T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

NOVEL RECOMMENDER SYSTEM IMPLEMENTATION BY USING COLLABORATIVE FILTERING

Master Thesis

ERGİN DEMİREL

İSTANBUL, 2009

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

Institute of Science
Computer Engineering Graduate Program

NOVEL RECOMMENDER SYSTEM IMPLEMENTATION BY USING COLLABORATIVE FILTERING

Master Thesis

Ergin DEMİREL

SUPERVISOR: ASSOC. PROF.DR. ADEM KARAHOCA

İSTANBUL, 2009

T.C

BAHÇEŞEHİR ÜNİVERSİTESİ

Graduate School in Sciences

Computer Engineering Graduate Program

Name of the thesis: **NOVEL RECOMMENDER SYSTEM IMPLEMENTATION BY USING COLLABORATIVE FILTERING**

Name/Last Name of the Student: Ergin DEMİREL

Date of Thesis Defense: 22/01/2009

The thesis has been approved by the Institute of Graduate School in Sciences.

Prof. Dr. A. Bülent Özgüler

Director

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. A. Bülent Özgüler

Program Coordinator

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Adem Karahoca

Supervisor

Examining Committee Members	Signature
Assoc. Prof. Dr. Adem Karahoca	
Prof.Dr. Nizamettin Aydın	
Asst. Prof. Dr. Yalçın Çekiç	

ACKNOWLEDGEMENTS

So many people have helped me with this thesis, that it would be impossible to name them all.

Firstly, I would like to thank my co-workers especially Dr. Ömer Artun and Dhruv Bhargava who helped me to introduce many concepts that is used in this thesis. Their helps are greatly appreciated. Also I would like to thank all other co-workers who have helped me out to achieve my goals at the experimental part of this thesis.

I would like to thank my supervisor Assoc. Prof. Dr. Adem Karahoca who was daring enough to accept me as graduate student. He was always very helpful about this thesis and my works whether they are positive or negative. He was always available when I needed help or advice, without his tolerance and support completion of this study wouldn't be possible. I can never repay him for the opportunity he gave me.

Finally, I would like to express my gratitude to my family (Ergün, Sevgi and Özbay Demirel) and friends. Although they do not understand what is written in this thesis, their love and support were essential for the completion of my thesis. It is to them I would like to dedicate this thesis.

ABSTRACT

NOVEL RECOMMENDER SYSTEM IMPLEMENTATION
BY USING COLLABORATIVE FILTERING

Ergin DEMİREL

The Institute of Sciences, Computer Engineering Graduate Program

Supervisor: Assoc. Prof. Dr. Adem Karahoca

January 2009, 62 pages

Recommender systems apply data mining techniques to solve the problem of making product recommendations. These systems, especially Collaborative Filtering (CF) based ones, are achieving widespread success in E-commerce nowadays. Key challenge in the E-Commerce is creating high quality recommendations to increase profit of the company. In order to solve the problem of producing high quality recommendations, new recommender system algorithms are needed that can quickly produce high quality recommendations, even for very large-scale databases.

Aim of this thesis is to introduce a new recommender system algorithm that is based on item-to-item collaborative filtering and analyzing the performance of the newly introduced recommender system by comparing it with the item-to-item clicked stream based recommender system that is currently in use. During our performance analysis we were concentrated on problem of producing high quality recommendations for the existing system rather than increasing response time of the system. We used several performance measures/metrics such as Receiver Operating Characteristic (ROC), Mean Square Error, Mean Absolute Error, Root Mean Squared Error, Hit Rate and other performance measures introduced by ourselves to compare quality of the recommendations made by each system.

According to our findings, we concluded that our algorithm gives better results than the existing recommender system for different sample outcome size N=313 and N=538. However for some of the performance measures such as statistical accuracy metrics and AUC, we saw that existing recommender system (item-item clicked stream based) gave better results than ours.

Implementation of our algorithm might not give expected results for different purposes and different contexts because of the algorithm given in this thesis aimed to produce high quality recommendation for specific business.

