International Conference on Hybrid Intelligent Systems, pp. 168-173, 2008

Liu B. 2010. *Sentiment Analysis and Subjectivity.* To appear in Handbook of Natural Language Processing, Second Edition, 2010

Pang B., Lee L. 2008. Opinion Mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2008

Kaji N., Kitsuregawa M. 2007. *Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents.* Proceedings of the joint conference on Empirical methods in NLP and Computational Natural language Learning, pp.1075-1083, 2007

Linden G., Smith B., York J. 2003. *Amazon.com recommendations. Item-to-Item Collaborative Filtering.* Internet Computing, IEEE 2003

Luhn H. P. 1958. *The automatic creation of literature abstracts.* IBM Journal of research and development, 1958

Mooney R., Nahm U. Y. 2003. *Text Mining with Information Extraction.* Proceedings of the 4th International MIDP Colloquium, pp.141-160, 2003

Miller B., Konstan J., Riedl J. 2004. *PocketLens: Toward a Personal Recommender System.* ACM Transactions on Information Systems, Volume 22, Issue 3, pp. 437-476, 2004

Mobasher B., Burke R., Sndvig JJ. 2006. *Model-Based Collaborative Filtering as a Defense Against Profile Injection Attacks.* Proceedings of the 21st national conference on Artificial Intelligence, 2006

McNee S., Riedl J., Konstan J. 2006. *Accurate is not always good: How Accuracy Metrics have hurt Recommender Systems.* CHI '06 extended abstracts on Human factors in computing systems, pp. 1097-1101. ACM, 2006

Melville P., Sindhwani V., 2010. *Recommender Systems.* Encyclopedia of Machine Learning, 2010

Mooney R., Roy L. 1999. *Content-Based Book Recommending Using learning for Text Categorisation.* Proceedings of the SIGIR-99 Workshop on Recommender Systems, 2009

Owen S., Anil R., Dunning T., Friedman E. 2012. *Mahout in Action.* Manning Publications Co, 2012

Papagelis M., Plexousakis D., Kutsuras T. 2005. *Alleviating the Sparsity Problem of Collaborative Filtering Using Trust Inferences.* Lecture Notes in Computer Science, Springe, Volume 3477/2005

Resnick P., Iacovou N., Suchak M., Bergstrom P., Riedl J. 1994. *GroupLens: An Open Architecture for Collaborative Filtering of Netnews.* CSCW '94 Proceedings of the 1994 ACM conference on Computer supported cooperative work, pp. 175-186. 1994

Sarwar B., Karypis G., Konstan J., Riedl J. 1998. *Item-Based Collaborative Filtering Recommendation Algorithms.* Proceedings of the 10th international conference on World Wide Web, pp. 285-295, ACM 2001

Su X., Khoshgoftaar T. 2009. *A Survey of Collaborative Filtering Techniques.* Journal Advances in Artificial Intelligence Volume 2009, Article No. 4

Segaran T., 2007. *Programming Collective Intelligence.* O'Reilly Media Inc, 2007.

Resnick P., Varian H., 1997. *Recommender Systems.* Communications of the ACM, Volume 40 Issue 3, pp. 56-58, 1997

Taghipour N., Kardan A., Ghidary SS. 2007. *Usage-Based Web Recommendations: A Reinforcement Learning Approach.* Proceedings of the 2007 ACM Conference on Recommender Systems, pp. 113-120, RecSys 2007

Shardanand U., Maes P. 1995. *Social Information Filtering: Algorithms for Automating "Word of Mouth".* Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 210-217. CHI 1995

Wang J., de Vries A., Reinders M. 2006. *Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion.* Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval , pp. 501-508, 2006

Ricci F., Rokach L., Shapira B., Kantor P. 2011. *Recommender System Handbook.* Springer Science+Business Media, LLC, 2011

Pazzani M. 1999. *A framework for Collaborative, Content-Based and Demographic Filtering.* Artificial Intelligence Review, Volume 13, Numbers 5-6

Schafer B., Konstan J., Riedl J. 1999. *Recommender Systems in E-Commerce.* Proceeding of the 1st ACM conference on Electronic commerce, pp. 158-166, 1999

Spiegel S. 2009 *A Hybrid Approach to Recommender Systems based on Matrix Factorization.* Diploma Thesis in Technische Universitat Berlin, 2009

Varadarajan S., Kasravi K., Feldman R. 2002. *Text-Mining. Application Development Challenges.* Application and innovations in text mining, springer verlag, 2003

Solka JL. 2008. *Text Data mining. Theory and Methods.* Statistical Surveys, Volume 2, 2008

Rashid AM., Karypis G., Riedl J. 2005. *Influence in ratings-Based recommender systems: An algorithm-independent approach.* Proceedings of the SIAM International Conference, 2005

Sarwar B., Konstan J., Borchers A., Herlocker J., Miller B., Riedl J. 1998. *Using Filtering Agents to Improve prediction Quality in the Grouplens Research Collaborative Filtering Systems*. Proceeding in the ACM Conference Computing Support Cooperation, pp. 345-354,1998

