RECO, A HYBRID VIDEO RECOMMENDATION SYSTEM USING USER BEHAVIOURS AND FEATURES OF VIDEOS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

ŞÜKRÜ BEZEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2014

Approval of the thesis:

RECO, A HYBRID VIDEO RECOMMENDATION SYSTEM USING USER BEHAVIOURS AND FEATURES OF VIDEOS

submitted by **ŞÜKRÜ BEZEN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _____

Prof. Dr. İsmail Hakkı Toroslu
Supervisor, **Computer Engineering Department, METU** _____

Assoc. Prof. Dr. Pınar Karagöz
Co-supervisor, **Computer Engineering Department, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _____

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU _____

Dr. Onur Tolga Şehitoğlu
Computer Engineering Department, METU _____

Dr. Güven Fidan
CEO, ArGeDor Information Technologies _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ŞÜKRÜ BEZEN

Signature :

ABSTRACT

RECO, A HYBRID VIDEO RECOMMENDATION SYSTEM USING USER BEHAVIOURS AND FEATURES OF VIDEOS

BEZEN, Şükrü
M.S., Department of Computer Engineering
Supervisor : Prof. Dr. İsmail Hakkı Toroslu
Co-Supervisor : Assoc. Prof. Dr. Pınar Karagöz

September 2014, 69 pages

If you are talking about video content on the internet, you have to think in big scales. That is why we need a system that handles the constant streaming of user behaviour on video content without delaying its recommendation system at the background. We approach to this problem by combining collaborative and content based recommendation algorithms on a framework which is completely scalable.

In the thesis, we explain how we combine behaviours of users and features of videos. In what ways a change on user behaviours effects the recommendation and how is meta data mapped to user behaviours for creating the recommendation.

Keywords: Recommendation Systems, Weighted Hybrid Recommendation Sys-

tems, Video Recommendation Systems, Data Mining

ÖZ

RECO, KULLANICI DAVRANIŞLARINI VE VİDEOLARIN
ÖZELLİKLERİNİ KULLANAN BİR HİBRİT VİDEO TAVSİYE SİSTEMİ

BEZEN, Şükrü
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. İsmail Hakkı Toroslu
Ortak Tez Yöneticisi : Doç. Dr. Pınar Karagöz

Eylül 2014 , 69 sayfa

İnternet üzerindeki video içeriğinden bahsettiğimizde büyük ölçeklerde düşünmemiz gerekmektedir. Bu sebeple video tavsiye sistemlerinin gerçek zamanlı olması ve sürekli artan video içeriği karşısında performans kaybı yaşamadan cevap verebilmesi gerekmektedir. Bu sorun karşısında hem video içeriklerini kullanan hem de benzer kullanıcılar arasındaki ilişkiler üzerinden ilerleyen ve tamamen ölçeklenebilen, video verisinin veya kullanıcı sayısının büyüklüğünden etkilenerek performans kaybı yaşamayan bir video tavsiye sistemi öne sürerek çözüm buluyoruz.

Tezimizde kullanıcı ilişkilerini ve videoların içerik özelliklerini nasıl birleştirerek kullandığımızdan bahsediyoruz. Kullanıcıların davranışlarındaki bir değişikliğin tavsiye sistemimizi nasıl etkilediğini ve kullanıcı verisi ile video içerik verilerinin nasıl bir arada kullanıldığından detaylı bir şekilde bahsediyoruz.

Anahtar Kelimeler: Tavsiye Sistemleri, Ağırlıklı Hibrit Tavsiye Sistemleri, Video Tavsiye Sistemleri, Veri Madenciliği

To my parents

Halil BEZEN, Tülay BEZEN

ACKNOWLEDGMENTS

I would like to thank to my supervisor Professor İsmail Hakkı Toroslu for his support, guidance, friendship. It was a great honor to work with him for the last 4 years and his cooperation influenced my view of academical world highly.

I would also like to thank my ex-workfellows in Nokta Internet Technologies, Erdem Ağaoğlu and Soner Büyükatalay for their technical mentorship and friendship that they provided to me for one year. I wouldn't be able to finish my thesis without them.

Finally I would like to thank to my friends for their friendship and trust, Aslıhan Bener, Uğur Doğan Gül, Ozan Şener, Sencer Burak Okumuş, Umut Bali, Uğur Dönmez.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER I

INTRODUCTION

World Wide Web was first introduced by Tim Berners-Lee [12] When we compare the state of web now and when it was first introduced, we see huge differences in terms of scalability, usability, analytics, content and etc.

Web 1.0 brought us hyperlinks. Hyperlinks provided linking pages to each other. Hyperlinks enabled us to reference to other pages from our own pages. Later PageRank algorithm [41] was introduced. It was a way of ranking pages according to incoming and outgoing links of the page had. This enabled us to search for specific results of pages based on a query. But still it was not dynamic.

Web 2.0 brought us dynamicly loading content on pages. Users needed to load the whole page content when even a small part of it was changed, but not anymore. Web 2.0 brought the ajax technology which enabled loading a small part of a page without loading it whole. This reduced the total time spent when a user wanted to reach to a content. In addition, now users could publish their own content on the internet. Users were not only reading but they were also contributing to the content on the web. The web was growing so fast with user generated content, text, videos, photos, etc.

Web 3.0 is about making the web semantic. Machines process the content on the web and give users personalized results. Every page contains specific meta tags that describe what the page does and what kind of content it contains. But this is just the data, it is nothing if the machine can not understand it. That is why we have our algorithms, architectures, state of the art methods to create

the desired personalized outcome to the user on a specific topic.

Personalization is important. A webpage should be able to collect the data users leave behind intentionally or unintentionally and process it to further provide personalized content to the user. The key part here is how to process it with a high success rate. In 2006, Netflix announced a competition about recommendation systems, Netflix Prize [11], which called for competitors around the world to further improve Netflix's recommendation system. It was a huge success because the world started to understand the importance and value of recommendation systems. This competition was the trigger of it.

Lots of sites use recommendation systems in their systems heavily. For example, LinkedIn uses it to recommend jobs to user which are related to user's skillset or people to connect to such that user may know them based on his/her skillset, school, job history and etc. Youtube uses it to recommend videos to the user based on the watch history of the user, artist of the video, genre of the video, geolocation of the video and etc. Twitter uses it to recommend accounts to the user to follow based on their tweet topics similarity, their distance to the user on the following/followed social network structure and etc. Spotify uses it to recommend songs to user based on listening history of the user, genre of the currently playing song, nationality of the artist and etc. And many more ... Those examples have all in common that they use the data generated by users, either by creating a content or surfing on the existing content, to produce personalized recommendations to the user. Popular websites have millions of active users nowadays and the data, content generated by those users are enormous. Those enormous amounts of content and data can be type of texts, images, videos, audios, etc. In this thesis we are going to focus on video content. Videos are much more effective on users when compared with other types of contents because it is visual and it provides a sequence of images that tells a story. In this thesis we will be introducing a video recommendation system, RECO. RECO is a state of the art, hybrid, scalable video recommendation system and it gives recommendations based on user behaviours and details of the video content.

I.1 Thesis organization

In Chapter 2, we will be talking about the related work. Various recommendation system algorithms will be explained here with advantages and disadvantages of them and also the algorithms that are being used in daily life will also be provided.

In Chapter 3, we will be talking about the technologies used for analyzing the performance of and building RECO. Some of

In Chapter 4, we will be talking about RECO itself. We will discuss why we built RECO and what algorithms we used in it.

In Chapter 5, we will be talking about the evaluation of RECO. Possible future work will also be discussed there.

CHAPTER II

RELATED WORK

In this chapter, we are going to explain various recommendation system algorithms in the literature. We are going to talk about advantages and disadvantages of each algorithm and later we will discuss which disadvantages of those algorithms are solved with RECO and what RECO adds into the literature.

II.1 Content Based

Websites consists of items. For example for an e-commerce site items are the products sold, for a video streaming site items are the videos, for a music streaming site items are the songs and etc. Websites keeps detailed information of the items in their database. You can reach to name of the artist, release date of the video, the place this video was recorded, length of the video, name of the video and etc. on a video item for instance.

Websites also store user data. The moment a user visits a webpage, whether the user is a member or not, this webpage logs the user based on his/her loca-

TableII.1: A song database example

Id	Name	Genre	Year	Artist
1	Deep	Doom Metal	1999	Anathema
2	I Want To Break Free	Rock	1984	Queen
3	Seperate Ways	Rock	1983	Journey
4	Stronger	Pop	2000	Britney Spears

tion, time of his/her visit, browser s/he uses, operating system s/he uses, ip of his/her computer and etc. Later if the user wants to register to the site, s/he gives voluntarily some specific personal info such as his/her email, telephone, city/country s/he lives in, interests and etc. This registration process creates a unique entry for the user on the database that user informations are kept. One can say that, being a member of a website is like carrying an id badge on them all the time while surfing on that specific site so that every action they trigger are processed on their behalf. Item descriptions are not always entered manually into the databases. Think of a song database that needs to update the lyrics of songs as a feature for each song but system has millions of songs so entering manually is not an option. Solution here is using one of the lyric services that provide the lyrics of a song when queried by the name of the song and singer. One can use external services also to gather extra information on items or users to provide better content based recommendations. Using external, trusted sources to gather data for content based recommendation is a great way of constructing the database.

Some popular platforms that we use frequently and have profiles on, such as Facebook, Twitter, Google+, Github, store our personal data. By using those sites we can also log into some third party sites by using our membership info. If we grant access to this third party site to read our personal data on those sites, then this third party site achieves gathering lots of information about us from trusted sources.

A great example of this is introduced in [30]. They simply analyze the tweets of Twitter users and later provide us the ability to query for some spesific set of users interested in some specific area. Their approach is based on classifying the tweets of users and also tweets of people those users follow. That's because following someone means sharing at least one interest area with them. This is a hybrid approach actually which we will be talking about in the further sections but for now lets just concentrate on content based part. They use information retrival methods in their work. They take the twitter profile of people as an unstructured text and use TF-IDF to find matches between profiles on a specific interest area provided by the query.

Now let's look at applications such as Shazam or SoundHound which identifies songs when you provide 20-30 seconds of any song to them. As described in [60], Shazam takes fingerprints of audio files, with a kind of hash function, and records them into their database based on their fingerprint, song name and artist and etc. When a user queries with a partial song they search in the database for a similar fingerprint and when a match is found they provide the answer to the user.

Advertising business also uses content based recommendation very frequently. In this specific area items are found after processing unstructured text, the webpahe itself as a whole, and they are processed by text classification, tf-idf and various other unstructured text analysis methods. When users are introduced advertisements which they are not interested in, it just annoys them and makes the brand lose its impact on the user. To stop this and give the user desired advertisement, advertisers use cookies stored in the user's computer and achieve a higher success rate. Content-targeted advertising uses keyword matching [46]. The most successful example of this is Google Adwords. Since Google have a huge amount of webpages on the Internet in their datacenters as indexed, Google Adwords can target users based on their interests and keywords on a page.

Decision trees are also used in content based recommendation systems . In [15], they create an ontology-based decision tree learner which predicts item ratings by semantically generalizing the item features by using the domain knowledge.

Bayesian classifiers are also another commonly used method in content based recommendation systems. In [27], they create a knowledge base which consists of products and their semantic attributes. They use text learning algorithms like naïve bayesian classifier to create this knowledge base from product data. In [44], they use naïve bayesian classifier to find out which World Wide Web pages on some specific topic would be interested to the user. They show naïve bayessian classifier method is at least as successful as other methods in the evaluation part.

Personalized television recommendation systems usually gives recommendations based on watch history of the user. In [32], they use activities, interests,

moods, experiences and demographic information of a user to recommend shows to him/her. They create a neural network and feed it with the data of the users. Later they predict the users' tv program preferences. They show that this method increases recommendation accuracy significantly.

Association rules find relationships between items which does not seem to be related at first sight. It finds probabilities of relations of items. There are two important concepts, support and confidence. Lets assume we have a database of lots of items. Lets assume X and Y are two items in this database and lets say that we have a rule such that X => Y, meaning whenever there is X there is a Y. Now lets define support and confidence by using X and Y. Support is the percentage of X and Y appearing together in the whole database and confidence is the percentage of Y appearing in the rows that X appear. Association rule mining takes a database of items and gives supports and confidences of itemsets that have a relation between. In [2] they provide an algorithm to apply the association rule mining on large databases in a fast way. When you think about it, content based recommendation systems also find relations between items so that it can give recommendations. Association rule mining is a method used in content based recommendation systems. We can find examples of association rules being used to provide recommendations in [37] and [24].