Key words: Recommender Systems (RS), Recommendation Engine, Collaborative Filtering (CF), Item-Based Collaborative Filtering, Content-Based Collaborative Filtering, Item-to-Item Recommender Systems, Similar Item Analysis

ÖZET

ORTAKLAŞA ENFORMASYON PAYLAŞMA KULLANARAK
YENI TAVSİYE SİSTEMİ UYGULAMASI

Ergin DEMİREL

Fen Bilimleri Enstitüsi, Bilgisayar Mühendisliği Yüksek Lisan Programı

Tez Danışmanı: Doç. Dr. Adem Karahoca

Ocak 2009, 62 Sayfa

Tavsiye sistemleri ürün tavsiyesi problemini çözmek için veri madenciliği tekniklerini kullanır. Günümüzde bu sistemler özellikle Ortaklaşa Enformasyon Paylaşma'ya (Collaborative Filtering) dayalı olanlar elektronik ticarette büyük oranda başarıya ulaşmışlardır. Elektronik ticaretteki en önemli alanlardan biri yüksek kaliteli tavsiyler yaratarak firmanın kazancının arttırılmasıdır. Yüksek kalitede tavsiye üretme sorununu çözmek için kısa sürede hatta çok büyük yapılı veri tabanları için yeni tavsiye sistemlerine (recommender systems) ihtiyaç vardır.

Bu tezin amacı üründen ürüne ortaklaşa paylaşmaya (item-to-item collaborative filtering) dayalı yeni bir tavsiye sistemini tanıtmak ve bu yeni sisteminin performansını şu an kullanilmakta olan üründen ürüne tıklamaya dayalı tavsiye sistemi (item-to-item clicked stream) ile karşılaştırıp analiz etmektir. Performans analizi sürecinde sistemin cevap verme süresinden çok, yüksek kalitede tavsiye üretme problemine yönelinmiştir. Her bir sistemin tavsiyelerinin kalitesini ölçmek için Receiver Operating Characteristic (ROC), Mean Square Error, Mean Absolute Error, Root Mean Squared Error, Hit Rate ve bizim tarafımızdan sunulan başka performans ölçütlerini kullandık.

Elimizde bulgulara dayalı olarak, bizim algoritmamızın örnek çıktı miktarı N=313 ve N=539 için mevcut sisteme göre daha iyi sonuçlar verdiğine karar verdik. Bunun yanında istatistiksel doğruluk ölçütleri ve AUC gibi performans ölçütlerinde mevcut sistemin (item-item clicked stream dayali) bizimkinden daha iyi sonuclar verdigini gördük.

Farklı amaçlar ve içeriklerde bizim algoritmamız beklenen sonuçları veremeyebilir çünkü bu tezde verilen algoritma işe özel yüksek kalitede tavisyeler üretmeyi amaçlamıştır.

Anahtar Kelimeler: Recommender Systems (RS), Recommendation Engine, Collaborative Filtering (CF), Item-Based Collaborative Filtering, Content-Based Collaborative Filtering, Item-to-Item Recommender Systems, Similar Item Analysis

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Recommender System	:	RS
Recommendation Engine	:	RE
Collaborative Filtering	:	CF
Content-Based Filtering	:	CBF
Receiver Operating Characteristic	:	ROC
Mean Square Error	:	MSE
Mean Absolute Error	:	MAE
Root Mean Squared Error	:	RMSE
Hit Rate	:	HR
Click stream-based Collaborative Filtering	:	CCF
Data Mining	:	DM
Aspect Model	:	AM
Personality Diagnosis Model	:	PDM
Association Rule	:	AR
Markov Model	:	MM
Area Under the Curve	:	AUC
Opportunity Success Rate	:	OSR
Key Performance Indicator	:	KPI

1-INTRODUCTION

Recommender systems (RS) apply data mining (knowledge based) and/or statistical techniques to the problem of making product recommendation and they are achieving widespread success in E-commerce nowadays. As you all well known, Amazon.com, Netflix and Last.fm can be shown as successful implementation of recommender systems (Figrue1.1) in different areas. At the tremendous growth in the amount of available information and the number of customers in the databases pose some keys challenges for recommender systems (Vozalis and Margaritis, 2003). First challenge is improve the scalability of Collaborative Filtering (CF) algorithms. Most successful recommender technologies are collaborative filtering (Breese et al, 1998, Kitts et al 2000, Chan 1999, Shardanand and Maes 1995). These algorithms are able to search tens of thousands of records in short time of period with the current technology. But the most of the E-commerce systems search millions of records. Some of the recommender systems nowadays create recommendation based on user preferences in-real time. Users of your system probably won't wait for long time to receive recommendation, if your algorithm response them in long time intervals. Therefore, response time of the algorithm is the one of the important performance indicator. Another challenge is to improve the quality of recommendations for potential customers. Customers usually need recommendations that they can trust or in other word they likely to buy. If the outputs of your recommender system have mostly poor recommendations then the consumers will be unlikely to use recommender system which is directly inversely proportional to the increase in your sales coming from your e-commerce site. Recommender systems other than other search systems have two types of error: false negatives which products are not recommended though the consumer would like them and false positives which product are recommended though the consumer wouldn't like them. In the E-commerce, the most important errors to avoid are false positives (Sarwar et al 2000).