APPENDIX A

Code in Python building recommendations

```
from __future__ import division
from math import sqrt
import string
import sys
from itertools import imap

def jaccard_distance(videos,user1,user2):
        sim={}
        for item in videos[user1]:
                if item in videos[user2]:
                        sim[item]=1
        if len(sim)==0: return 0

        intersect = 0
        s1 = 0
        s2 = 0
        dist=0
        for item in videos[user1]:
                if item in videos[user2]:
                        if videos[user1][item]==videos[user2][item]: intersect += 1
                        for item in videos[user1]: s1 +=1
                        for item in videos[user2]: s2 +=1
                        dist = float(intersect/(s1+s2-intersect))
                        return dist

def euclidean_distance(videos,user1,user2):
        sim={}
        for item in videos[user1]:
                if item in videos[user2]:
                        sim[item]=1
        if len(sim)==0: return 0
        square_sum=sum([pow(videos[user1][item]-videos[user2][item],2)
        for item in videos[user1] if item in videos[user2]])
        return 1/(1+square_sum)

def pearson_correlation(prefs,p1,p2):
   si={}
   for item in prefs[p1]:
      if item in prefs[p2]: si[item]=1
   if len(si)==0: return 0
   n=len(si)
   sum1=sum([prefs[p1][it] for it in si])
   sum2=sum([prefs[p2][it] for it in si])
   sum1Sq=sum([pow(prefs[p1][it],2) for it in si])
   sum2Sq=sum([pow(prefs[p2][it],2) for it in si])
   pSum=sum([prefs[p1][it]*prefs[p2][it] for it in si])
   num=pSum-(sum1*sum2/n)
   den=sqrt((sum1Sq-pow(sum1,2)/n)*(sum2Sq-pow(sum2,2)/n))
```

```
    if den==0: return 0
    r=num/den
    return r


def topSimilar(videos,person,n=5,similarity=jaccard_distance):
            counts=[(similarity(videos,person,other),other)
                        for other in videos if other!=person]
            counts.sort( )
            counts.reverse( )
            return counts[0:n]

def RecommendVideos(videos,choose,person,n=5,similarity=jaccard_distance):
  totals={}
  simSums={}
  a=0
  for other in videos:
    if other==person: continue
    sim=similarity(choose,person,other)

    if sim<=0: continue
    for item in videos[other]:

      if item not in videos[person]:
        totals.setdefault(item,0)
        totals[item]+=videos[other][item]*sim
        simSums.setdefault(item,0)
        simSums[item]+=sim

  rankings=[(total/simSums[item],item) for item,total in totals.items()]
  rankings.sort()
  rankings.reverse()
  return rankings[0:n]

def RecommendVideosNum(videos,person,n=5,similarity=euclidean_distance):
  totals={}
  simSums={}
  for other in videos:
    if other==person: continue
    sim=similarity(videos,person,other)

    if sim<=0: continue
    for item in videos[other]:
      if item not in videos[person]:
        totals.setdefault(item,0)
        totals[item]+=videos[other][item]*sim
        simSums.setdefault(item,0)
        simSums[item]+=sim

  rankings=[(total/simSums[item],item) for item,total in totals.items()]
  rankings.sort()
  rankings.reverse()
```

```
  return rankings[0:n]

def transformvideos(videos):
        result={}
        for person in videos:
                for item in videos[person]:
                        result.setdefault(item,{})
                        result[item][person]=videos[person][item]
        return result



def calculatesimilarItems(videos,n=10):
        # Create a dictionary of items showing which other items they
        # are most similar to.
        result={}
        # Invert the preference matrix to be item-centric
        itemvideos=transformvideos(videos)
        c=0
        for item in itemvideos:
                # Status updates for large datasets
                c+=1
                if c%100==0: print "%d / %d" % (c,len(itemvideos))
                # Find the most similar items to this one
                counts=topMatches(itemvideos,item,n=n,similarity=sim_distance)
                result[item]=counts
        return result

def getRecommendedItems(videos,itemMatch,user):
        userRatings=videos[user]
        counts={}
        totalsim={}
        # Loop over items rated by this user
        for (item,rating) in userRatings.items( ):
                # Loop over items similar to this one
                for (similarity,item2) in itemMatch[item]:
                        # Ignore if this user has already rated this item
                        if item2 in userRatings: continue
                        # Weighted sum of rating times similarity
                        counts.setdefault(item2,0)
                        counts[item2]+=similarity*rating
                        # Sum of all the similarities
                        totalsim.setdefault(item2,0)
                        totalsim[item2]+=similarity
                        # Divide each total score by total weighting to get an average
                        rankings=[(score/totalsim[item],item) for item,score in counts.items( )]
                        # Return the rankings from highest to lowest
        rankings.sort( )
        rankings.reverse( )
        return rankings
```

```
def loadKiwiData(path='c:/Users/Tair/Desktop/Recommender systems/Data/kiwi_data'):
	# Get movie titles
	videos={}
	for line in open(path+'/video-copy.csv'):

	(id,account_id,category_id,adult,views,rating,duration,count_comments,count_likes,creat
ed)=line.split(';')
		videos[id]=id
		videos[views]=views
	# Load data
	likes={}
	for line in open(path+'/video_likes'):
		(id,account_id,video_id,created)=line.split(';')
		likes.setdefault(account_id,{})
		likes[account_id][views]=video_id
	return likes

def loadWatchedData(path='c:/Users/Tair/Desktop/Recommender systems/Data'):
	watched={}
	for line in open(path+'/test/view_history'):
		(id,account_id,video_id,created)=line.split(';')
		watched.setdefault(account_id,{})
		watched[account_id][video_id]=0
	return watched

def loadNumericLikesData(path='c:/Users/Tair/Desktop/Recommender systems/Data'):
	test={}
	int_likes={}
	for line in open(path+'/test/matrix'):
		line = line.strip('\n')
		(account_id,video_id,likes)=line.split(';')
		test.setdefault(account_id,{})
		test[account_id][video_id]=likes
		for likes in test[account_id][video_id]:
			mylikes=int(likes)
		test[account_id][video_id]=mylikes
	return test


def loadVectorLikesData(path='c:/Users/Tair/Desktop/Recommender systems/Data'):
	test={}
	for line in open(path+'/test/matrix'):
		line = line.strip('\n')
		(account_id,video_id,likes)=line.split(';')
		likes=likes.replace('\"','')
		test.setdefault(account_id,{})
		test[account_id][video_id]=video_id+"-"+likes
	return test

def loadMoviesLikesData(path='c:/Users/Tair/Desktop/Recommender systems/Data'):
	test={}
	for line in open(path+'/test/matrix'):
```

```
            line = line.strip('\n')
            (account_id,video_id,likes)=line.split(';')
            likes=likes.replace('\"','')
            test.setdefault(video_id,{})
            test[video_id][account_id]=account_id+"-"+likes
    return test

def loadCommentsData(path='c:/Users/Tair/Desktop/Recommender systems/Data'):
    scores={}
    #int_likes={}
    for line in open(path+'/test/comment_score'):
            line = line.strip('\n')
            (user_id,video_id,score)=line.split(';')
            scores.setdefault(user_id,{})
            scores[user_id][video_id]=int(score)
    return scores
```

APPENDIX B

Data preparation and transformation

Collaborative filtering techniques use a database of preferences for items by users. To build recommendations user-item matrix should be built on the data. Original data provided by kiwi.kz consists of 7 files. Figure A-1 shows the original name and size of each files provided by the company.