Content based recommendation systems has emerged from information retrieval research [7] [48]. The importance of texts was realized when information retrieval researchers found the knowledge that can be extracted from texts with using numerous text mining methods. Since then content based recommendation systems have been using text data such as web pages, documents and any kind of related unstructed text.

II.2 Collaborative Filtering

Colloborative filtering is the most commonly used approach in recommender systems as mentioned in [4]. Colloborative filtering basically finds users which have similar behaviour patterns in the context of rating a movie or giving a

TableII.2: User – item matrix based on movie ratings

User Id	Hobbit	LOTR-1	LOTR-2	LOTR-3	Saw	Hostel-1	Hostel-2	Rec
1	4	5	2	?	2	1	1	5
2	?	5	3	4	1	2	2	4
3	5	5	4	5	1	1	1	5
4	2	2	?	2	5	5	4	2
5	1	2	2	1	4	5	5	1
6	2	2	2	2	5	5	5	2

feedback or giving an opinion on a topic, and use these similarity relations to recommend them content by using each others' past behaviours. To simply put it, if there two persons X and Y and they have a history of going together or enjoying the same type of movies in the past, then there is a big chance that in the future they will be sharing a common taste pattern on movies as well.

For example think of a movie recommendation system in which users rate movies in a scale of one to five. In this system there will be two types movies in different genres and two kind of user profiles each liking a different genre. Based on these assumptions, we will be trying to figure a future behaviour of users whose data is missing meaning they have not yet seen the movie and a perfect candidate to give a recommendation to.

In table 2.2, you can see six users and their ratings on eight different movies and with some ratings missing. This is just a simulation of a real life scenario actually because sites that recommend movies to users based on ratings such as IMDB [39] (www.imdb.com) and Netflix (www.netflix.com) deal with cases like this one.

When we look into the ratings given by users in table 2, we see that there are two user groups. First user group consists of User 1,2 and 3 who like Hobbit, LOTR-1, LOTR-3 and Rec. Second user group consists of User 4,5 and 6 who like Saw, Hostel-1, Hostel-2. Missing ratings in table 2 can be found by using ratings of other users in the table who have similar taste of movies with the owner of the missing rate. For example User-2's rating on Hobbit is missing. We know that User-2 have similar movie taste with User-1 and User-3. We can

use User-1's and User-3's rating on Hobbit to find User-2's missing rating on Hobbit. Taking the average of two values, User-2's rating on Hobbit becomes 4,5 .

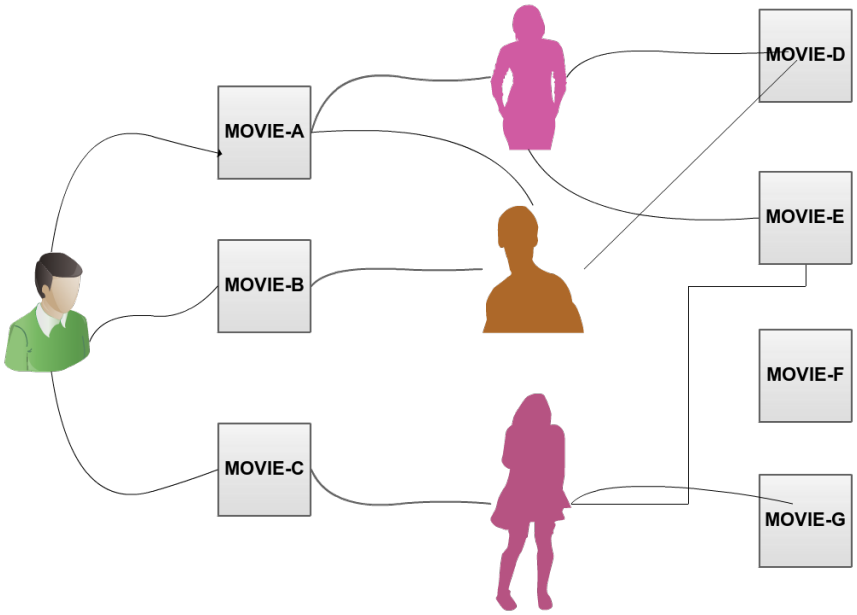
When we look into the Table 2.2, we can find missing entries and see similarities but what if the table was much bigger and similarities were not so easy to find? We need a way of automating these steps, we need an algorithm. Firstly we need to be able to create a matrix consisting of M rows and N columns (in our case M rows were Users and N columns were Movies) and creating a MxN matrix. In this matrix rows will be Users and columns will be Items, this generalization is important because various kinds and quantities of items can be used in colloborative filtering systems. Now we have our data storing schema set, we can insert values into our data store by simply saying go to Mth row and Nth column and store the value there. But we have a problem, sites like our big video website where number of registered users and items the site store are counted with millions would make a big matrix and the most important of all, a big percentage of this matrix would be empty. If you think about it makes sense since not every user in a website gives his/her opinion about every item that this website stores. This problem is introduced and discussed in [1].

Colloborative filtering have two primary areas neighbourhood methods and latent factor models [34]. In addition, according to [16], colloborative filtering can be grouped into two general classes, memory based and model based. Now lets discuss those areas and classes in detail.

II.2.1 Neighbourhood Methods

Neighbourhood methods takes relations into consideration. User watching a movie is a relation between the user and the movie, actor playing in a movie is a relation between the actor and the movie, user liking an actor is a relation between the actor and the user and etc. Lets assume we have a user U and U wants to watch a movie today that s/he did not watch before. To be able to recommend a movie to U, we look into U's ratings on other movies, actors U liked, directors U follows and etc. Each choice of U is a relation which helps

Figure II.1: User based neighbourhood method



us to find the movie that we will be recommending. In a way, what we are trying to accomplish is to predict the missing relations of U with items. This prediction depends on other users' relations as well. If there is a group of users that watched some of the movies U watched and watched some other films U never heard until now, then we can recommend those new movies to U as well based on neighbourhood level.

Neighbourhood methods can be divided into two as well. In [34] it is told that those sub methods are user based neighbourhood method and item based neighbourhood method. Item based neighbourhood methods simply tries to find neighbours of items such that those items would get similar ratings if they would be rated by the same user. If our items are movies then director, genre, actors of a movie and various other features of the movie would be used to find neighbours of this movie. Eventhough this looks like a content based method, actually it is not because let's assume the movie we want to find neighbours of is "Lord of the Rings" and when we say "Orlando Bloom" was an actor in this movie we are not talking about a feature of the movie but a relation of the movie with the actor node labeled with "Orlando Bloom". This means that now we are dealing with item nodes as primary entities and relations between them and also some

properties of those item nodes and thus making a big connected graph structure (Being connected is not mandatory).

On the other hand, user based neighbourhood methods tries to find likeminded users who have common relations on a set of items. For example in the Figure 1 we can see users and their relations with movies. We see that MOVIE-A is watched by three users so when we want to recommend a movie to one of those users we can use other two user's movie relations because those three are like-minded based on MOVIE-A.

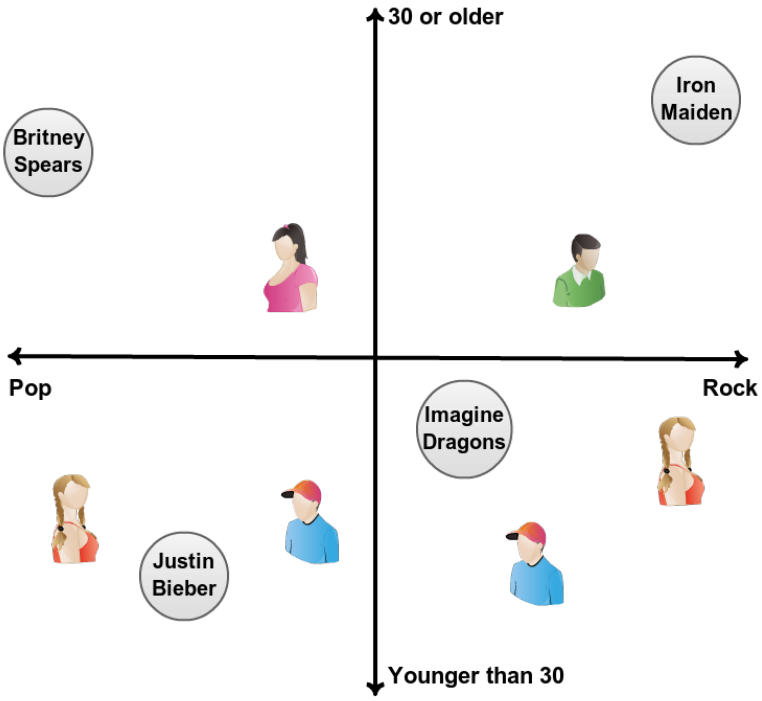
II.2.2 Latent Factor Models

Latent factor is an alternative to the neighbourhood methods. Based on the the data on user-item matrix, we find patterns on the data and characterize it for both users and items. Movie genres like drama, action, horror and etc and song genres like pop, rock, metal and etc. can be found by latent factor models. Those things are not the only ones that can be found but kind of relations we can not even think about or explain can also be found with latent factor models.

In Figure 2.2 we can see four hypothetical axes named Pop, Rock, 30 or older and Younger than 30. Pop and Rock will be patterns for genres of music items and 30 or older and Younger than 30 will be a pattern for users illustrating their age. On the Figure 2 there are both items and users on the matching quarters according to their features. For example Justin Bieber is younger than 30 and he is producing songs on pop genre. But beware of the fact that this is a hypothetical two dimensional factor example because in reality there can be a hundred or more factors.

To be able to find those hundreds of different factors, one needs data to fill the user-item matrix. In general, there are two ways of gathering data, by using explicit feedback or implicit feedback from users [34].

Figure II.2: Latent factor method illustration



II.2.2.1 Explicit Feedback

Explicit feedback is the method of gathering data from user in such a way that user triggers the action, that creates the data, on purpose. Lets look at some real life examples. A user can rate movies on Netflix, rate videos on Youtube, star sound tracks on Spotify, thumbs up webpages on StumbleUpon and etc. Those data are generated by the user willingly. User shows his/her interest level to us. This kind of data gives us a direct approach when we are trying to identfy interests of users on items. In addition, the user-item matrix constructed from explicit feedback is a highly sparse matrix because users can only rate a small group of items which they are interested in.

II.2.2.2 Implicit Feedback

Implicit feedback is the method of gathering data from user in such a way that user is not aware that his/her data is being gathered but the site logs user's

mouse behaviours, ip, time spent on a page, percentage of video watched, page surfing behaviour and etc. All those data can be gathered to identify the user based on his/her behavioural patterns. Implicit feedbacks are necessary when explicit data we gathered is not enough or implicit data contains some valuable information for our potential future factors. The user-item matrix constructed from implicit data is a dense matrix because every user has those data since they are creating it without being aware of it.

II.2.2.3 Matrix Factorization

Matrix factorization is one of the most widely used methods in latent factor analysis. It simply finds latent factors of users and items by using the data gathered by explicit and/or implicit feedback before.

Firstly, users and items each are represented with vectors. For an item, this vector shows the factors this item resembles of and for a user this vector shows the factors of items interested to the user. When a user vector and item vector's dot product is taken interest of a user for an item can be found and by using this value we can create recommendations. The challenge here is to match all user vectors with item vectors and this can be a huge problem since user-item matrix can be really large like 1 million x 4 million in our website's case.

Another serious problem is the sparsity of the user-item matrix. This effects the amount of time it takes to apply matrix factorization. Sparsity problem can be solved by using dimensionality reduction. Since not all features of an item are really important when giving recommendations (number of vowels in the name of the video for instance) to the user and maybe there are some features that are sub-features of a feature (consider we hold the name of the video and the length of the name of the video in two seperate features for instance), then if we can reduce the dimensionality of user-item matrix then we could lower the time for matrix factorization and give better recommendations since redundant features could effect negatively our recommendations.