Nowadays, it's more important to design recommendation engines that can overcome those challenges. There are two ways to design recommendation system: Content Based (CB) and Collaborative Filtering (CF) Systems. CF systems recommend products to target customers based on opinions of other customers.

These systems employ statistical techniques to find a set of customers known as neighbors who either rate different product similarly or they tend to buy similar set of products (Sarwar et al, 2000). In this thesis, we are focusing on Collaborative Filtering method to design a new recommender system algorithm.

Algorithm simply uses statistical methods to calculate similarity between products by use of historical data of customers.

Figure 1.1: Famous recommender systems' implementations

All these calculations are done offline and the recommendations are served to customer when they are viewing or purchasing any product in the system. Recommendations are expressed as a list of N item(s) where N is less than number of items in the system and items which customers most likely to buy. Also recommended items must be items where customer hasn't purchased yet or item that not exist in the user's basket (cart).

New recommender system algorithms are needed to solve either scalability problems or quality of recommendation problems (or both of them). To overcome this kind of problems, comparison between the existing and the new algorithms must be done. Two questions that needs to be asked are how such comparison/validation can be accomplished and what your expectations from a new recommender system are. Before we started to design a new recommender system, we asked our customers about their expectations.

They didn't interest much of scalability problem even not even mention it. According to our findings, only thing that our customers concern is to increase the web based sales and this can be accomplished, if you are able recommend product that customer interested in.

We designed our algorithm as an alternative to the click stream based recommendation algorithm. Click stream based CF algorithms are collects click stream information of user where the click streams are defined as users' paths through a web site. Analysis of click streams shows how a web site is navigated and used by its visitors. Click stream data of online stores contains information useful for understanding the effectiveness of marketing and merchandising efforts, such as how customers find the store, what products they see, and what products they purchase. Recommender system can be designed by use of click streams and knowledge based algorithms like the Markov model, sequential association rules, association rules, multiple association rules and clustering.

During our study, we didn't know how recommendations are made by the system being compared. We evaluated it as a black box solution. Therefore, we needed to define performance measures to compare two different systems where we don't much about the algorithm behind the one of the systems. However, we had the historical recommendations made by this system for the same context. We researched the previously introduced evaluation metrics for recommender systems and decided to use metrics which we could apply to our problem. We used some statistical methods such as Receiver Operating Characteristics (ROC), Mean Square Error (MSE), Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Opportunity Success Rate (OSR), Hit Rate (HR), correctness and precision (defined in information retrieval). Also we asked business owner to record growth in the sales after using our recommendation engine. This was the one another key performance indicator (KPI) for us. While we were comparing results, we worked with real world data that is specific to business.

This thesis is organized as six sections. In next section, we define recommender systems; explain type of recommender systems and all the data mining techniques in details mentioned above. One of our objectives is to introduce new data mining algorithm to create a recommender system as an alternative to existing recommender system for business specific e-commerce site.

We explain our algorithm in details in section 3. Section 4 explains metrics used in experiments. Section 5 presents experimental results. In this section, we provide dataset used in analysis and results of benchmarking for different samples and tests. Finally, conclusion and future works are provided in section 6.

2- RESEARCH METHOD

2.1 DATA MINING AND RECOMMENDER SYSTEMS

The amount of operational data collected and stored by organizations is steadily increasing. Adriaans and Zantinge (2006) state that the amount of information stored as data form in the world doubles roughly every 20 months. But at the same time, organizations and individuals continue to be starved for knowledge. The key is to seek hidden nuggets of valuable, strategic information in this data thus converting it into knowledge. Data mining and data analytic tools allow organizations to probe into the mass of collected data and effectively uncover these nuggets of new information which can aid strategic decisions as well as provide feedback critical to monitoring business performance. Data mining accomplishes this task by applying statistical algorithms to data which assist analysts in finding meaningful patterns and relationships. These patterns and relationships in turn can be used to both increase revenues and reduce costs for organizations (Berry & Linoff 2000).