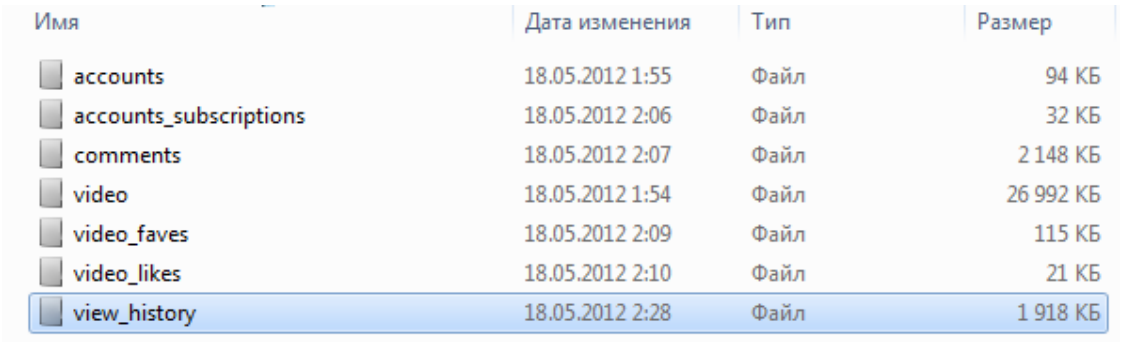


Figure A-1.

File 1. *ACCOUNTS*

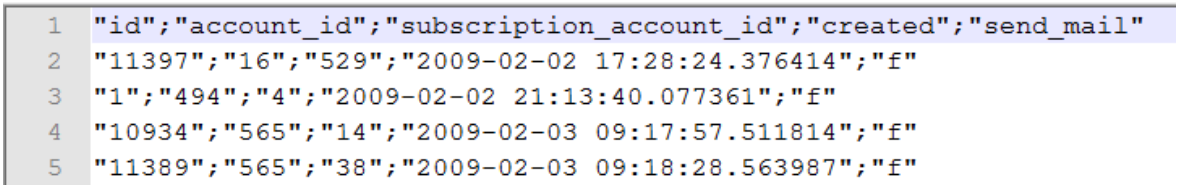
The file *ACCOUNTS* consists information about user, except personal information due to privacy policy and agreement. Each user has unique ID, which is linked to other files as ACCOUNT_ID. The file consists information about user's gender, birthday, account creation time, the number of group, subscriptions and subscribers, city and country of a user, and verify is user banned or not. The file consists of information about 998 users.

```
"id";"created";"banned";"gender";"birthday";"group_id";"count_video";"count_subscribers";"count_subscriptions";"city_id";"country_id";"group_id"
"96";"2008-11-24 13:46:07.876374";"f";"1";"2008-02-02 00:00:00";"7";"17";"112";"0";"10451";"82";"7"
"83";"2008-11-24 12:17:50.977424";"f";"1";"1987-11-21 00:00:00";"1";"689";"689";"9";"10622";"82";"1"
"208";"2008-11-25 11:22:09.882859";"f";"1";"1988-07-29 00:00:00";"7";"82";"24";"1";"10451";"82";"7"
"908";"2009-02-08 07:04:57.257035";"f";"1";"1984-03-20 00:00:00";"0";"3";"0";"1";"10451";"82";"0"
"58";"2008-11-19 12:06:54.029539";"f";"1";"1981-11-30 00:00:00";"1";"5";"3";"2";"10520";"82";"1"
"581";"2009-02-04 13:08:00.952056";"f";"1";"1990-04-17 00:00:00";"0";"11";"4";"10";"10423";"82";"0"
"321";"2008-12-06 22:11:09.422047";"f";"1";"1992-05-05 00:00:00";"0";"10";"3";"0";"10518";"82";"0"
"828";"2009-02-07 23:31:09.170359";"f";"1";"1988-03-17 00:00:00";"0";"11";"0";"0";"10495";"82";"0"
"865";"2009-02-08 05:42:51.107916";"f";"1";"1991-11-20 00:00:00";"0";"0";"0";"0";"";"82";"0"
```

Figure A-2.

File 2. *ACCOUNT_SUBSCRIPTIONS*

The file *ACCOUNT_SUBSCRIPTIONS* consists information about each users subscriptions. The data about each user ID, ID of a subscribed user, the time when it was subscribed, and confirmation of sending email consists in the file. The file has 603 rows.

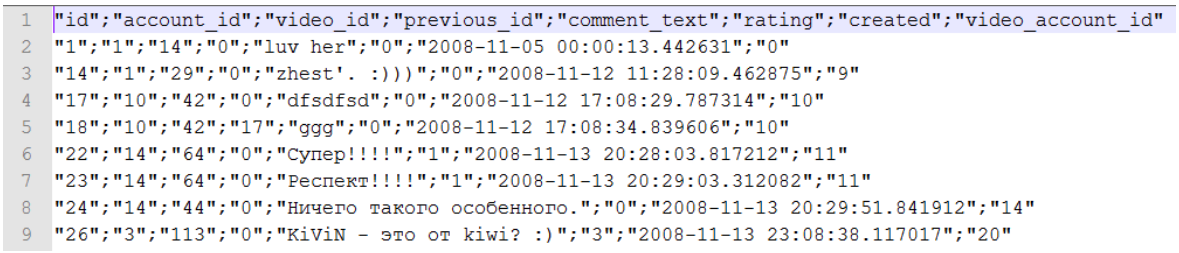


```
"id";"account_id";"subscription_account_id";"created";"send_mail"
"11397";"16";"529";"2009-02-02 17:28:24.376414";"f"
"1";"494";"4";"2009-02-02 21:13:40.077361";"f"
"10934";"565";"14";"2009-02-03 09:17:57.511814";"f"
"11389";"565";"38";"2009-02-03 09:18:28.563987";"f"
```

Figure A-3

File 3. *COMMENTS*

File *COMMENTS* consist of information about textual comment each user left under a particular video clip. The file store information about user ID, ID of commented video, textual comment of a user, the time when comment was left, the rating of comments, ID of user who upload the video, and ID of the user, if it is reply to particular user comment. There are 17999 rows in the file.