SVD, Singular Value Decomposition, is one of the most used methods in matrix

factorization to find latent factors. SVD is used in dimensionality reduction too[13]. In SVD three matrices are generated. First one holds the singular vectors corresponding to the columns of the user-item matrix, second one is a dioganal matrix containing corresponding singular values and third one holds the singular vectors corresponding to the rows of the user-item matrix. Singular values resembles the variance in original vector data. This is important in dimensionality reduction because the amount of information that is gonna be lost when features are discarded can be calculated now. For example in [40], effects of different tag similarity techniques are analyzed on three dimensional SVD recommendation systems. Another example of using SVD in geo-activity recommendations can be seen in [50].

II.2.3 Memory Based Collaborative Filtering Algorithms

In memory based recommendation systems, whole user-item matrix is stored. Neighbourhood methods are an example of memory based recommendation system. In neighbourhood methods, user based or item based, we needed to find relations and to be able to find those relations in real time, we need to store the user-item matrix. Firstly, we analyze the user-item matrix in terms of which objects are in relation with the object that we would like to find similars of and then we find neighbours of the object that we would like to find similars of based on some similarity metrics. Similarity metrics are at utmost importance here because it changes the neighbours of our current object directly and that's why it changes the performance of the system directly.

There are various kinds of similarity metrics that we can find on literature such as Pearson's correlation coefficient, Cosine similarity, Spearman's rank correlation coefficient, Mean squared difference, Euclidean distance, Jaccard coefficient, Tanimoto coefficient and etc. One have to choose the similarity metric carefully because it will directly influence the performance of the recommendation.

II.2.4 Model Based Collaborative Filtering Algorithms

In model based recommendation systems user-item matrix is not stored but instead a model is constructed from user-item matrix. By using this model, we can give recommendations to the user based on the previous data and update the system later with the current data for future recommendations. This model learns from the previous data in spare time not in run time. This approach makes it easier in terms of space complexity. Not storing user-item matrix provides us a huge storage benefit but on the other hand realtime recommendations gets delay because this system does not work in real time.

To create a model we can use matrix factorization or probabilistic models. In 2.2.2.3 we explained matrix factorization in detail. Probabilistic models use proven formalities of probabilistic theory. Bayesian belief networks[17], markov decision processes[51] are examples of methods used.

An example to probabilistic models would be Google news personalization engine. "Google News is a free news aggregator provided and operated by Google Inc., selecting most up-to-date information from thousands of publications by an automatic aggregation algorithm" [61].

II.3 Knowledge Based

We have been discussing content based and collaborative based recommendation systems so far. They cover a large percentage of recommendation systems area but not all. Weak part of collaborative recommendation systems are lack of user preference data cases. Some rare life events, buying a house or changing the car does not happen so frequently on life time. That is why collaborative systems are weak in situations like this. Weak part of content based systems are their lack of tracking user profile changes. For example consider a user who was a runner and bought lots of Adidas branded shoes but later he had a traffic accident and lost his legs. If content based system recommends more shoes to this user it would be a big failure.

Knowledge based systems can handle the situations discussed in the previous paragraph. In [57] and [25] it is stated that "Recommenders that rely on knowledge sources not exploited by collaborative and content-based approaches are by default defined as knowledge-based recommenders" .

An example of knowledge-based application is VITA. VITA is a financial service recommendation engine. VITA is developed by CWAdvisor method discussed in [26]. In sales dialogs, VITA serves as a sales representative to customers.

Knowledge-based recommendation systems are superior to collaborative systems in terms of user-item matrix data at the beginning. Knowledge-based systems operates on the data gathered by data mining systems and that is why cold-start problem seen in collaborative systems is not a problem for it.

II.4 Hybrid Recommendation Systems

In knowledge base part we mentioned about the weak parts of collaborative filtering and content based methods with examples and also we discussed their advantages and disadvantages by using various examples from the literature. As discussed in [57], hybrid recommendation systems are combination of various recommendation systems(at least two) and benefit from strength of each one of them. Since strength of one of them can be weakness of another one, hybrid recommendation systems' performance usually overcomes them in terms of performance and accuracy.

According to [19], types of hybrid recommendation systems can be separated into seven categories based on their mixture type.

In our recommendation system framework, RECO, we are using weighted hybrid recommendation system. As seen on Table 3, weighted hybrid recommendation systems works by combining the scores of various different recommendation systems and that is what we are doing. We are combining 12 different recommendation algorithms based on their scores and producing an output accordingly.

TableII.3: Types of Hybrid Recommendation Systems

Type of Hybrid Recommendation System	Description
Weighted	Different recommendation systems are combined based on their score numerically
Switching	One of the recommendation systems is chosen by the system and applied
Mixed	Different recommenders' recommendations are presented together
Feature Combination	Features taken from various sources are combined and used by one algorithm
Feature Augmentation	A computed set of features by a recommendation system is input to another
Cascade	With the property of strict priority on recommenders, lower prioritized ones break the ties of higher prioritized ones in scoring.
Meta Level	A computed model by a recommendation system is input to another

II.4.1 Weighted

Weighted hybrid recommendation system combines scores of all the output of various recommendation systems that are actively used in the whole platform. In [22], Claypool uses a weighted hybrid recommendation system, PTango system. At first, PTango starts by assigning equal weights to the content based and collaborative based systems but later starts to adjust the weights according to the predictions on being confirmed or discomfirmed of the users. In [45], Pazzani combines not the scores but he represents outputs of various different recommendation systems as votes and then he chooses the final output based on a consensus.

In [18], It is said that the benefit of the weighted hybrid recommendation systems are their straightforward approach to the problem. The system itself can be analyzed quickly in terms of differentiating the weights easily and see which recommendation system is giving better results and which is not. The downside of this type as mentioned in [18] is it assumes all the algorithms used in the system are gives uniformly accurate results in the itemset but we know that is not the case unfortunately. A collaborative recommendation system can give weak results if cold-start problem has risen at the start.

II.4.2 Switching

Switching hybrid recommendation system uses different types of recommendation system algorithms based on item-level criterias. For instance in [14], DailyLearner, a switching hybrid recommendation system, is introduced. It firstly tries a content based recommendation approach and if it can not give results that are above a threshold in terms of confidence then collaborative filtering approach is tried.

What switching enables is providing results for which the previous algorithm could not have provided. For instance switching from content based system to collaborative system enables us to have results extracted by using relations between like minded users but only items that are similar to each other. For a user

owning a Swatch branded watch, content based could only recommend another similar valued watches but collaborative filtering approach could recommend Lacoste branded shoes or Diesel branded trousers because that is a general user behaviour.

In DailyLearner case, the system needed to apply an algorithm to see if it satisfies the conditions or not but in [56] the data and past given recommendations of each algorithm are being used to select the algorithm to apply on without applying each of them one after another.

Eventhough switching hybrid recommendation systems needs to pre-calculate which algorithm to use and thus use more resources than the others, they can outcome weaknesses of various recommendation algorithm by analyzing the data and applying the corresponding algorithm.

II.4.3 Mixed

Mixed hybrid recommendation system uses different kinds of recommendation system algorithms together to create a recommendation set as an output. Important thing to focus on in this method is it uses all the different algorithms together to create a result set not one by one.

In [52] a television viewing recommendation system, PTV is introduced. PTV uses content based and collaborative filtering together. At the beginning when users did not rated programs yet content based gives most of the recommendations and later when users start rating programs collaborative filtering starts to have its percentage of recommendations provided to the result set.

Other examples of mixed hybrid recommendation systems can be seen on [3] as ProfBuilder and PickAFlick in [20]. Lastly, it is important to mention that in this method a ranking system must be used at the last stage because different algorithms will be producing lots of result sets and we need to sort them with a ranking algorithm to prensent it to the last user.

II.4.4 Feature Combination

Feature Combination hybrid recommendation system uses the information from one algorithm in another one as a feature and by doing this it combines features and creates a merged version of algorithms. In [10], it is shown that a recommender both using user ratings and content features gave much better results when compared with a purely collaborative based system in terms of precision.

Hydra [53] is an example of this method used for movie recommendations. Hydra basically combines MovieLens rating data with IMDB content information data and later gives recommendations to users from one single system.

II.4.5 Feature Augmentation

Feature Augmentation hybrid recommendation system uses the output of an algorithm, ratings or classifications, as an input and produces result set accordingly. This method is popular because it enables to improve the core part of a system by taking its output as an input and further improve the results. The difference between Feature Combination and this method is, in feature combination raw data was being added but in this method processed data is being used.

In [36], content boosted collaborative filtering system is shown. First content based recommendation is done and after collaborative filtering makes predictions about users' ratings with extra data coming from content based's result set.

In [49], Usenet news filtering case is introduced. GroıupLens research team is using feature augmentation to improve the performance of Usenet news filtering. Firstly they are creating a knowledge based system. This knowledge base includes rules based on size of the messages or number of spelling errors made in a message and etc. Artificial bots that use this knowledge base are then sent to the system as users so that number of ratings of users are increases. And as a result they find that this method improves the email filtering.

II.4.6 Cascade

Cascade hybrid recommendation system is based prioritized recommendation algorithms. After the first one runs on data and produces an output, other algorithms can only change the ranking of this output but not the content adding or removal. There can also be cases like not needing other algorithms to run besides the first one if first one made an excellent job and there is no need to re-rank or first one failed hard and whatever the ranking is this result set will not be used.

In [18], EntreeC, a restaurant recommender system, uses cascaded hybrid recommendation system with knowledge based and collaborative filtering algorithms. EntreeC creates recommendations of restaurants by using its knowledge base and preferences of users. After finding the set of restaurants to be recommended, collaborative filtering is used to break the ties and further rank restaurants.

II.4.7 Meta Level

Meta level hybrid recommendation system uses a model generated by one of the recommendation system algorithms as input and sequentially goes like that. The difference of this method from feature augmentation is that in this model input from one algorithm is the model itself not the features extracted from the output of the algorithm.

Balabanovic created the first meta level hybrid recommendation system [8] [9], a web filtering system, Fab. Fab uses Rocchio's method [47] to perform a term vector model based on interests of user in content based filtering. The system gathers documents from the web based on the model generated from the interests of the users. In addition, Fab used cascade hybrid method with collaborative and content based recommendation systems.

In [35], a content based model is introduced. Each user describing the features for predicting the restaurants a user likes of. In [23], a two stage bayesian mixed-effects scheme: a content based naïve Bayes classifier is introduced. This

classifier is built for each user and then parameters of classifiers are linked across different users using regression.

Advantage of meta level hybrid recommendation is that specifically on content/collaborative coupled systems from the side of collaborative system, working on a model is faster when compared with raw data.

II.5 Problems in Recommendation Systems

Various hybrid recommendation systems exist and we explained them in detail. Those methods eliminate weaknesses of recommendation algorithms by combining various algorithms together. On the other hand, hybrids that use content based and collaborative filtering suffer from cold start problem regardless of their type. That is because they both need data at the start to give recommendations. Eventhough such a weakness exists, that kind of hybrids are popular because users of that hybrid method have the starting data already in their system usually. And of course knowledge based hybrid model can always be used as well for those who suffer from cold start problem since this method does not have the weakness of it.

Sparsity of the data is another serious problem. In [28] it is shown that sparsity of the data directly effects the collaborative filtering based systems' performance. Meta level hybrid recommendation model avoids sparsity problem. Meta level model compresses ratings over many examples and by doing that it enables to comparing between users easier.

CHAPTER III

TECHNOLOGIES USED

In this chapter we will be introducing the technologies used on the site, the website where RECO runs currently on. Those technologies are important because analysis of RECO and also performance metrics that we used also depends on those technologies. We will be also talking about why we chose them, their advantages and in what areas they helped us to solve which kind of problems.

III.1 Databases

First let's discuss types of databases. We can seperate databases into two groups, relational databases and non-relational databases. Relational databases appies ACID properties to themselves. A holds for **atomicity**. It applies all or nothing approach. If a transaction completes partially then that transaction is assumed not to be existed. Atomicty needs to be applied in every condition including power failures, crashes and etc. C holds for **consistency**. Consistency ensures that any valid transaction will change the state of the database from one state to another. That ensures that any data written into the database is valid. I holds for **isolation**. Isolation ensures that all transactions whether concurrently or serially applied, must result with the same state of the database. Isolation enables concurrency control. D holds for **durability**. Durability ensures that when a transaction is committed into the database, it should remain there whatever happens such as power loss or crashes.