One of the most popular implementation of the data mining tasks is recommender systems (RS). Recommender systems use the opinions of a community of users to help individuals in that community more effectively identify content of interest from a potentially overwhelming set of choices (Resnick & Varian 1997). We can categorize recommender systems in two main categories:

1. **Direct recommender systems:** the user interacts directly with the system that helps him search for the item in a list with the n-articles that most likely with his request in relation to his profile. This provides direct marketing.
2. **Pervasive recommender systems:** this system works as backup to the system of marketing. The advertising system fills in the content over the full page in a personalized way. This provides a method to cross-sell marketing.

In order to create recommendations some reference characteristics are collected about item or user and compared with the others. These characteristics may be from the information item (content-based approach) or the user's social environment (CF approach).

In both approaches, data (information) collection is the essential part of the RS. Data can be collected explicitly or implicitly.

Explicit data collection may include the following:

1. Asking a user to rate an item on a sliding scale.
2. Asking a user to rank a collection of items from favorite to least favorite.
3. Presenting two items to a user and asking him/her to choose the best one.
4. Asking a user to create a list of items that he/she likes.

Explicit data collection requires more user interaction. Because of that collecting information is generally hard. This type of data collection may result better recommendations since these type of recommendations are based on users preferences.

Implicit data collection may include the following:

1. Observing the items that a user views in an online store.
2. Analyzing item/user viewing times.
3. Keeping a record of the items that a user purchases online (users transactions).
4. Obtaining a list of items that a user interested in.
5. Analyzing the user's social network.

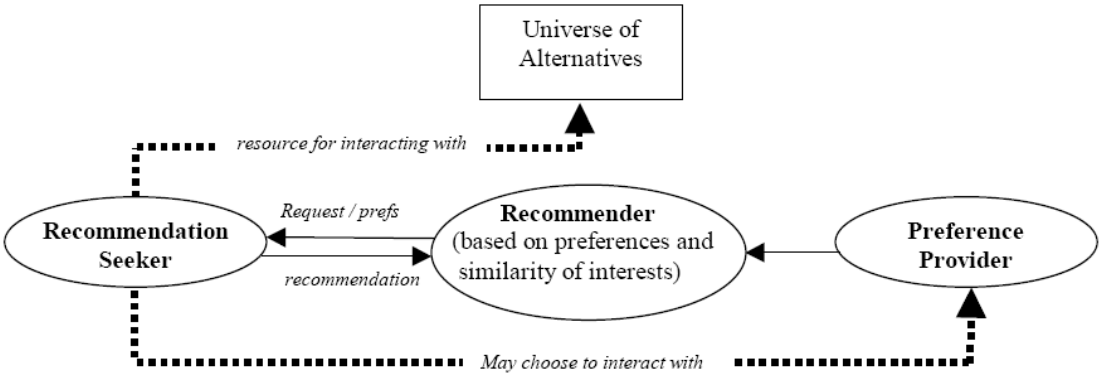

Figure 2.1: General Model for the Recommender Systems (Terveen & Hill, 2001)

Implicit data collection is the most popular way of collecting information since it is much easier to implement. But it may hard to create more personalized recommendations. In either ways, the recommender system compares the collected data to similar data collected from others and calculates a list of recommended items by use of statistics and DM techniques. DM techniques used in this thesis are using both methodologies for data collections. The system that we introduce in this thesis concentrates on cross-selling of products and implicit data collection.

2.2 DATA MINING METHODS USED IN RECOMMENDER SYSTEMS

Today recommendation systems have been used in many fields; virtually all topics that could be of potential interest to users are covered by special purpose recommendation systems: Web pages, news, emails, social networks, music, books, movies and many more. These recommendation systems predict the users' interest and preference based on all users' profiles, using information retrieval techniques.

Many different algorithmic approaches have been applied to the basic problem of making recommendations. The underlying techniques used in today's recommendation systems fall into two distinct categories: content-based filtering (CBF) and collaborative filtering (CF) methods. The CBF uses actual content features of items, while the CF predict new user's preference using other users' rating, assuming the like-minded people tend to have similar choices (Sarwar et al,2001 , Sarwar et al 2002).