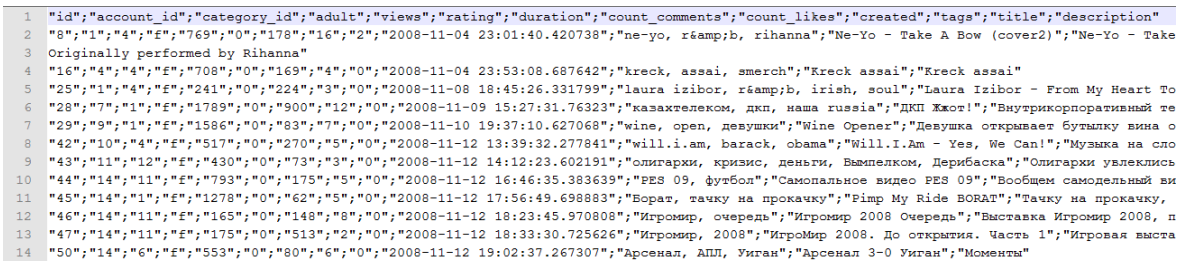


```
"id";"account_id";"video_id";"previous_id";"comment_text";"rating";"created";"video_account_id"
"1";"1";"14";"0";"luv her";"0";"2008-11-05 00:00:13.442631";"0"
"14";"1";"29";"0";"zhest'. :)))";"0";"2008-11-12 11:28:09.462875";"9"
"17";"10";"42";"0";"dfsdfsd";"0";"2008-11-12 17:08:29.787314";"10"
"18";"10";"42";"17";"ggg";"0";"2008-11-12 17:08:34.839606";"10"
"22";"14";"64";"0";"Супер!!!!";"1";"2008-11-13 20:28:03.817212";"11"
"23";"14";"64";"0";"Респект!!!!";"1";"2008-11-13 20:29:03.312082";"11"
"24";"14";"44";"0";"Ничего такого особенного.";"0";"2008-11-13 20:29:51.841912";"14"
"26";"3";"113";"0";"KiViN - это от kiwi? :)";"3";"2008-11-13 23:08:38.117017";"20"
```

Figure A-4.

File 4. *VIDEO*

File *VIDEO* consist of information about video clip. The data about a uploaded user, category belong to, video title, tags, description, created time, duration, the number of likes, views and comments are stored in the file. File store 105412 rows of information about videos.

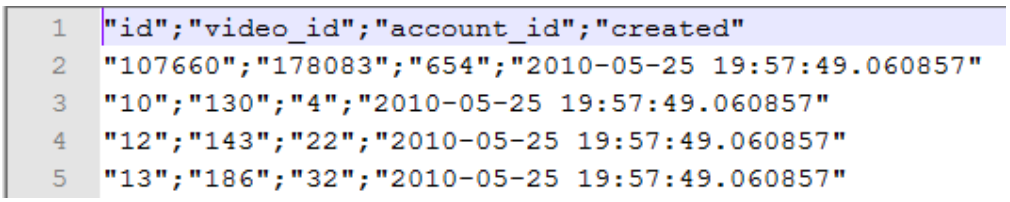


```
"id";"account_id";"category_id";"adult";"views";"rating";"duration";"count_comments";"count_likes";"created";"tags";"title";"description"
"8";"1";"4";"f";"769";"0";"178";"16";"2";"2008-11-04 23:01:40.420738";"ne-yo, r&amp;b, rihanna";"Ne-Yo - Take A Bow (cover2)";"Ne-Yo - Take
Originally performed by Rihanna"
"16";"4";"4";"f";"708";"0";"169";"4";"0";"2008-11-04 23:53:08.687642";"kreck, assai, smerch";"Kreck assai";"Kreck assai"
"25";"1";"4";"f";"241";"0";"224";"3";"0";"2008-11-08 18:45:26.331799";"laura izibor, r&amp;b, irish, soul";"Laura Izibor - From My Heart To
"28";"7";"1";"f";"1789";"0";"900";"12";"0";"2008-11-09 15:27:31.76323";"казахтелеком, дкп, наша russia";"ДКП Жжот!";"Внутрикорпоративный те
"29";"9";"1";"f";"1586";"0";"83";"7";"0";"2008-11-10 19:37:10.627068";"wine, open, девушки";"Wine Opener";"Девушка открывает бутылку вина о
"42";"10";"4";"f";"517";"0";"270";"5";"0";"2008-11-12 13:39:32.277841";"will.i.am, barack, obama";"Will.I.Am - Yes, We Can!";"Музыка на сло
"43";"11";"12";"f";"430";"0";"73";"3";"0";"2008-11-12 14:12:23.602191";"олигархи, кризис, деньги, Вымпелком, Дерибаска";"Олигархи увлеклись
"44";"14";"11";"f";"793";"0";"175";"5";"0";"2008-11-12 16:46:35.383639";"PES 09, футбол";"Самопальное видео PES 09";"Вообщем самодельный ви
"45";"14";"1";"f";"1278";"0";"62";"5";"0";"2008-11-12 17:56:49.698883";"Борат, тачку на прокачку";"Pimp My Ride BORAT";"Тачку на прокачку,
"46";"14";"11";"f";"165";"0";"148";"8";"0";"2008-11-12 18:23:45.970808";"Игромир, очередь";"Игромир 2008 Очередь";"Выставка Игромир 2008, п
"47";"14";"11";"f";"175";"0";"513";"2";"0";"2008-11-12 18:33:30.725626";"Игромир, 2008";"ИгроМир 2008. До открытия. Часть 1";"Игровая выста
"50";"14";"6";"f";"553";"0";"80";"6";"0";"2008-11-12 19:02:37.267307";"Арсенал, АПЛ, Уиган";"Арсенал 3-0 Уиган";"Моменты"
```

Figure A-5

File 5. *VIDEO_FAVES*

Users on the website can add videos as their favourite ones. The *VIDEO_FAVES* file store information about each user ID, video ID, and time when a video was added as favourite. The number of rows in the file is 2312.

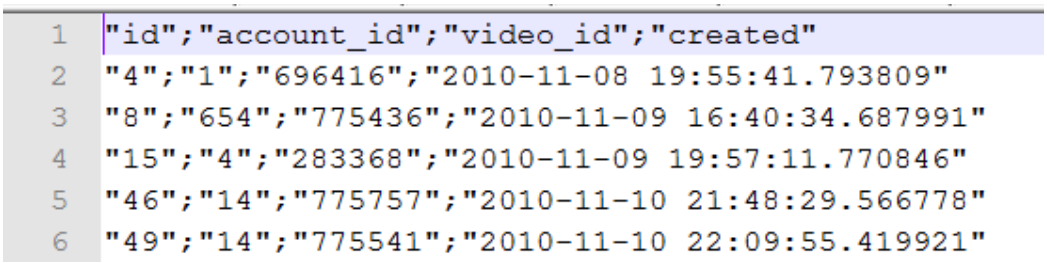


```
1  "id";"video_id";"account_id";"created"
2  "107660";"178083";"654";"2010-05-25 19:57:49.060857"
3  "10";"130";"4";"2010-05-25 19:57:49.060857"
4  "12";"143";"22";"2010-05-25 19:57:49.060857"
5  "13";"186";"32";"2010-05-25 19:57:49.060857"
```

Figure A-6

File 6. *VIDEO_LIKES*

Additionally user can express their attitude to a particular video clip by "liking" it. Each time user press "like" button, information about it recorded to *VIDEO_LIKES* file. Each user ID, video ID and created time are stored in the file. There are 384 rows in the file.