Relational databases can be queried using SQL language. SQL is the short ver-

sion of Structured Query Language. SQL has been used many years frequently on the internet but not so frequent lately. Relational databases were even named SQL databases in the common talk. But when the internet started to be used in large scales, the data generated got larger exponentially and then the need for other kinds of databases arisen.

When the need of storing larger and larger data and accessing that data faster and faster started to become a problem, NOSQL databases were born. NOSQL holds for Not only SQL. As mentioned in [58], when the data grew bigger, additional columns were needed in relational databases but this caused sparsity in the databases because not all rows of data needed that extra column. Also the increased connectedness of the data was a weakness on relational databases because relations were getting larger and larger proportional with the big data and relational databases were using becoming very inefficient in terms of storage and time spent on per process.

NOSQL is a recent solution to those weaknesses of relational databases. It has four different types[54].

- Key – Value Stores
- Big Table
- Document Databases
- Graph Databases

III.1.1 Key – Value Stores

In this NOSQL type, each data is matched with a key. Keys are the index of tis type. Benefits of this type is its speed. If you know python programming language then key-value stores are like dictionaries, every key is mapped top a value. This type of NOSQL is mostly suitable when there is a need for very frequent number of reads are necessary and concurrency is needed.

Redis is extremely fast because it does not guarantee the durability of the data.

In traditional relational databases, durability is a mandatory property(D in ACID) but redis favors speed rather than durability.

In our website, we use redis, a key–value store, as a cache layer. We use it at cache level because its speed enables us to decrease response times of some specific tasks in large amounts. On the other hand we do not store any data that has utmost importance in redis because in a system failure, redis can lose data because it stores the mappings inside the memory.

Redis also supports master-slave replication. That enables scaling in redis. In our website, we run redis on a cluster of machines and use replication property frequently.

III.1.2 Big Table

A big table is a storage system designed to handle very large amounts, petabytes, of structured data. In big table, data is indexed by using row and column names which can be arbitrary strings. In [21], a big table is defined as "sparse, distributed, persistent multidimensional sorted map". Handling petabytes of data is not an easy job and that is why it works with clusters of machines and it depends on cluster management systems to manage resources on shared machines, handling machine failures, monitoring the status of machines and job scheduling. Big Table is implemented by Google and it uses GFS, Google File System, to store logs and data files in a distributed way.

Just like the Big Table that Google implemented, HBase is also a distributed, scalable, big data store implemented by Apache. HBase is a Hadoop database and just like BigTable runs on GFS, Hbase runs on HDFS, Hadoop FileSystem. [5] defines Hadoop as, "The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models". Hadoop is a really popular framework used by sites like Facebook, Twitter, Google and etc. Hbase takes advantage of Hadoop's simple distributing power of Hadoop. In [6], features of Hbase are mentioned as:

- Linear and modular scalability
- Strictly consistent reads and writes
- Automatic and configurable sharding of tables
- Automatic failover support between RegionServers.
- Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables.
- Easy to use Java API for client access.
- Block cache and Bloom Filters for real-time queries.
- Query predicate push down via server side Filters
- Thrift gateway and a REST-ful Web service that supports XML, Protobuf, and binary data encoding options
- Support for exporting metrics via the Hadoop metrics subsystem to files or Ganglia; or via JMX

HDFS is a filesystem used by Hbase and it is robust and fault-tolerant. HDFS was written with the inspiration from GFS. In [55] it is mentioned that Hadoop streaming enables users to create and run mappers or reducers on HDFS by using any programming language they want. Benefits of running map-reduce tasks on Hadoop is, Hadoop sorts the data according to the data needed by mappers and it improves data locality and thus improves the performance and decreases the amount of time spent. We said HDFS is fault tolerant and what that means is that if a crash occurs on one of the nodes, Hadoop restarts the related task in another node. That works because fault-tolerant system means share nothing principled architecture with the exception of relation between mappers and reducers.

Hbase is just a open source project built on Hadoop. There are other projects that works on Hadoop also[55]. You can see them below.

- **Hive**, a datawarehouse framework developed by Facebook used for ad hoc querying using SQL like language

- **Pig**, a high level data-flow language that produces sequences of map-reduce programs
- **Cascading**, a programming API for defining and executing fault-tolerant data processing workflows
- **Hbase**, a Apache Hadoop based project which is modeled on Google's Big Table database.

In the website we use Hbase too. We store logs in HBase to further analyze them later and to monitor realtime charts of various systems that have logging system embedded in. In the next section we will be discussing what kind of analytical solutions we are using on our Hbase database. Some examples of systems that have logging enabled and thus we have the ability of analyzing are:

- Visiting a page
- Percentage of video that the user currently watched so far on a video page
- Referral page of a visited page
- Type of recommendation system algorithm that generated the recommendation that user clicked on
- Loading more recommendations on a video page
- Sharing on social media actions on a video page
- Type of player and platform the user is using
- Recommendations generated on a video page

In the previous chapters we mentioned that in this big video website have 4 million videos and have 15000 active users on the site on average. Amount of logs generated are huge. 30-35 gb of log data is being generated and stored on hbase daily. Hbase provides realtime writing to the database not to disturrb the flow of logs and also realtime reading for monitoring. Below you can find an example log sent from the site to the Hbase.

http://logger.virgul.com/count?

- **m** = alive&
- **g** = h&
- **o** = websitename:n:25:5:7165777:52aa4330-861d-4a9e-a4ce-9bb5adac8ed3&
- **iv** = 111374684.853925677.1383504173.1389663450.1389795867.23&
- **wVID** = 111374684.853925677.1383504173.1389663450.1389795867.237165777.84&
- **info** = 7165777@31585974@31585974@5:5@;111374684.853925677.1383504173.1389663450

Each parameter type and related value of the parameter are bulleted above. Bold ones are representing the type of the parameters. "logger.virgul.com" is the backend server that takes our event requests, preprocesses the data before entering it accordingly to the Hbase. "m" holds for method name and in our example it is alive which is responsible for percentage of video that the user currently watched so far on a video page.

"g" holds for granularity and in our case it is "h" meaning hourly. It could be "m" or "d" meaning minutely or daily also. This defines the counting part in Hbase. Hbase stores counts of some table properties on some time levels and that parameter is an input to Hbase for that specific job.

"o" holds the parametric information related with the currently active method and in our case it means this event is from the website and the currently active user who triggered this event watched 25% of the video whose id is 7165777 and whose category is 5.

"iv" holds for a unique number generated by the system based on the user. Unique user sessions can be identified based on this variable. This variable is generated by using Google Analytics cookies and identification of the user on the site.

"wVID" holds for a unique number generated for a video and a user. This number identifies unique video watches of unique users. This can be thougth as an extension of iv with video information.

All the events that are logged behave similarly. Each have their own parameters defining the data and event info they hold. Logging every event triggered is really important. It helps analyzing various features or parts of the site. We can have answers to questions like "Which videos have been shared most on some specific social media site in a specific date" or "How many videos are watched by mothers on average". We can answer those questions because we have all the events of video watches, social media sharings and also we have the data of surveys asking like "are you a mother?" and by combining those data, Hbase can provide us all the answers that we need. But beware of the fact that Hbase is just a storage, we need to extract the data from there and analyze it accordingly to be able to answer those questions. In the next chapter we will be talking about this.

III.1.3 Document Databases

Primary concern of document databases are to be able to store the big data and achieve good query performance[29]. Unlike Big Table case, document databases does not try to achieve high performances on reading in a frequent way and writing concurrently.

Right now there are two major examples of document database systems, CouchDB and MongoDB.

CouchDB is a fault tolerant and flexible document database system and it works with JSON data formats. CouchDB comply with ACID properties that we mentioned before. It only provides a HTTP REST interface to read and write data.

MongoDB is a non-relational document database system but it includes lots of functionalities of relational databases as well. MongoDB uses indexing and also it has a powerful query language just like the SQL in relational databases.Mongo DB uses BSON data format. BSON is binary encoded serialization of JSON data format. In [54], it is mentioned that if the data stored is bigger than 50GB then access speed of MongoDB is ten times greater than traditional relational

database and that is why companies dealing with large amounts of data are prefering MongoDB.

In our website, we use MongoDB as well. One of the down sides of relational databases was when the data got larger, relational databases started becoming more sparse and that is because not all data have all the properties in each sample. This is exactly the reason we preferred MongoDB in our website because we are using MongoDB for storing recommendations and not all videos have the same amount of recommendations and even they are the same they do not have the same amount of information about the recommended videos and that is why we preferred MongoDB.

III.1.4 Graph Databases

Graph databases have three main components. Nodes, Edges and properties assigned to nodes and edges as key-value stores. By using these three components any relation can be defined in an optimized way and also accessing data gets a lot easier because database does not need to fetch all the indexes and cross check them but only traverse along the graph and that's all.

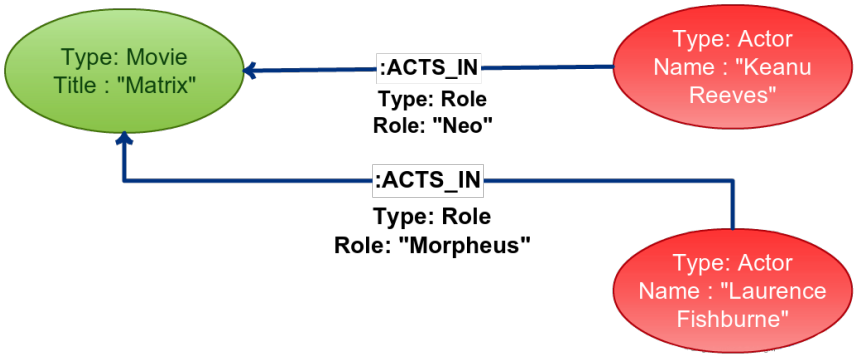
In daily life, graphs can be applied to many areas such as:

- Social Networks (Relationships with other people)
- Route finding (Shortest paths on traffic)
- Recommendations (Online shopping, watching movies, etc)
- Logistics (Optimizing the paths)

As seen on examples, graphs can be applied nearly everywhere. It is the structure of relations in real life. So by using it in our applications we are simply copying from the best, the nature.

One of the leading graph databases on the market currently is Neo4j. It is being developed by Neo Technologies and it has proven itself with its performance and

Figure III.1: Data Modeling in Neo4j



ease of use. Eventhough it is developed with java, there are lots of ports to other languages as well. Neo4j introduces two ways of interaction with the database. First one is through directly accessing to the files of the database and modify it by using the preferred programming language with the api Neo4j provides or access to the database via http interface and insert delete according to the need of use. One thing to mention is http REST interface is very slow when compared with the direct access because of extra layers. On the other hand if it is being used in a dynamic site http REST interface can have more advantage.

Now lets look into the data modeling of Neo4j. We will be introducing the abilities of this graph database and show some query examples from it.

As seen on Figure 3.1, Neo4j have nodes and relations and they both have properties assigned to them. These properties can be used on queries to differentiate the nodes. And as seen above, different nodes can have different types of properties as well. This reduces the amount of data stored because just the info that's needed and associated with the related node or relation is saved. This is a property of NOSQL databases unlike relational databases we do not need to store empty values on samples of data just because we have that table and that is why the sparseness of the database is eliminated on NOSQL databases in general. On the other hand, we may not have sparseness in our data but we may have duplicates of data because of denormalization. Denormalization is a trade off to improve query processing time while incresing the space complexity of the system.

Now let's look into the query part of Neo4j. Cypher is the name of the query language Neo4j uses by default. But it is not the only type of language that can be used for querying. There are various graph traversal languages which gives a more abstract and improved ease of use solutions for graph databases. One of them which we are also using in RECO is Gremlin graph traversal language developed by Tinkerpop. As mentioned in [31], Gremlin is less friendlier for a SQL type of person when compared with Cypher but Gremlin beats the Cypher in terms of FOAF(friends of a friend) type queries.