The earliest recommender systems were CBF systems designed to fight information overload in textual domains. These were often based on traditional information-filtering and information-retrieval systems. CF's systems provide generalized recommendations by aggregating the evaluations of the community at large. More personalized systems (Resnick & Varian, 1997) employ techniques such as user-to-user correlations or a nearest-neighbor algorithm.

The application of user-to-user correlations derives from statistics, where correlations between variables are used to measure the usefulness of a model. In recommender systems correlations are used to measure the extent of agreement between two users (Breese, Heckerman & Kadie , 1998) and used to identify users whose ratings will contain high predictive value for a given user.

Nearest-neighbor algorithms compute the distance between users based on their preference history. Distances vary greatly based on domain, number of users, number of recommended items, and degree of co-rating between users.

Predictions of how much a user will like an item are computed by taking the weighted average of the opinions of a set of neighbors for that item. As applied in recommender systems, neighbors are often generated online on a query-by-query basis rather than through the offline construction of a more thorough model.

Both nearest-neighbor and correlation-based recommenders provide a high level of personalization in their recommendations, and most early systems using these techniques showed promising accuracy rates.

As such, CF-based systems have continued to be popular in recommender applications and have provided the benchmarks upon which more recent applications have been compared.

Following section discusses the CF algorithms in details and identifies challenges and opportunities for each type. Each CF algorithm is categorized according the recommendation process they are using.

2.3 COLLABORATIVE FILTERING BASED RECOMMENDER SYSTEMS

Recommender systems apply data analysis techniques to the problem of helping users to find the items they would like to purchase by producing likelihood (prediction) score or a list of Top-N recommended items for a given user. Recommendation can be done using different approaches. It can be done using demographic of users, overall top selling items or past buying habits of user (Sarwar et al 2001).

The basic idea of Collaborative Filtering based recommendation is to provide item recommendation based on other users opinions who are likeminded to given user.

2.3.1 COLLABORATIVE FILTERING PROCESS

The goal of a collaborative filtering algorithm is to suggest new items or to predict the utility of a certain item for a particular user based on the user's previous likings and the opinions of other like-minded users. In a typical CF scenario, there is a list of m user $U = \{u_1, u_2, \ldots, u_m\}$ and a list of n items. $I=\{i_1, i_2, \ldots, i_n\}$ Each user u_i has a list of items I_{ui}, which the user has expressed his/her opinions about. As we discussed before opinions can be explicitly given by the user as a rating score, generally within a certain numerical scale, or can be implicitly derived from purchase records, by analyzing timing logs, by mining web hyperlinks and etc. Note that $I_{ui} \subseteq I$ and it is possible for I_{ui} to be a *null-set*. There exists a distinguished user $U_a \in U$ called the *active user* for whom the task of a collaborative filtering algorithm is to find an item likeliness that can be of two forms (Sarwar et al, 2001).

1. **Prediction** is a numerical value, $P_{a,j}$, expressing the predicted likeliness of item $I_j \notin I_{ua}$ for the active user u_a. This predicted value is within the same scale (e.g., from 1 to 5) as the opinion values provided by u_a.

2. **Recommendation** is a list of N items, $I_T \subseteq I$, that the active user will like the most. Note that the recommended list must be on items not already purchased by the active user, i.e: $I_T \cap I_{ua}$. This interface of CF algorithms is also known as *Top-N recommendation*.

Table 2.1: Sample transactions for five customers

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	

A basic example to describe CF process can be given as follows: assume rows of transactions for few customers shown as Table 1, and we would like to make recommendation for *Customer C*; in this case *Customer B* is the most similar customer to the *Customer C* because of the number of items in common.

Item 5 can be recommended to the *Customer C* with high probability. *Customer D* is somewhat similar to the *Customer C*. Therefore *Item 6* can be recommended with low probability respect to the *Item 5*. *Customer A* and *Customer E* have no item in common with *Customer C*. Similarity for these customers are 0.

Researchers have devised a number of CF algorithms that can be divided into two main categories. User Based (Memory Based) and Model Based (Item based) algorithms (Breese et al 1998).