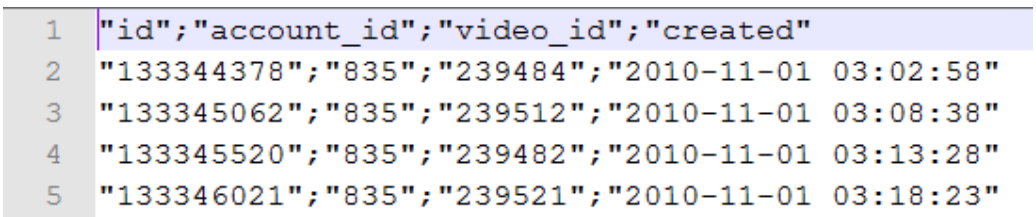


```
1  "id";"account_id";"video_id";"created"
2  "4";"1";"696416";"2010-11-08 19:55:41.793809"
3  "8";"654";"775436";"2010-11-09 16:40:34.687991"
4  "15";"4";"283368";"2010-11-09 19:57:11.770846"
5  "46";"14";"775757";"2010-11-10 21:48:29.566778"
6  "49";"14";"775541";"2010-11-10 22:09:55.419921"
```

Figure A-7

File 7. *VIEW_HISTORY*

Each time user has watched a particular video clip, information about it recorded to *VIEW_HISTORY* file. Information about user ID, ID of watched video and time of each transaction is stored in the file. File has 39080 rows.



```
1  "id";"account_id";"video_id";"created"
2  "133344378";"835";"239484";"2010-11-01 03:02:58"
3  "133345062";"835";"239512";"2010-11-01 03:08:38"
4  "133345520";"835";"239482";"2010-11-01 03:13:28"
5  "133346021";"835";"239521";"2010-11-01 03:18:23"
```

Figure A-8

Data transformation

LIKES_MATRIX file

User likes matrix, where each cell $R_{u,i}$ corresponds to the preference (like or not) of user u for item i was build based on *VIDEO_LIKES* and *VIEW_HISTORY* files. The new file was created by using MySQL 5.5. The full code of generated new file is attached in Appendix A. And the full process of creating it described in Appendix B. Created new file called LIKES_MATRIX represent data about user ID, video ID, and whether user liked or not liked it (Figure 9). The file has 20264 rows, where the number of liked video is 378. It is around 2% from all watched video.

```
account_id;video_id;likes
610;6495;0
610;102290;0
568;644496;0
568;644501;0
485;763209;1
609;776184;1
997;755520;1
```

Figure A-9

FAVOURITES_MATRIX file

FAVOURITES_MATRIX file, consisting of user favourites matrix was created in the same way as *LIKES_MATRIX* file. The new file (Figure A-10) has 22171 rows and the number of users, added videos as their favourite is 2211. It is almost 10% of all watched videos.

```
account_id;video_id;favourites
835;591538;0
986;773294;0
525;248896;1
994;17575;1
654;692479;1
316;326427;1
885;140290;1
```

Figure A-10

It should be mentioned that all three files have duplicated data (number of rows of original files is more than in created new files). In data preparation process, all duplicated data were removed in MySQL, and new created files do not contain duplicated data.

COMMENTS_MATRIX file

One of the promising available data provided by kiwi.kz was *COMMENTS* file. Users left comments on particular video clip and expressed their opinions about the video. Collaborative filtering techniques can be used to comments, if the comments will be transformed to some numerical data type. R software programming tool has been used to transform textual data into numerical data type by semantically analysing data.

Only information about user ID, video ID and textual comment by itself was left in the new file called *COMMENTS_TEXT* (Figure A-11).

```
1  account_id;video_id;comment_text
2  1;14;luv her
3  1;29;zhest'. :)))
4  14;64;Супер!!!!
5  14;64;Респект!!!!
6  14;44;Ничего такого особенного.
7  3;113;KiViN - это от kiwi? :)
8  22;129;мля  я оказывается закачал whatever you like вместо what you know/ =((
```

Figure A-11

After analysing comments text, the program produce the numeric value for each comment from minus to plus, like -1 negatively, 0 neutral or +2 double positively. In Figure N, analysis of comments is shown, and numeric values of column score added to user and video ID to produce new Comments_Matrix file (Figure A-12).

```
> result[1:5,]
  score                      text
1     0              zhest'. :)))
2     0                   dfsdfsd
3     1                 Супер!!!!
4     1               Респект!!!!
5     0 Ничего такого особенного.
```

Figure N

```
> ss[1:5,]
  X1 X14 score                      text
1  1  29     0              zhest'. :)))
2 10  42     0                   dfsdfsd
3 14  64     1                 Супер!!!!
4 14  64     1               Респект!!!!
5 14  44     0 Ничего такого особенного.
```

Figure A-12

```
1  user_id;video_id,score
2  1;14;1
3  1;29;0
4  10;42;0
5  14;64;1
```

Figure A-13.

The new file containing user_id, video_id and numeric values to comments provided in Figure A-13.

APPENDIX C

Code in MySQL.

1. Creating a table consisting user-item matrix, where rating is video likes

```
1) insert into video_liked_history(`account_id`, `video_id`, `liked`)
    SELECT DISTINCT
     l.account_id, l.video_id, l.liked
     #v.account_id, v.video_id, v.liked
 FROM video_likes_edit l, view_history_edit v
WHERE v.account_id=l.account_id and v.video_id=l.video_id
2) (SELECT l.account_id AS account_id, l.video_id AS video_id, l.liked AS liked
        FROM  video_likes_edit AS l
        LEFT JOIN view_history_edit AS v ON (v.account_id=l.account_id AND
v.video_id=l.video_id)
        WHERE (v.account_id AND v.video_id) IS NULL)
3)      insert into video_liked_history(`account_id`, `video_id`, `liked`)
        (SELECT distinct v.account_id AS account_id, v.video_id AS video_id, v.liked AS liked
        FROM  view_history_edit AS v
        LEFT JOIN video_likes_edit AS l ON (v.account_id=l.account_id AND
v.video_id=l.video_id)
    WHERE (l.account_id AND l.video_id) IS NULL)
```

2. Creating a table consisting user-item matrix, where rating is favourited videos

```
    1) insert into video_faved_history(`account_id`, `video_id`, `favourite`)
SELECT DISTINCT   f.account_id, f.video_id, f.favourite
     #v.account_id, v.video_id, v.liked
 FROM video_faves_edit f, view_history_edit v
WHERE v.account_id=f.account_id and v.video_id=f.video_id
    2) insert into video_faved_history(`account_id`, `video_id`, `favourite`)
```