Now let's give some examples of Cypher queries by using the graph from Figure 3.1 .

```
MATCH (movie:Movie title:"Matrix")
RETURN movie;
```

In the above query, we are trying to find nodes with Movie type and having the title "Matrix". MATCH and RETURN are constants used in every Cypher query like SELECT, FROM, WHERE in SQL. MATCH identifies a filtering rule on the graph and RETURN returns the entities given after applying the filtering rule on the graph.

```
MATCH (movie:Movie title:"The Matrix")
RETURN movie.id, movie.title;
```

In the above query, we are trying to find nodes with Movie type and having the title "Matrix" and we want "id" and "title" properties of those nodes only. This example shows properties of nodes can be accessed like "node.property" as we do in classes in object oriented languages.

```
MATCH (m:Movie title:"Matrix")<-[:ACTS_IN]-(actor)
RETURN actor;
```

In the above query, we are trying to get all the actors in the graph which acted

in a movie with the title “Matrix”. Here we are filtering according to a specific relation named “:ACTS_IN” between actor and movie nodes. The syntax is standart. The arrow shows the flow of the relation path. One can read it like “Actors -> acts_in -> movies”

```
MATCH (m:Movie title:"The Matrix")<-[:ACTS_IN]-(actor)
RETURN count(*);
```

In the above query, we are trying to get number of actors who acted in the movie whose title is “Matrix”. This is an example of aggregation in Cypher. After filtering the graph, in the RETURN statement we are counting the number of results.

Those examples shows the basic work logic of the Cypher. We said that Cypher is not the only query language for Neo4j and Gremlin can also be used to query Neo4j databases, which we used in RECO also. Gremlin uses pipes to perform graph traversals. Pipes enable the traversals of complex graphs easily. Below you can see some databases that are supported by Gremlin.

- TinkerGraph – In memory graph
- OrientDB – Graph database
- Neo4j – Graph database
- DEX – Graph database
- Titan – Graph database
- Faunus – Graph analytics engine
- InfiniteGraph – Graph database
- Rexster – Graph server
- Sesame 2.x – Compliant RDF stores

Gremlin can be used for graph query, analysis, manipulation cases. By default Gremlin supports Java and Groovy implementations and the examples that we will be showing below are Groovy based examples.

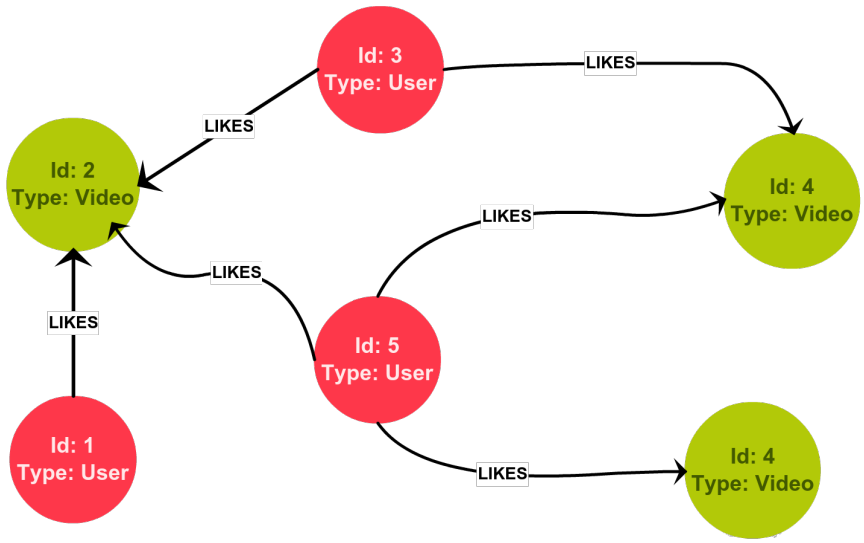
```
m = [:]
g.v(1).out('likes').in('likes').out('likes').groupCount(m)
m.sort-it.value
```

In the above query, first we are creating an dictionary named "m". "g" is our graph and "g.v(1)" is returning the node in the graph with id 1. Remember the pipeline structure, now we have one node in our pipe and we are delivering it to the next cycle of computation. "out" and "in" are relational type of methods. "out" method only gets nodes which are connected to the source node with edges going out from the source node and "in" method only gets nodes which are connected to the source node with edges coming in to the source node. "out(some_property)" will get nodes which are connected to the source node with edges going out from the source node and having "some_property". "groupCount(m)" works like groupby in sql and it groups nodes and counts them and writes them into m. And at the end we are sorting our dictionary descending.

Actually the query above is a simple collaborative filtering algorithm in three lines. In Figure 3.2, there is a graph that we can use this code on and simulate the execution.

- First of all we get the node with id attribute 1 and it is a user node on the left-bottom of the Figure 4.
- Then we find nodes which have edges coming from node-id-1 and there is only one and it is the node with the id attribute 2 and it is a video node. (user-item graph)
- Then we find nodes which have edges going into node-id-2 and there are 3 nodes that satisfy this property and they are node-id-1, node-id-3, node-id-5.

Figure III.2: Gremlin Query Example Visualized



- Then we find nodes which have edges coming from node-id-1, node-id-3, node-id-5. There are 6 nodes satisfying this rule and if we group them according to their id we find that we have three node-id-2, two node-id-4, one node-id-4. At the end we sort this dictionary of node results and find the ranking of the collaborative filtering.
- Titan – Graph database
- Faunus – Graph analytics engine
- InfiniteGraph – Graph database
- Rexster – Graph server
- Sesame 2.x – Compliant RDF stores

In our website, we use Neo4j and Gremlin for RECO's collaborative filtering part (remember RECO is a weighted hybrid recommendation framework). In the example we used here collaborative filtering was being applied at one level. By one level, we mean videos that my friends watched and by two level, we mean videos that friends of my friends watched. It can be thought as the depth of search in collaborative filtering, the deeper we go, the larger and diverse the result set would be.

III.2 Analyzing Tools

In the databases part, we mentioned that in the website we store our event logs in Hbase. 15.000 real-time active users on the site on average and 4 million video pages being visited by those users creates lots of event data. Since every new feature or algorithm we create needs to be evaluated to decide whether it is a better version or not, we need some analytical evaluation metrics and tools to extract those metrics from the big data being generated, 30-40 TB daily. Now let's give explanations of these tools.

III.2.1 Scalding

Scalding is a Scala library that makes it easy to create and run Hadoop map-reduce jobs. In databases part and in Big Tables sub part, we mentioned about Cascading which was a programming API for defining and executing fault-tolerant data processing workflows. What Cascading simply does is abstracting Hadoop's low level details. Scalding is built on top of Cascading and thus it abstracts the low level details of Hadoop and also it has the resources of scala behind it. Scalding can be used for custom ad targeting algorithms, market insight, click prediction, traffic quality to pagerank on the related website's page graph, ad-hoc data analysis jobs and etc ..

Scalding is an open source project, developed by Twitter. It has two types of APIs, field based and type based. In our website we are using field based type of APIs of scalding. The main reason we chose scalding on our site is because scalding have the features below:

- Functional programming and it's natural way of implementing data flows
- Power of Scala which runs on JVM
- Statistically typed meaning there will be no type errors on runtime

Scalding is already being used by large companies which can be seen below for

their Hadoop related data anaylsing workflows. Some companies using Scalding are: Twitter, Etsy, eBay, SoundCloud, LinkedIn, Stripe and etc.

Just like Gremlin, Scalding also works with pipes. Since scalding is used for analysing big data, it uses map-reduce algorithms. Map-reduce is simply dividing a computation into two, mapping and reducing parts. In mapping part, the data that needs to be processed are assigned to different entities for computation so a paralel execution of tasks run side by side and when all of them are finished, reducer takes the processed data, gathers them and produces the expected output. If big data needs to be processed, map-reduce allows us to do the same computation in a short amount of time.

Now to understand Scalding, one needs to understand Cascading first. Cascading consists of flows. A flow consists of a source tap, a sink tap and pipes connecting them. Pipes hold a particular transformation on the input data. One can create new complex pipes by combining them. Figure 3.3 taken from [43] shows this.

In figure 3.4 that's taken from [59], you can see a word count example in scalding. TextLine creates the first pipe and it includes the input. FlatMap method takes the first pipe as input and it splits each line in first pipe according to tokenize function and creates the second pipe. GroupBy method creates a third pipe and groups words and by using size attribute it counts each unique words' appearance in the second pipe and creates a third pipe. Write method takes third pipe as input and writes it into the output sink. As you see scala embedds java libraries by default and scalding has the power of huge java libraries behind it also.

In our website, we use scalding for analyzing daily event logs to understand the behaviours of users, observing results of a test, getting performance metrics of our recommendation system RECO, number of unique videos recommended that day and etc.

Figure III.3: Word count example of Map-reduce and Cascading concepts

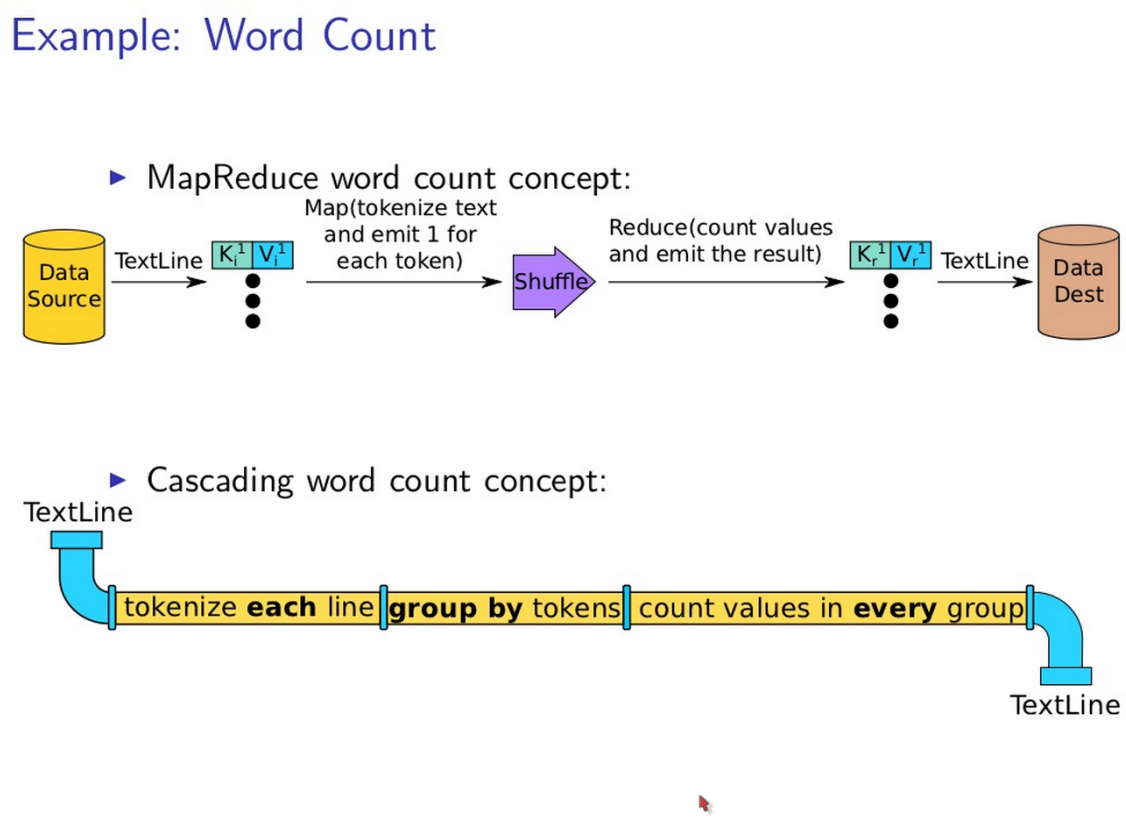


Figure III.4: Word count example of Scalding

```
package com.twitter.scalding.examples

import com.twitter.scalding._

class WordCountJob(args : Args) extends Job(args) {
  TextLine( args("input") )
    .flatMap('line -> 'word) { line : String => tokenize(line) }
    .groupBy('word) { _.size }
    .write( Tsv( args("output") ) )

  // Split a piece of text into individual words.
  def tokenize(text : String) : Array[String] = {
    // Lowercase each word and remove punctuation.
    text.toLowerCase.replaceAll("[^a-zA-Z0-9\\s]", "").split("\\s+")
  }
}
```

III.2.2 Python libraries

Python is an scripting language and it is really easy to read and write programs with it. There are some general tools to deal with specific areas like R language for statiscal analysis, Matlab for image processing, machine learning, Excel for table structured data and etc. Python has a library for dealing with each of those. It is like a handset of researcher one can say. It is really import to be able to deal with all those areas in one language because output of one tool can be an input to another and also readability, maintainability is easier and the time it takes to learn the required tool takes much less time.