2.3.2 MEMORY BASED COLLABORATIVE FILTERING ALGORITHMS

Memory based CF algorithms utilize the entire user-item database to generate recommendation. These systems employ statistical techniques to find a set of users (called neighbors). Once a neighborhood of users is formed, these systems use different algorithms to combine the preferences of neighbors to produce the prediction or Top-N recommendation for a given user (Sarwar et al, 2001). This technique is also known as nearest neighbor or user-based CF.

All users' preferences could be represented by their votes (or ratings) to the products. The new user has an average vote over the products he/she has rated. Then the predicted votes of the new users over other products could be calculated by adding weighted sum of other users' votes. The weights could be determined by the similarity between the new user and other users. The more similar they are, the more contributions they have to the sum, so the large the weights are. The user's average vote could be represented as below, the I_i is the set of items the new user i has voted, V_{ij} is the user i vote to product j.
The average vote is formulated:

$$\overline{v_i} = \frac{1}{|I_i|} \sum_{j \in I_i} V_{i,j}$$

And the predicted vote of the active user is:

$$p_{a,j} = \overline{V_a} + k \sum_{i=1}^{n} w(a,i)\left(v_{i,j} - \overline{V_i}\right)$$

Where the k is a normalizing factor, while w(a,i) is the weight that the user i contributes to the active user. The weights are calculated by comparing a set of common products, which the active user and all other users in the database have rated. Weights can be defined by using different techniques such as Pearson correlation, Vector similarity and Mean Square difference.

Traditional method for memory based CF algorithm searches the whole database. Apparently this method suffers from poor scalability when more new users and new items are added into the database. By using the concept of a clustering eliminates the drawback of the memory based collaborative filtering. From the process speed point of view, clustering can help to speed up the computation of similarity calculation as well as remove some irrelevant information.

As opposed to the poor scalability problem of the memory based algorithms Breese (1998) describes and evaluates two probabilistic models, which they term the Bayesian clustering and Bayesian network models. In the first model, likeminded users are clustered together into classes. Given his or her class membership, a user's ratings are assumed to be independent (i.e., the model structure of a Naive Bayesian network). The number of classes and the parameters of the model are learned from the data. The second model also employs a Bayesian network, but of a different form. Variables in the network are titles and their values are the allowable ratings. Both the structure of the network, which encodes the dependencies between titles, and the conditional probabilities are learned from the data. Ungar and Foster (1998) also suggest clustering as a natural preprocessing step for CF. Both users and titles are classified into groups; for each category of users, the probability that they like each category of titles is estimated.

2.3.3 MODEL BASED COLLABORATIVE FILTERING ALGORITHMS

Model Based CF algorithms provide item recommendation by first developing a model of user ratings or purchases. Algorithms in this category take a probabilistic approach and envision the CF process as computing expected value of a user prediction, given his/her ratings/interests on other items (Sarwar et al 2001, Pennock and Horvitz, 2000).

The model building process is performed by different machine learning algorithms such as Bayesian network, clustering and rule based approaches. The Bayesian network model formulates a probabilistic model for CF problem (Breese et al 1998).

Clustering model treats CF as a classification problem and works by clustering similar users in same class and estimating the probability that particular user is in particular class, and from there conditional probability of ratings. The rule-based approach applies association rule discovery algorithms to find association between co-purchased items and then generates item recommendation based on strength of the association between items.

Two popular model-based algorithms are the Aspect Model (AM) (Claypool et al 1999 & Hofmann et al 2003) and the Personality Diagnosis Model (PDM) (Pennock et al 2000).

AM is a probabilistic latent space model, which models individual preferences as a convex combination of preference factors (Claypool et al 1999 & Hofmann et al 2003). The latent class variable $z \in Z = \{z_1, z_2, ..., z_k\}$ is associated with each observed pair of a user and an item. The aspect model assumes that users and items are independent from each other given the latent class variable. Thus, the probability for each observation tuple (x, y, r) is calculated as follows:

$$p(r \mid x, y) = \sum_{z \in Z} p(r \mid z, x) p(z \mid y)$$

In this equation $p(z \mid y)$ stands for the likelihood for the user y to be the class z and $p(r \mid z, x)$ stands for the likelihood of assigning the item x with the rating r by the class z of users. The ratings of each user are normalized to be norm distribution with zero mean and one variance. A Gaussian distribution is used for the parameter $p(r \mid z, x)$ and a multinomial distribution for $p(z \mid y)$ (Hofmann et al 2003).