(SELECT f.account_id AS account_id, f.video_id AS video_id, f.favourite AS favourite

FROM video_faves_edit AS f

LEFT JOIN view_history_edit AS v ON (v.account_id=f.account_id AND v.video_id=f.video_id)

WHERE (v.account_id AND v.video_id) IS NULL)

3) insert into video_faved_history(`account_id`, `video_id`, `favourite`)

(SELECT distinct v.account_id AS account_id, v.video_id AS video_id, v.liked AS liked

FROM view_history_edit AS v

LEFT JOIN video_faves_edit AS f ON (v.account_id=f.account_id AND v.video_id=f.video_id)

WHERE (f.account_id AND f.video_id) IS NULL)

3. Cleaning up comments file

1) Deleting rows where user id = 0

delete from comments_id where user_id="0"

Affected rows: 435

2) Deleting all user comments, which are replies to comments of other user

DELETE FROM comments_id where prev_id>'0'

Affected rows: 4635

3) Deleting all duplicated data

insert into comments_previd(`user_id`, `video_id`, `prev_id`,`comment`)

select distinct * from comments_id

Affected rows: 6796

4. Combining Likes_matrix and Favourite_matrix files, with coefficient=2 for favourites and coefficient=1 for likes.

select l.account_id, l.video_id, l.liked, f.account_id, f.video_id, f.favourite from video_liked_history l, video_faved_history f where f.account_id=l.account_id and f.video_id=l.video_id and f.favourite='1' and l.liked='1'

insert into fav_and_like (account_id, video_id, rating) select l.account_id, l.video_id, l.liked from video_liked_history l, video_faved_history f where f.account_id=l.account_id and f.video_id=l.video_id and f.favourite='1' and l.liked='1'

update fav_and_like set rating='3'

delete from fav_and_like_2

update fav_and_like_2 set rating='2'

select f.account_id,f.video_id,f.favourite from fav_and_like fl, video_faved_history f where (fl.account_id!=f.account_id) and (fl.video_id!=f.video_id)

insert into fav_and_like (account_id, video_id, rating) select f.account_id, f.video_id, f.favourite*2 from video_faved_history f left join fav_and_like fl on (fl.account_id=f.account_id and fl.video_id=f.video_id) where (fl.account_id is NULL and fl.video_id is NULL)

5. Combining Likes, favourites and scores of comments file

update fav_and_like_comment flc, comment_score_change ch set flc.rating=flc.rating+ch.score where (flc.account_id=ch.account_id and flc.video_id=ch.video_id)

insert into fav_and_like_comment (account_id, video_id,rating) select flc.account_id, flc.video_id, flc.rating+ch.score from fav_and_like_comment flc,comment_score_change ch where (flc.account_id=ch.account_id and flc.video_id=ch.video_id)

Code in R for analyzing comments

```
setwd("C:/Users/Tair/Desktop/Text analysis")

positive = scan('positive-words-list.txt',what='character',comment.char=';')

negative = scan('negative-words-list.txt',what='character',comment.char=';')

score.sentiment = function(sentences, pos.words, neg.words, .progress='none')

{

  require(plyr)
```

```
  require(stringr)
  scores = laply(sentences, function(sentence, pos.words, neg.words) {
    sentence = gsub('[[:punct:]]', '', sentence)
    sentence = gsub('[[:cntrl:]]', '', sentence)
    sentence = gsub('\\d+', '', sentence)
    sentence = tolower(sentence)
    word.list = str_split(sentence, '\\s+')
    words = unlist(word.list)
    pos.matches = match(words, pos.words)
    neg.matches = match(words, neg.words)
    pos.matches = !is.na(pos.matches)
    neg.matches = !is.na(neg.matches)
    score = sum(pos.matches) - sum(neg.matches)
    return(score)
  }, pos.words, neg.words, .progress=.progress )
  scores.df = data.frame(score=scores, text=sentences)
  return(scores.df)
}
comments<-read.csv("test.csv", sep=";", header=T)
result=score.sentiment(comments[,3],positive,negative)
hist(comments[,3], breaks = bins,main = paste("Histogram of faves,likes and comments"),
xlab='rating',ylab='number',xlim=range(bins),labels=TRUE)
bins<-seq(-2,5,by=1)
```

List of positive words

Luv, love, Супер, супер, респект, Респект, хехехе, Хехехе, hehehe, Hehehe, whahaha, Whahaha, ухахаха,Ухахаха, ахахаха, Ахахаха, хахаха, Хахаха, hahaha, lol, грамотно, нравится, тема, Тема, Охринеть, Ohrinet, Ohrinet’, Ахринеть, Прикольно, прикольно, Prikol’no, Prikolno, prikolno, Prikol, prikol, Прикол, otlichnyi, отличный, нереально, нереальный, 'без б', nereal'no, nerealno, 'bazaru net', 'базара нет', красава, krasava, horoshi, horowi, хороши, понравилось, ponravilos, realno, real'no, реально, реальный, realnyi, классно, класно, классная, класная, классный,класный, klasnaya, klasnyi, klasno, нормально, норма, ништяк, Ништяк, угар, ugar, ugarno, угарно, молодцы, молодца,