III.2.2.1 Pandas

Pandas is an open sourced and BSD licenced open source library of python providing easy to use data analysis tools and data structures. It enables one to focus on the research rather than programming. It can read various formatted files and process them intelligently such as handling missing data automatically or reshaping the arrays in a fast and robust way since it uses numpy arrays. In addition, pandas is really fast because some critical area codes are written in Cython or C to improve performance.

As mentioned in [42], pandas is really well suited for:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

Pandas have three main data structures, Series, DataFrames and Panels. Series is a one dimensional data structure which includes not only data but also

axis labels too. DataFrame is a two dimensional, tabular data structure which includes axis labels on both rows and columns. DataFrame is much more than R languages' "data.frame" thanks to abilities of numpy integrated by default in pandas. Lastly Panel is a three dimensional data structure used less commonly when compared with DataFrame and Series but if one needs it, pandas provide it.

In our website, we use pandas for analyzing the data we got after running scalding on Hbase. It makes our job much easier. Instead of dealing with file problems, missing data problems we focus on the research, the algorithm. The time complexity of our analyzing jobs also is faster because of pandas.

III.2.2.2 Matplotlib

Matplotlib is a python 2D plotting library[33].It can be used in python scripts. With just few lines of code, one can draw plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc.

In our website, we use matplotlib for plotting the results. Not everyone have the ability to read numeric values and make a statement about it so visual representation of data matters a lot. We are using matplotlib for that. Pandas library has a direct connection to matplotlib such that every data structure except Panel (remember 2D) can be plotted directly by calling plot method of them.

III.2.2.3 Scikit-learn

Scikit-learn is a machine learning library for python. It is built on numpy, scipy and matplotlib libraries of python. It can be used for data mining, data analysis and various machine learning methods to apply on the related data.

In our website, we use scikit-learn for finding weights of different recommendation algorithms by using regression analysis methods. We will be talking about this in the next chapter in detail.

CHAPTER IV

RECO

RECO is the name of our weighted hybrid recommendation system framework currently running on our website. In this chapter, we are going to talk about the recommendation system used in website before RECO and why we needed to change it and in what ways RECO is better than the old system.

IV.1 Myrrix

Before RECO, Myrrix was being used in the website. As mentioned in [38], Myrrix is a complete, real-time, scalable clustering and recommender system, evolved from Apache Mahout. Currently myrrix is migrated into Cloudera but in summer of 2012 Myrrix was a standalone recommendation engine and it was still in very early versions and it was lacking documentation at the time. In addition, it was not open sourced.

In İzlesene, Myrrix was being fed by user-watched_video matrix and recommendations for a video were being taken from it. The challenge was that the time that myrrix completed a cycle of computation was taking nearly a day and that is why our daily recommendations were static. We could only change the recommendations of a video manually which was a serious amount of job when one consider that we have 4 million videos. When the person responsible for handling Myrrix left the company, we did not have any know-how about myrrix and when recommendations were not as successful as possible, we could not tweak algorithm parameters in Myrrix.

Our metric for evaluating the success of recommendation system in İzlesene is CTR, click through rate and it basically calculates for the case of a video being watched by a user, how many times we succeeded making the user click to the recommendations of that video as a percentage. When we were using Myrrix, this value for the whole İzlesene was %15.

We wanted to increase this CTR value. In addition we also wanted various algorithms running with different weights on different categories since not every category is in the same level of recommendation with each other. For example animals category can use content based recommendation because generic type of animal videos are already known and people do not make playlists of specific types of animals and watch them in general. On the other hand in music category every person's taste is different and this category needs a heavy weighted collaborative filtering algorithm.

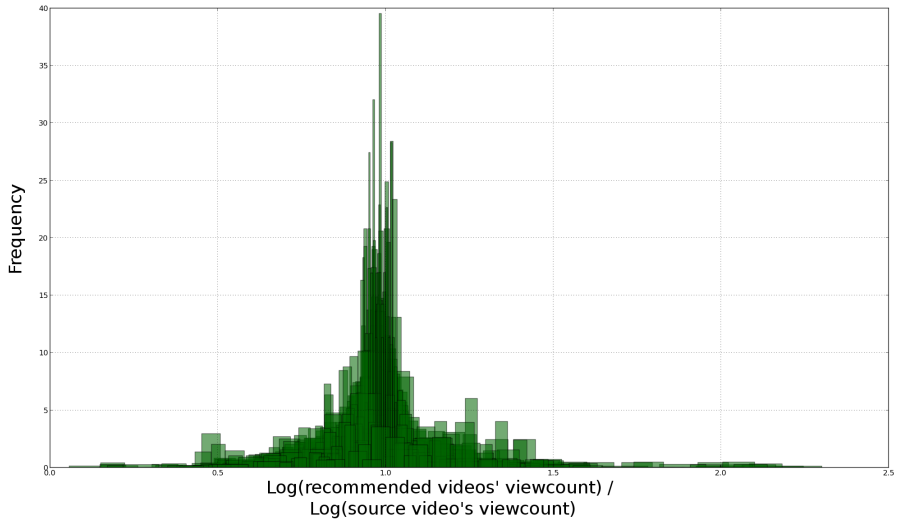
Myrrix was using latent factor analysis and more specifically a modified version of alternating least squares algorithm for matrix factorization. We wanted to use open sourced, well-documented, scalable (Myrrix was not scalable at that moment or our know-how didn't allow us because we could not see the codes) and fast system. That is why we started searching for technologies being used by other companies that are using real-time recommendation systems just like what we need.

IV.2 Reverse engineering Youtube

Youtube is the leader website in terms of video and it uses recommendation systems for video content heavily. So before constructing our own recommendation system, we wanted to investigate the way Youtube gives recommendations to its users.

We wrote a script that runs through Youtube and gets viewcount and name of recommendations of a video. Result of our script showed that Youtube gives 20 - 25 recommendations per video at first, if user clicks on "show more recommendations" (for videos that have more than 20-25 recommendations) user can get

Figure IV.1: Relation between the videos and their recommendations



more recommendations but we just analyzed the first part because they are the ones user is mostly interested in just like the first page of a resultset a search engine returns. Firstly our script ran on 500 unique videos by jumping from one video's recommendation into another. But when we ran the same script on again and again, we realized that Youtube gives different videos and even different rankings on the same set of videos for the same video as recommendations. That information made us change the script in such a way that the script takes the recommendations of a video 20 times with 30 second gaps between them. With that approach we got all the possible recommendations Youtube was providing for a specific video and we got 500*20*20 sized dataset. We were trying to find a pattern on the source video and the recommendations Youtube given. We tried various approaches on linear level but we could not find a pattern. Later we tried log scale of view counts and we found a pattern.

In Figure 4.1, we are looking into a histogram and on x axis we have logarithm of recommended videos' viewcounts divided with logarithm of source videos's viewcounts and on y axis we have the amount of videos for a specific range or frequency of related bin on that histogram. We can easily see that this looks like a gaussian distribution, good news! Confidence levels of histogram in figure

4.1 can be seen below.

[0.5 – 1.5] %99.6196705184

[0.6 – 1.4] %98.9122050324

[0.7 – 1.3] %97.6309220126

Now we got our confidence levels, we found our pattern. This was the the inspiration for us to create a specific algorithm in RECO. The algorithm works with content based approach and filters according to viewcount relation between source video and candidate recommendations by checking if their viewcounts' logarithms' dvision are in range [0.5 – 1.5] or not.

We ran the the script we wrote time to time to see if there was a difference on our results and when were doing that we realized that Youtube was sometimes recommending videos outside of [0.5-1.5] range and more specifically videos that have just been uploaded in the recommendations of some popular videos. That was another inspiration for us to create a specific algorithm. The reason Youtube was doing this was to give a chance to the video that have just been uploaded to be part of the user-video graph in terms of connectedness and also not to disqualify newly uploaded videos under popular ones. If the video did not succeeded being a popular video or could not attract users' attention then Youtube was no more providing that video in recommendations.

IV.3 Weights

RECO is a weighted recommendation system framework and that is why it has two major important structures which are its algorithms and these algorithms' weights. Before, we mentioned that we use scikit-learn library for regression analysis on weights. Time to time, we update weights and for updating we either use machine learning or manually change some weights in terms of adapting to a strict change.

For example once a year in Turkey, Ramadan is celebrated and in this bairam

people stop eating till evening and that is why housewives concentrate on meals to create great foods. In Ramadan, in the website, our meal-recipes category jumps like %400 just in a moment and goes like that for one month. To adapt to those changes accurately sometimes we also manually change the weights. That is a feature of RECO, being able to manipulate the weights fast and easy to adapt to fast changes.

IV.4 Algorithms

Earlier we mentioned that RECO is a weighted hybrid recommendation system framework thus it has various algorithms running inside. Each of those algorithms provide different abilities to the RECO. In this part we are going to explain each of them in detail about how they operate and what advantages they provide to the RECO.

IV.4.1 Collaborative Filtering Algorithm

As we mentioned before, in the website we use neo4j for storing user behaviours of video watching and amount of time they watch each video. These two are really valuable for us because our collaborative filtering algorithm makes use of them heavily. When we need to find candidate recommended videos for a source video, we look into the source video's FOAF(friend of a friend) related videos and do that twice because in our first attempt the videos we get are in a potentially biased area (we experimented that in our first attempt of this algorithm) and types of videos do not change a lot and that is the reason we go one more level of FOAF relation and get a huge dataset of candidate videos. The reason we stop at second level is because we reached to the point where cost of computation exceeding the benefits because of the size of the graph. One can go further on the level of computation if s/he does not care about the fetching time of the recommendations. We did not have that luxury because RECO stands as a robust and fast recommendation framework.

In addition, there is a term called trueview which means the percentage of video a

user has watched after visiting that video from another video's recommendations. This is important because it is an effective metric to calculate whether user was pleased with the recommendation or not. That is why we keep trueview value on edges in our graph and while we are traversing we only select edges that are equal or more than %50 (this value also is a result of experimentation. We tried %25 and we had lots of videos with low quality and we tried %75 and quality of videos were really high but quantity was really low) video completion rate. That makes our collaborative filtering algorithm really powerful because we eliminate the videos users came across to and watched some time but did not like. This is the major power of this algorithm. In our graph we only select videos that a user liked while watching it.

Our collaborative filtering algorithm runs on AKKA which is a toolkit for highly concurrent, distributed and fault tolerant applications on JVM. We use AKKA because concurrency and being distributed are really important key points for us. There are 15000 active users at a moment on our website and the system needs to be able to provide the same quality of service to each one of them and also if this number increases, system must be able to be scale so that quality of service would not fall down.

AKKA handles concurrency with actor-based model. In actor-based models there are instances of different types of actors and they communicate by sending message to each other. AKKA is fault-tolerant because if a message never arrives to its destination this is not a problem because I/O blocking is not an issue. In AKKA, neo4j communicates with mysql and redis to create or update new nodes and relations. Every video in Neo4j have one week timeout by default and for every video-watch-event happening for that video we decrease this timeout variable. When this value hits zero we update its viewcount, relations and etc by making queries to our backend services. This delay mechanism enables us not to check for not used, not popular videos and focus on the popular, up-to-date, the ones users are mostly interested in kind of videos. That is a feature of this algorithm, efficient distribution of the resources.

IV.4.2 Content Based Algorithm

Content based recommendation uses the features extracted from the content of the item and in our case items were videos. Features we used were:

1. Name of the video
2. Artist of the video
3. Category of the video
4. View-count of the video
5. Likes of the video from social networks
6. Genre of the video ...

We imported those features of every video uploaded into İzlesene into the Solr. Solr uses indexes and it enables to query large amounts of data sets and retrieve the data in a short time. Solr also have the ability to boost some specific fields in the query which means we can ask Solr to fetch us results in the way we prioritize them. For instance we can say that "Bring me the videos that have 'dance' in its name with %80 priority and 'garden' in its name with %20 priority" and Solr will give put videos having "dance" keyword in front of videos having "garden" in its name in the rankings. It is a way of boosting our query fields which we used a lot in İzlesene.