Personality diagnosis approach treats each user in the training database as an individual model. To predicate the rating of an item by a test user, this approach first computes the likelihood for the test user to be in the model of each training user and then uses the aggregate average of ratings for the item by the training users as the estimator.

By assuming that the observed rating of the test user y_t on an item x is drawn from an independent normal distribution with the mean as the true rating $\sum_{yt}^{True}(x)$, we have

$$p\left(R_{y_t}(x) \mid R_{y_t}^{True}(x)\right) \propto e^{-(R_{y_t}(x)-R_{y_t}^{True}(x))^2}/2\sigma^2$$

Then, the probability for the test user y_t to be in the model of any user y in the training database can be written as:

$$p(y_t \mid y) \propto \prod_{x \epsilon X(y_t)} e^{-(R_y(x)-R_{y_t}^{True}(x))^2/2\sigma^2}$$

Previous empirical studies have shown that the PD method is able to outperform several other approaches for collaborative filtering (Pennock et al 2000), including the PCC method, VS method and the Bayesian network approach.

$$p\left(R_{y_t}(x) = r\right) \propto \sum_{y} p(R_{y_t} \mid R_y)\, e^{-(R_2(x)-r)^2/2\sigma^2}$$

2.4 ITEM BASED COLLABORATIVE FILTERING ALGORITHMS

Here in this section, we describe the item based algorithms to recommend items for users. Unlike the user based algorithms explained above, item-based approach calculated similarities between items based on users ratings or purchases and recommend similar *top N* item for target item *i*. There are several ways to compute similarity between items. Here we introduce four well know methods: cosine based similarity, correlation based similarity, adjusted cosine based similarity and Conditional Probability-Based Similarity. Regardless of the *sim(i,j)* function chosen, item based algorithms compute similarities between items for m x n rating matrix shown below where m represent user and n represent items. Another challenging task for item based CF is prediction computation to obtain recommendations which user would mostly like to buy.

For this purpose, we introduce two well known methods Weighted Sum and Regression (Sarwar et al, 2001).

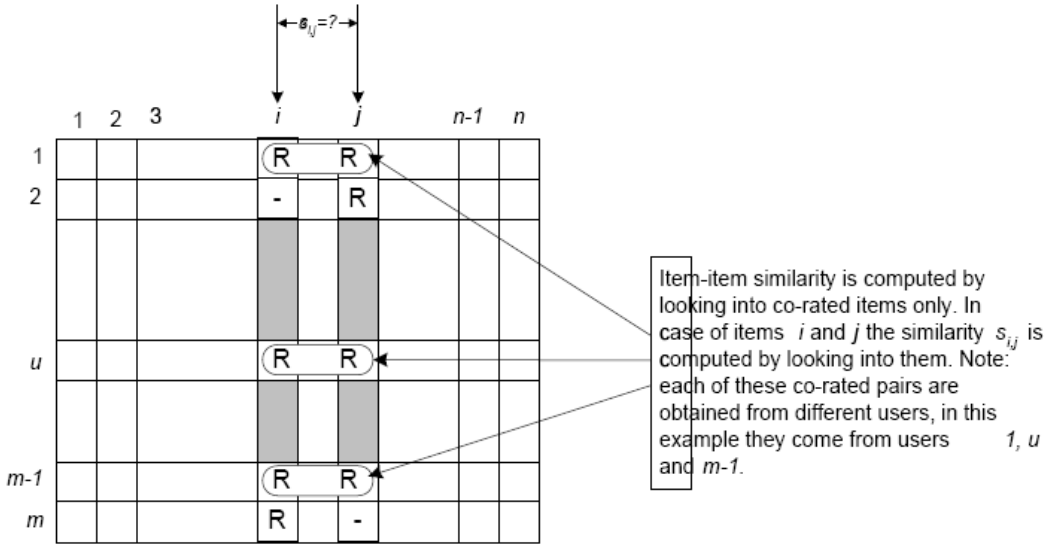

Figure 2.2: similarity computation model for m x n rating matrix (Sarwar et al, 2001).

2.4.1 COSINE BASED SIMILARITY

Two items are thought of as two vectors in the m dimensional user space. The similarity between these items is measured by computing the cosine of the angel between these two vectors (Sarwar et al 2001).

$$sim(x,y) = \cos(\vec{i},\vec{j}) = \frac{\vec{i} \,.\, \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

Where "." denotes dot products of two vectors.