молодец, molodec, molotok, молоток, смешно, smeshno, потрясающее, потрясно, красавчики, красавчик, krasavchiki, krasavchik, neploho, неплохо, +100500, ржачь, ржака, rjach, шикарно, shikarno, респектище, шик, любимый, любимая, люблю, lyublyu, lublu, lubimaya, lubimyi, жгут, ниче, niche, цепляет, прет, хороший, хорошо, хороша, хорошенько, horosho, horosha, horoshyi, horoshi, ulibnulo, улыбнуло, вахахаха, вахаха, маладцы, прикооол, круто, kruto, захватывающее, ух, улыбаюсь, веселая, весело, жееесть, здорово, Здорово, zdorovo, gramotno, klassno, nice, Лучшие, лучшие, рулит, rulit, зачотно, зачет, zachet, Афигенскии, пасиб, Зашибенно ,креативно, мощно, Мощные, жжжесть ,ахнуть ,нравица ,нравиццо ,ниче ,) ,A.W.E.S.O.M.E., awesome, пять, +5, жсть, классика, зачетно, коры, шикарные, =), :), ;), :D, Афигеть, Капец, rulit, рулит, 'убил ваще', Раритет, Позитивно, охренеть ,Вахахахааха, шоке, шок, shok, эхехехехе, кул, cool, тащусь, masterclass, мастеркласс, мастер, впечатлил, впечатление, Уахахаахах, пять, приятный, завораживает, завораживающе, Кайф, кай, тащусь, оригинально, отличное, зачот, жжет, Жжет, маладетс, Плюсую, аахахах, позитив, ржал, хех, Хех, интересно, Вахахаха, Ахахахах, awesome, Щикарно, 'Нефига себе', отжиг, отлично, Ура, Шпасибо, 'Спасибо за', Шикарный, Интересный, понравился, великолепно, Уххаахха, Стеб, жжошь, ахаха, креативненького, 'самый лучший', 'От души', хаха, 'просто прет', Ржач, ахах, йа плакаль, айс, вахах, Круть, Красавцы, xD, good, Офигенно, thanks, кульноооо, красиво, охуенно, Акуеть, прикольная, +10, 'O_o', 'love it', ахуенна, задорно, 'очень нравится', 8-D, хахахаха, Уиииии, вау, красаво, АХЕРЕННАЯ, уахахахахахахаха, Щикарна, избранное, красота, Бугага, вхаха, отличная, Отличный, Хаахах, +1, Ваахаххаха, Ураааа, Офигенный, ржакаааа, ЛЫЩ, teeeema, жесть, жжоте, СУУУПЕР, XD, xD, оригинально, 'эта пять', прикольная, Молодцы, Nice, офигенный, Понравился, klassnaya, найс, 5, ржак, Охуенна, Базара нет, Гут, кору, нефиговый, посмеялся, опупенное, ахуенное, жгет, ПАЦТАЛОМНАХ, хехехех, охерееть, Улыбнуло, 'не хуйня', Офигеть, ахахахаха, обожаю, уахахахахаха, классссс, уаахахахаха, ахахах, Обожаю, Прикольная ,Бест ,+100000000 ,Cool, Оооо, бугагашки, прикольный, realnyui, суперская, 'ох какой', 'но слабовато', убило, Восхитительно.

List of negative words

Мля, мммда..., мда..., фигня, лажа, нет, скучная, ниочем, ХУЕЮ, нах, пистес, настоибали, неновая, Бляяяяя, Бля, 'немогут', :(, (, блядь, фуу, отстой, 'Прибилбы', ссаное, идиканахуйблять, жестко, херня, 'неувидел', плохой, жалею, жуть, поганый, странные, 'невтеме', Дибилы, тьфу, Бред, спижжена, гавно, 'ничеговыдающегося', 'неполучается', 'неизменяется', ебанутый, ублюдки, Ебало, 'Неверю', странно, хрень, ужасный, наёбка, тупой', неправ', 'неочень', позор, ждёсткий, НЕЛЫЩ, Капец, убогое, мешки, shit, ЕБНУТЫЕ, Страшная, 'NEPONEL', ЖУТКО, нихуя, хуйня, постанова, пиздешь, дура, несамый, бред, уроды, долбан, лошары, фуууууу, ДОЛБОЁБЫ, слабоВАТО, Тупая, дура,'таксебе', вате, вата.

APPENDIX D

Examples of building recommendations on Python

The first thing in experiment was to upload user-item matrix into python, by transforming the file *VIDEO_LIKES* into matrix, represented as an array. The full code of Python is attached in Appendix A. First of all array of user preferences, where their attitude to particular video represented as vector is uploaded.

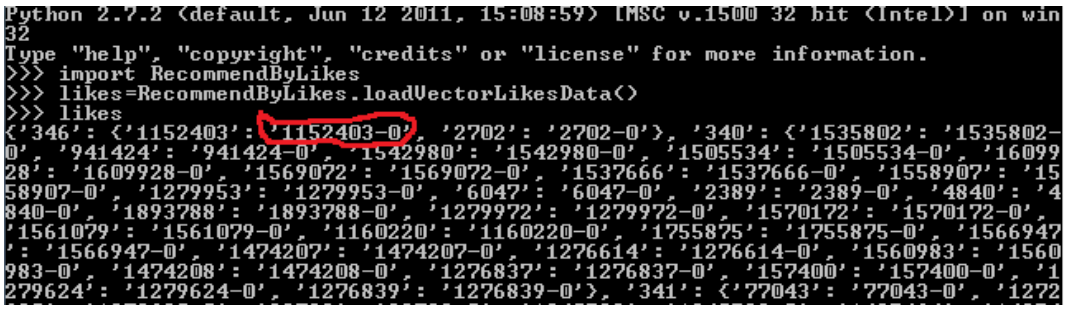


Figure D-1

Array likes represents the matrix user-item, where each implication to a particular video by a user is represented as a vector "video_id-likes". This was done to calculate jaccard_distance similarities based on vector.

To calculate Euclidean distance, numeric values of user likes are needed.

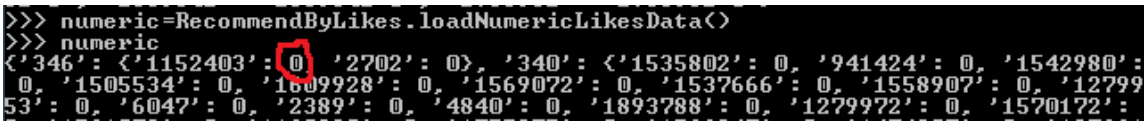


Figure D-2

Jaccard_similarity function calculates similarity between two users. In Figure D-3, the function calculates similarity based on jaccard coefficient between users 1 and 4, 1 and 14, and between 556 and 14. Results are shown that user 1 and user 4 are more similar than others based on jaccard similarity.

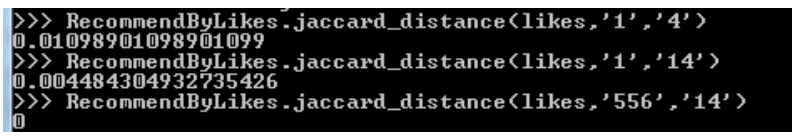


Figure D-3

Euclidean_distance function is used to calculate similarity between two users based on Euclidean distance. In Figure D-4 similarity between users 1, 4, 14 and 556 calculated based on Euclidean distance.

```
>>> RecommendByLikes.euclidean_distance(numeric,'1','4')
1.0
>>> RecommendByLikes.euclidean_distance(numeric,'1','14')
0.3333333333333333
>>> RecommendByLikes.euclidean_distance(numeric,'556','14')
0
>>>
```