For content based recommendation we used two algorithms. First algorithm just directly queries Solr according to the related video's name, artist, view-count and etc. On the other hand, our second algorithm queries Solr by boosting videos whose view-count's logarithm divided by source video's view-count's logarithm is between [0.5 - 1.5]. We mentioned about this algorithm before in Reverse engineering Youtube part.

In evaluation phase we will show that that second algorithm really works better than the first one. Boosting the right field makes the search results much more successful and accurate.

IV.4.3 Music Algorithm

Music is a major category in the website. %80 of videos belong to music category in İzlesene. We have genres, lyrics, artist info of a large amount of videos in our database and every music piece has one artist at least but that artist may not be in the related video's feature set. That is why we crawl various sites by cross-matching lyrics of the video, name of the video and query those sites to retrieve the artist of the video.

This crawling and retrieving the artists of the videos process works as a job at the background and fills the feature set of music category videos. What this algorithm does is to simply search for that artist's other trending or popular songs. Trend is measured by calculating the ratio of increase of the view-count of the related video. Popularity is simply the amount of view-count a video has. Trending metric is superior to the popular metric for music videos in ranking because a new video being watched in an increasing speed is always better to push to the end user rather than an old trending song. This enables us to move higher on search engines and also to be part of that viral view-count increase of that specific video.

This algorithm only works for music category.

IV.4.4 Boosted Algorithm

We mentioned about the power of boosting in content based algorithm part. This algorithm is again about boosting but boosting manually the content.

In this big video site, we have a department responsible from the content. Every video uploaded to our website must be approved by them and also they follow trends, popular songs in the world and when they see a trend in one song they can manually tell RECO to put this video in recommendations more than others by using this algorithm. This is mainly to catch the wind and go viral in a short amount of time.

This algorithm takes place after every algorithm finished executing, returned

their rankings, merged them and after then this algorithm starts working. We have a database which stores boosted videos and rules to show them at which videos' recommendations. If there is a match with the related video and the rules provided with the boosted videos in the database then we push those boosted videos into the resulting set of recommended videos. Rules are like:

- Show this boosted video in that specific genre
- Show this boosted video in every videos uploaded by that user
- Show this boosted video in that specific tagged videos ...

IV.4.5 Unseen Algorithm

In reverse engineering Youtube part, we mentioned that Youtube gave a chance to videos that was just uploaded into them in terms of keeping the connectedness of the graph. We were inspired by this approach and we created unseen algorithm which detects not connected videos from logs and gathers them in a pool and feeds the system with those videos. If they join the system after this process, they are out of pool, if not this keeps going.

If not boosted by boosted algorithm, newly added videos have very low chances of being shown in recommendations unless they are trending. Unseen algorithm overcomes this disadvantage. It makes videos which do not have or very low likes from social media, very low view-counts to be shown on trending or popular videos' recommendations. It is a way of giving them a chance to be trending or popular.

Those videos that are being shown on recommendations of popular or trending videos will be fading by time. If they are not watched by users after we give them this chance, their edge weights gets lower and lower and when the graph edge elimination process starts their edge connecting them to the strongly connected component of İzlesene video graph will be erased unfortunately.

After we started running this algorithm, number of unique recommended videos daily increased significantly. We will be talking about that in evaluation phase.

CHAPTER V

EVALUATION

Evaluating a recommendation system is a tough but important job because feeding the system with feedbacks generated from evaluations is a must to further improve the success rate of the framework. In addition, evaluating a recommendation system for music videos is another tough job because a recommended video can be liked or not liked based on the taste of the user, so one can say that different profiles of users can prefer different recommendations.

We will be evaluating RECO in two main categories, accuracy and execution. On accuracy part, we will be explaining why results of RECO are more accurate and on the execution part we will be explaining what benefits RECO provides in terms of execution time and resources.

V.1 Accuracy

As we mentioned RECO uses various algorithms for various categories. Our evaluation process breaks into each algorithm-category pair so that we can get independent evaluation results for every algorithm-category pair.

We used two main evaluation metrics in RECO, CTR(Click Through Rate) and NUVR(Number of Unique Video Recommendations). Scalding framework and pandas library was used for generating and processing those metrics. Accuracy of the system can be seen by inspecting the CTR and NUVR metrics which will be explained in detailed in the subsections below.

TableV.1: A sample algorithm-category CTR spanning five days and all categories versus each algorithm

Category	Algorithm	Day-1	Day-2	Day-3	Day-4	Day-5
All	All	%17.86	%16.68	%16.68	%16.68	%16.68
All	boosted	%17.57	%13.89	%15.40	%14.27	%13.87
All	Solr-V2	%39.92	%41.44	%42.93	%44.95	%42.86
All	Music	%10.00	%12.00	%11.50	%13.19	%11.70
All	Collaborative Filtering	%16.84	%16.21	%16.55	%16.38	%16.92
All	Unseen	%10.00	%10.06	%10.40	%7.55	%14.21

V.1.1 CTR

The ratio of videos which are recommended to and being watched by the user and the total amount of videos being watched is called CTR. Basically CTR is a metric that shows how successful our recommendations are based on the choice of the users. We can see CTR values of algorithm-category pairs in real-time therefore we can observe the accuracy of the RECO in real time.

To be more spesific, let's assume in a period of time users watched 100 videos and 20 of those videos were reached by the users from the recommendations generated from RECO. In this spesific case our CTR would be %20.

Below you can see table 5.1 which is a sample from our real-time algorithm-category pair.

Before RECO, our CTR values were around %15. There is an important issue that needs to be clarified, after old recommendation system was discarded we added a new feature to the big video site which is radio format on music category which was showing recommendations as a standart video shows but when the video was over it was directly going to the next song. This is a very critical major reason for CTR to fall down. Music category is %80 of whole site, huge amount. Because users have no time to click on recommendations and what is worse is users usually open radio and then leave and that causes even a faster CTR drop rate. Even after that situation we managed to increase CTR by %1-%2 points, that is a huge success mostly thanks to filtered content based and

TableV.2: Unique generated recommendations / Total generated recommendations ratio

Days	Unique Count	Total Count	Ratio
Day 1	857391	4792443	%17.89
Day 2	813612	4608384	%17.68
Day 3	830067	4477253	%18.54
Day 4	820124	4327874	%18.95
Day 5	842195	4337502	%19.42

collaborative filtering algorithms.

Myrrix which was using matrix factorization had a CTR value of %15 while RECO had %17 CTR rate with radio feature enabled. This clearly shows that RECO, a hybrid recommendation framework, is giving better results than matrix factorization methods.

V.1.2 NUVR

This is a metric that started after RECO was born. That is why we have no past data before RECO to compare with the current data. On the other hand, we can comment on the current data by looking at the increase and decrease rate of it.

NUVR is directly related with the unseen algorithm. What unseen algorithm does is to increase the connectedness of the graph and by doing that making users watch videos that have never been watched before. A denser graph means better and more meaningful recommendations. In addition, NUVR itself does not mean much to us because what matters is ratio of unique videos that were recommended to users over the set of recommended videos.

Below you can see table 5.2 which is a sample from NUVR metric. As you can see we are generating %20 of the videos that we generate as recommendation set to users are unique videos.

V.2 Execution

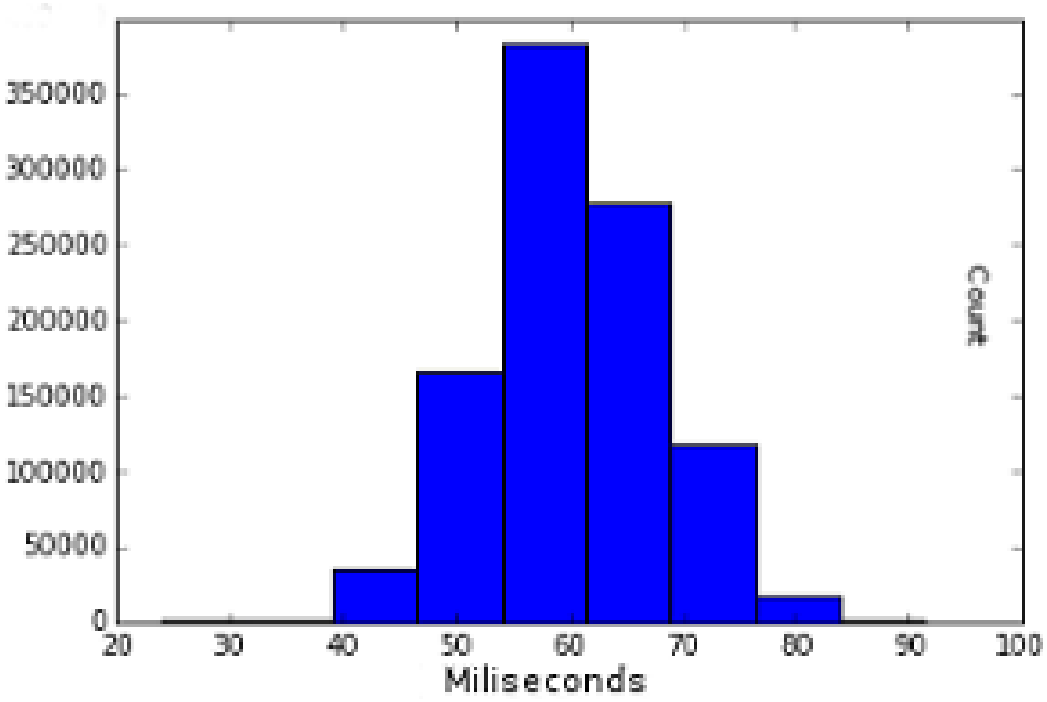
Execution time of the system can be analysed in two ways, the time it takes to introduce the recommended videos to the user when user clicks on a video and the time it takes to generate the recommendations of a video at the back-end.

For the first part, if the system makes the user wait on the video page for introducing the recommendations it would be a bad user experience and user would leave the page. To prevent this, we added a cache layer, redis, to RECO and it provided nearly instant access(60ms on average) to the recommendations of a video from the perspective of the user as it can be seen on figure 5.1.

For the second part, there were lots parameters that is effecting the time it takes to generate a video's recommendations since RECO consists of multiple algorithms and modules. If the video is a popular one, one could expect that in our graph database the traversals we need to make to generate the collaborative filtering would take much more time when compared with a less popular video but in fact as we mentioned earlier we are saving every recommendation we generated into our database and until the video reaches a certain number of video watch count from that moment and so on we are not updating its recommendations (if it does not reach to it in one week, we update it). In addition if the traversal of the graph takes so much time because of inefficient resources on the machine at the moment, we time-out the process, return the old recommendations fetched from the database and push the video to the job queue to be re-evaluated later. That is why we can say that the performance of RECO is not affected by the overweight of graph traversal or inefficient resources of the machine from time to time.

For generating a reliable system and prevent over-fitting of the whole system, we used cross-validation in RECO's evaluation phase. Since evaluations were being calculated based on the logs, we were simply creating those cross validations in real time by providing users differently ranked set of videos. Differently ranked set of videos are either same video appearing in a different ranking than before in the recommended set or a video that has not been recommended before. By

Figure V.1: Histogram of load-times of the recommendations on the video pages



doing that and logging the ranking of each recommended video in the set, we tracked and cross validated our results in the processing part of the evaluation. We made our calculations for each group, which were divided based on each video's different ranking positions, separately and then compared and merged them.

Below we will be explaining those two metrics and some other analyses we made in detail.

V.3 Effects of Time on Recommendations

As we mentioned earlier, we measure everything in RECO. That includes the time a user watches a video and if user was directed to the video from a recommended video or not. So we wanted to evaluate the effects of time on recommendations to see if we could differentiate our recommendations to a user on different times with different recommendations.

We made an analysis based on our measurements of time and clustered the videos and compared it with our current system. We used one million lines of video watch logs and used cross-validation with a percentage of %20. We found that RECO was already clustering videos based on time since users had different motives on different times and RECO could cluster those motives very successfully. For example a worker was watching news videos before s/he goes to work and a student was watching game videos when s/he comes to home after school afternoon. RECO could even detect sudden behavioural changes, for instance in the afternoons general profile of housewives are watching TV shows or fashion videos but in Ramadan bairam they were watching meal recipe videos. Those general behavioural patterns were already discovered in RECO.

So we started thinking about cases like what if the same video was being watched by two different users on different motives at the same time. For example a game video can be a fun time for a gamer but can be a technical issue for a researcher. We tried to find a way to differ those two motives based on time but we saw that this is not possible because two different users watching same video with

different motives can not be distinguished unless they are generally observed by patterns on our graph, meaning they create paths with high weighted edges describing their different behaviours.

We can conclude that RECO favors majority's behaviour over minority.

V.4 Effects of Age Groups and Gender

In RECO, we also show surveys to users time to time and we save the answers of users to their specific profiles in HBase. This is totally for the needs of advertising works. We ask questions like which car brand would you prefer or are you interested in child related products or which age range do you fit in or what is your gender and etc.

We wanted to measure the behaviours of users based on gender and age groups of them by using those survey answers. Since not all of the users answer those surveys this analysis can not be generalized to all of the users but can guide us for future analyses.

We had two gender groups male, female and four age groups 16-25, 25-34, 34-45, 45- . First thing we noticed was the huge difference of video watch counts for females having age 34-45 against others in work hours. This was not a surprise for us because most of our users are housewives. In addition, 16-25 age group of males were watching lots of game and fashion videos in work hours. This also was not a surprise because male students' main interest areas are fashion and game category videos and since they do not work in day time they have the free time to watch those videos.

Our analysis was based on users who answered survey questions and we used 300.000 users in this analysis. In table 5.3 you can find amount of users based on age group and gender.

TableV.3: Number of users based on age group and gender

Gender	Age Group	Total Count
Male	16-25	34324
Male	25-34	31204
Male	34-45	53047
Male	45-	29204
Female	16-25	28084
Female	25-34	30906
Female	34-45	46806
Female	45-	56168

CHAPTER VI

CONCLUSION AND FUTURE WORK

içioz

VI.1 Conclusion

When we were creating RECO our aims were:

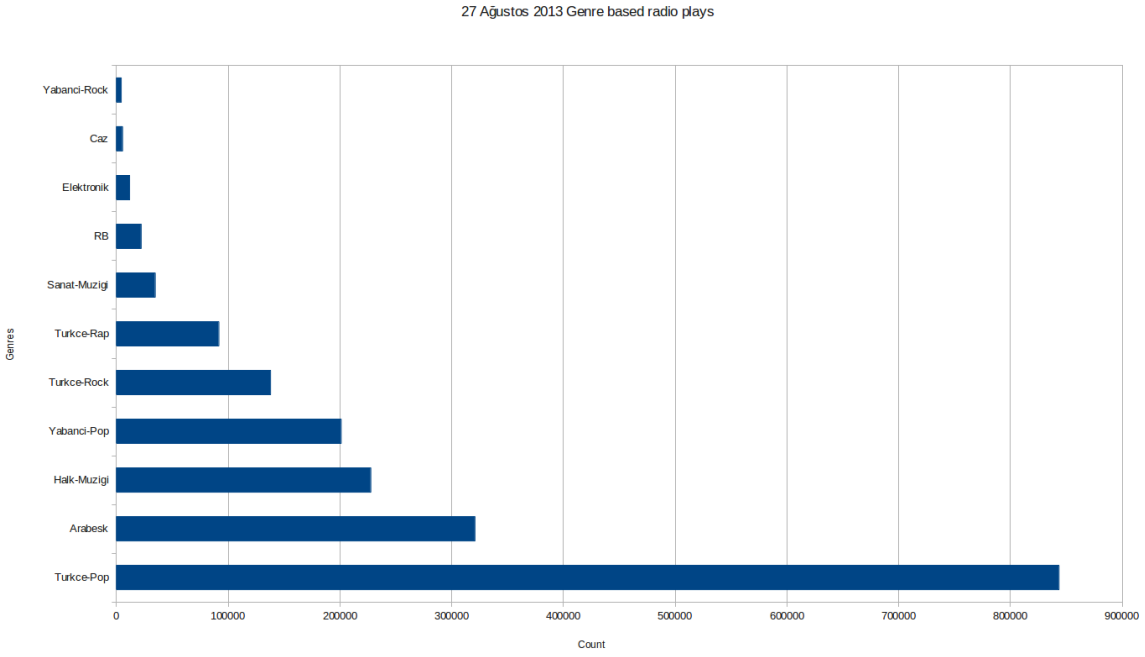
- To be able to find the video-watch behaviour patterns of users
- Being able to produce fast, accurate recommendations to users
- Creating a recommendation framework for video recommendation by using up to date technologies and state of the art algorithms

We managed doing these. Our framework is being run on a big video site right now and results are better than expected.

In this thesis, we explained various recommendation algorithms and focused on hybrid recommendation algorithms since RECO uses hybrid recommendation system algorithm. We explained the motives to build RECO, the system that were being used before and why it did not suit our needs and how we built RECO to suit those needs.

We talked about the technologies we used. We wanted to use the recent technologies and in every part of RECO a different recent technology is being used which were explained in detailed.

Figure VI.1: 27 August 2013 Genre based radio plays



Since RECO consists of many different algorithms, we explained each of them with their benefits and reasons for creating them and using in RECO.

In the end, we evaluated RECO based on its results compared with its predecessor and shared some interesting results.

To sum up, we created a robust and fast hybrid video recommendation framework using recent technologies which can handle one million users and four million videos easily.

VI.2 Future Work

RECO can be improved in two ways, adding a new successful algorithm or improving weight distribution method. Since we have enormously big data, we can easily extract data from our Hbase as shown on figure 6.1 .

Since that kind of data is accessible and extractable, if that kind of specific distinct types of genres are not a coincidence but a pattern, then we can use

latent-factor analysis and further improve RECO with another algorithm.

In addition, new technologies can further improve accuracy and execution of RECO too. Adding or replacing new technologies with the existing ones can be a future work too.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994.

[3] A. M. Ahmad Wasfi. Collecting user access patterns for building user profiles and collaborative filtering. In *Proceedings of the 4th international conference on Intelligent user interfaces*, pages 57–64. ACM, 1998.

[4] X. Amatriain. Mining large streams of user data for personalized recommendations. *ACM SIGKDD Explorations Newsletter*, 14(2):37–48, 2013.

[5] Apache. Apache hadoop, 2014. [Online; accessed 23-January-2014].

[6] Apache. Apache hbase, 2014. [Online; accessed 23-January-2014].

[7] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[8] M. Balabanović. An adaptive web page recommendation service. In *Proceedings of the first international conference on Autonomous agents*, pages 378–385. ACM, 1997.

[9] M. Balabanović. Exploring versus exploiting when learning user models for text recommendation. *User Modeling and User-Adapted Interaction*, 8(1-2):71–102, 1998.

[10] C. Basu, H. Hirsh, W. Cohen, et al. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.

[11] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[12] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann. World-wide web: the information universe. *Internet Research*, 2(1):52–58, 1992.

[13] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *ICML*, volume 98, pages 46–54, 1998.

[14] D. Billsus and M. J. Pazzani. User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2-3):147–180, 2000.

[15] A. Bouza, G. Reif, A. Bernstein, and H. Gall. Semtree: Ontology-based decision tree algorithm for recommender systems. In *International Semantic Web Conference (Posters & Demos)*. Citeseer, 2008.

[16] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[17] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

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

[19] R. Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

[20] R. D. Burke, K. J. Hammond, and B. Yound. The findme approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.

[21] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.

[22] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer, 1999.

[23] M. K. Condliff, D. D. Lewis, D. Madigan, and C. Posse. Bayesian mixed-effects models for recommender systems. In *Proc. ACM SIGIR*, volume 99. Citeseer, 1999.

[24] M. Eirinaki, C. Lampos, S. Paulakis, and M. Vazirgiannis. Web personalization integrating content semantics and navigational patterns. In *Proceedings of the 6th annual ACM international workshop on Web information and data management*, pages 72–79. ACM, 2004.

[25] A. Felfernig and R. Burke. Constraint-based recommender systems: technologies and research issues. In *Proceedings of the 10th international conference on Electronic commerce*, page 3. ACM, 2008.

[26] A. Felfernig, K. Isak, K. Szabo, and P. Zachar. The vita financial services sales support environment. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 22, page 1692. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[27] R. Ghani and A. Fano. Building recommender systems using a knowledge base of product semantics. In *Proceedings of the Workshop on Recommendation and Personalization in ECommerce at the 2nd International Conference on Adaptive Hypermedia and Adaptive Web based Systems*, pages 27–29, 2002.

[28] M. Grčar, D. Mladenič, B. Fortuna, and M. Grobelnik. *Data sparsity issues in the collaborative filtering framework*. Springer, 2006.

[29] J. Han, E. Haihong, G. Le, and J. Du. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pages 363–366. IEEE, 2011.

[30] J. Hannon, M. Bennett, and B. Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 199–206. ACM, 2010.

[31] F. Holzschuher and R. Peinl. Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 195–204. ACM, 2013.

[32] S. H. Hsu, M.-H. Wen, H.-C. Lin, C.-C. Lee, and C.-H. Lee. Aimed-a personalized tv recommendation system. In *Interactive TV: a Shared Experience*, pages 166–174. Springer, 2007.

[33] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[34] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[35] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pages 256–261. IEEE, 1989.

[36] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI*, pages 187–192, 2002.

[37] B. Mobasher. Data mining for web personalization. In *The Adaptive Web*, pages 90–135. Springer, 2007.

[38] Myrrix. Myrrix, 2014. [Online; accessed 23-January-2014].

[39] A. Oghina, M. Breuss, M. Tsagkias, and M. de Rijke. Predicting imdb movie ratings using social media. In *Advances in Information Retrieval*, pages 503–507. Springer, 2012.

[40] O. Osmanli and I. Toroslu. Using tag similarity in svd-based recommendation systems. In *Application of Information and Communication Technologies (AICT), 2011 5th International Conference on*, pages 1–4. IEEE, 2011.

[41] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[42] Pandas. Pandas, 2014. [Online; accessed 23-January-2014].

[43] M. Pastorelli. Scalding programming model for hadoop, 2014. [Online; accessed 23-January-2014].

[44] M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

[45] M. J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[46] B. Ribeiro-Neto, M. Cristo, P. B. Golgher, and E. Silva de Moura. Impedance coupling in content-targeted advertising. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 496–503. ACM, 2005.

[47] J. J. Rocchio. Relevance feedback in information retrieval. 1971.

[48] G. Salton. Automatic text processing. *Science*, 168(3929):335–343, 1970.

[49] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 345–354. ACM, 1998.

[50] M. Sattari, M. Manguoglu, I. H. Toroslu, P. Symeonidis, P. Senkul, and Y. Manolopoulos. Geo-activity recommendations by using improved feature combination. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 996–1003. ACM, 2012.

[51] G. Shani, R. I. Brafman, and D. Heckerman. An mdp-based recommender system. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 453–460. Morgan Kaufmann Publishers Inc., 2002.

[52] B. Smyth and P. Cotter. A personalised tv listings service for the digital tv age. *Knowledge-Based Systems*, 13(2):53–59, 2000.

[53] S. Spiegel, J. Kunegis, and F. Li. Hydra: a hybrid recommender system [cross-linked rating and content information]. In *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 75–80. ACM, 2009.

[54] C. J. Tauro, S. Aravindh, and A. Shreeharsha. Comparative study of the new generation, agile, scalable, high performance nosql databases. *International Journal of Computer Applications (0975–888) Volume*, pages 7461–0336, 2012.

[55] R. C. Taylor. An overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics. *BMC bioinformatics*, 11(Suppl 12):S1, 2010.

[56] T. Tran and R. Cohen. Hybrid recommender systems for electronic commerce. In *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press*, 2000.

[57] S. Trewin. Knowledge-based recommender systems. *Encyclopedia of library and information science*, 69(Supplement 32):69, 2000.

[58] B. G. Tudorica and C. Bucur. A comparison between several nosql databases with comments and notes. In *Roedunet International Conference (RoEduNet), 2011 10th*, pages 1–5. IEEE, 2011.

[59] Twitter. Scalding, 2014. [Online; accessed 23-January-2014].

[60] A. Wang et al. An industrial strength audio search algorithm. In *ISMIR*, pages 7–13, 2003.

[61] Wikipedia. Google news — wikipedia, the free encyclopedia, 2014. [Online; accessed 23-January-2014].