Figure D-4

Based on similarity measures, program can identify the most similar users to particular user. topSimilar function calculates and identifies the user ID and shown the degree of similarity of the user. By default the similarity function is jaccard_distance, and function shows the top 5 similar users. In Figure D-6, topSimilar function shows the most similar users to users with ID 1, 4, 14, and 556.

```
>>> RecommendByLikes.topSimilar(likes,'1')
[(0.021739130434782608, '290'), (0.02127659574468085, '85'), (0.0212765957446808
5, '735'), (0.02127659574468085, '45'), (0.02127659574468085, '426')]
>>> RecommendByLikes.topSimilar(likes,'14')
[(0.005617977528089875, '290'), (0.00558659217877095, '85'), (0.005586592178770
95, '81'), (0.00558659217877095, '735'), (0.00558659217877095, '45')]
>>> RecommendByLikes.topSimilar(likes,'556')
[(0.004132231404958678, '779'), (0.004132231404958678, '605'), (0.00413223140495
8678, '336'), (0.004098360655737705, '331'), (0.004081632653061225, '58')]
>>> RecommendByLikes.topSimilar(likes,'4')
[(0.021739130434782608, '652'), (0.021739130434782608, '290'), (0.02127659574468
085, '85'), (0.02127659574468085, '81'), (0.02127659574468085, '735')]
>>> _
```

Figure D-6

To calculate similarities based on Euclidean distance, the name of the functions should be added into topMatches function. Figure D-7 shows top 5 similar users based on Euclidean distance and Pearson correlation for users 1, 4, 14 and 556.

```
>> RecommendByLikes.topSimilar(numeric,'1',similarity=RecommendByLikes.euclidea
_distance)
(1.0, '997'), (1.0, '995'), (1.0, '994'), (1.0, '986'), (1.0, '974')]
>> RecommendByLikes.topSimilar(numeric,'4',similarity=RecommendByLikes.euclidea
_distance)
(1.0, '995'), (1.0, '994'), (1.0, '986'), (1.0, '983'), (1.0, '974')]
>> RecommendByLikes.topSimilar(numeric,'14',similarity=RecommendByLikes.euclide
n_distance)
(1.0, '997'), (1.0, '995'), (1.0, '986'), (1.0, '983'), (1.0, '974')]
>> RecommendByLikes.topSimilar(numeric,'556',similarity=RecommendByLikes.euclid
an_distance)
(1.0, '986'), (1.0, '952'), (1.0, '862'), (1.0, '824'), (1.0, '802')]
```

Figure D-7

The next step in program is to suggest videos to watch with a certain degree of confidence, which is degree of similarity for particular video. RecommendVideos function suggests a list of videos to watch based on one of the similarity metrics. In Figure D-8, the list of 30 videos based on jaccard distance similarity metric suggested to users 1 and 556. This list gives not only a ranked list of movies, it also get a guess of a degree of liking video for each movie. For example, video with ID 985723 probably will be definitely liked by user 1, whereas, video with ID 935398 probably will be liked only by 0.73 of 1 by user 556.

```
>>> RecommendByLikes.RecommendVideos(numeric,likes,'1',n=30)
[(1.0, '985723'), (1.0, '942870'), (1.0, '935290'), (1.0, '935212'), (1.0, '9294
42'), (1.0, '923128'), (1.0, '922503'), (1.0, '912112'), (1.0, '867372'), (1.0,
'857589'), (1.0, '830651'), (1.0, '823847'), (1.0, '823259'), (1.0, '812185'), (
1.0, '803460'), (1.0, '792496'), (1.0, '783182'), (1.0, '755520'), (1.0, '6897')
, (1.0, '5725'), (1.0, '5714'), (1.0, '5648'), (1.0, '456827'), (1.0, '45253'),
(1.0, '407006'), (1.0, '406672'), (1.0, '381027'), (1.0, '3564'), (1.0, '3238'),
 (1.0, '311114')]
>>> RecommendByLikes.RecommendVideos(numeric,likes,'556',n=30)
[(1.0, '985723'), (1.0, '935290'), (1.0, '935212'), (1.0, '854087'), (1.0, '8306
51'), (1.0, '811365'), (1.0, '792496'), (1.0, '783630'), (1.0, '783182'), (1.0,
'781897'), (1.0, '776291'), (1.0, '775757'), (1.0, '696416'), (1.0, '6897'), (1.
0, '459943'), (1.0, '423851'), (1.0, '3238'), (1.0, '306270'), (1.0, '261352'),
(1.0, '25335'), (1.0, '22789'), (1.0, '195421'), (1.0, '1834677'), (1.0, '182677
1'), (1.0, '1550209'), (1.0, '1529438'), (1.0, '1260647'), (1.0, '1169824'), (1.
0, '1155813'), (0.733385004587142, '935398')]
>>>
```

Figure D-8

RecommendVideosNum function gives a list of ranked videos with a guess of liking based on Euclidean distance or Pearson correlation. In Figure D-9, the list of 30 videos for users 1 and 556 has shown based on Euclidean distance metric.

```
>>> RecommendByLikes.RecommendVideosNum(numeric,'556',n=30)
[(1.0, '998759'), (1.0, '985723'), (1.0, '935290'), (1.0, '935212'), (1.0, '9294
42'), (1.0, '922503'), (1.0, '867372'), (1.0, '830651'), (1.0, '820736'), (1.0,
'812185'), (1.0, '798996'), (1.0, '792496'), (1.0, '783182'), (1.0, '781897'), (
1.0, '776291'), (1.0, '755520'), (1.0, '69534'), (1.0, '6897'), (1.0, '434882'),
 (1.0, '423851'), (1.0, '363697'), (1.0, '3564'), (1.0, '3238'), (1.0, '306270')
, (1.0, '291657'), (1.0, '261352'), (1.0, '25335'), (1.0, '23221'), (1.0, '22789
'), (1.0, '19675')]
>>> RecommendByLikes.RecommendVideosNum(numeric,'1',n=30)
[(1.0, '998592'), (1.0, '985723'), (1.0, '942870'), (1.0, '935290'), (1.0, '9352
12'), (1.0, '929442'), (1.0, '923128'), (1.0, '922503'), (1.0, '912112'), (1.0,
'867372'), (1.0, '857589'), (1.0, '830651'), (1.0, '823847'), (1.0, '823259'), (
1.0, '812185'), (1.0, '803460'), (1.0, '792496'), (1.0, '783182'), (1.0, '755520
'), (1.0, '69534'), (1.0, '6897'), (1.0, '5725'), (1.0, '5714'), (1.0, '5648'),
(1.0, '456827'), (1.0, '45253'), (1.0, '434882'), (1.0, '407006'), (1.0, '406672
'), (1.0, '381027')]
>>>
```

Figure D-9