2.4.2 CORRELATION BASED SIMILARITY

Correlation based similarity is computed by Pearson-r Correlation corr(i, j). While calculating correlation based similarity, only co-rated /co-purchased items are picked (Mild And Natter, 2001). Similarity for set of users who rated / purchased the item i and j donated by U and calculated as follows:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}}$$

Where $R_{u,i}$ is the rating of user u on item i and $\bar{R}_i$ is the average rating / purchasing for item i.

2.4.3 ADJUSTED COSINE BASED SIMILARITY

One fundamental difference between the similarity computation in user based CF and item based CF is that in user based CF the similarity is calculated along the rows of the matrix shown figure 2.2 but in item based CF the similarity is calculated along the columns. Cosine base similarity calculation has drawback which the differences in rating / purchasing scale between users are not taken into account. Adjusted cosine similarity is a substitute for this drawback.

So the similarity between items i and j is computed as follows:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}}$$

Here $\bar{R}_u$ is the average of the u-th user's ratings / purchases (Sarwar et al 2001).

2.4.4 CONDITIONAL PROBABILITY BASED SIMILARITY

The similarity between each pair of items i and j is to use a measure that is based on the conditional probability of rating / purchasing one of the items given that the other has already been rated / purchased. In particular, the conditional probability of purchasing j given that i has already been purchased P(j | i) is nothing more than the number of customers that purchase both items i and j divided by the total number of customers that purchased i, that is:

$$P(j \mid i) = \frac{Freq(i,j)}{Freq(i)}$$

where Freq(i) is the number of customers that have purchased the items in the set i. Note that, in general, P(j | i) $\neq$ P(i | j) and using this as a measure of similarity leads to asymmetric relations.

One of the limitations of using an asymmetric similarity function is that each item i will tend to have high conditional probabilities with items that are being purchased frequently. This problem has been recognized by researchers in information retrieval and recommender systems (Breese et al., 1998 & Kitts et al, 2000 & Chan 1999). The problem can be corrected by dividing P(j | i) with a quantity that depends on the occurrence frequency of item j. Deshpande and Karypis (2004) offer two solutions to this problem. They use following formula to compute similarity between two items:

$$sim(i,j) = \frac{Freq(i,j)}{Freq(i)\ X\ (Freq(j))^{\propto}}$$

Where α is a parameter that takes a value between 0 and 1. Note that, when α = 0, becomes identical to $P(j \mid i)$, whereas if α = 1, it becomes similar to the formulation in which $P(j \mid i)$ is divided by $P(j)$. The similarity function as defined in above equation does not discriminate between customers that have purchased different number of items. To achieve this discrimination and give higher weight to the customers that have purchased fewer items, they

have extended the similarity measure of above equation in the following way: First normalize the each row of matrix *R* to be of unit length.

The similarity between items *i* and *j* as:

$$sim(i,j) = \frac{\sum_{\vee q:R_{q.j}>0} R_{q.j}}{Freq(i)\ X\ (Freq(j))^{\propto}}$$

The only difference between third and second equation is that instead of using the co-occurrence frequency, the sum of the corresponding nonzero entries of the j-th column in the user–item matrix is used. Since the rows are normalized to be of unit length, customers that have purchased more items will tend to contribute less to the overall similarity. This gives emphasis to the purchasing decisions of the customers that have bought fewer items.

2.4.5 PREDICTION COMPUTATION

Prediction is a numerical value, pr_{aj}, which represents the predicted opinion of specific user u_a about item i_j. It is a necessary condition that item i_j does not belong in the set of items for which the active user has expressed his opinion about.

2.4.6 WEIGHTED SUM BASED PREDICTION COMPUTATION

This method generates a prediction for item i_j for active user u_a by computing the sum of ratings/purchases given by the specific user on items belonging to the neighborhood of i_j. Those ratings/purchases are weighted by the corresponding similarity, sim_{jk}, between item i_j and item i_k, with $k = \{1, 2, ..., l\}$, taken from neighborhood N, weighted sum is calculated as follows:

$$pr_{aj} = \frac{\sum_{k=1}^{l} sim_{jk} * r_{ak}}{\sum_{k=1}^{l} |sim_{ak}|}$$

2.4.7 REGRESSION BASED PREDICTION COMPUTATION

This is a similar method to the weighted sum but instead of directly using the ratings/purchases of similar items it uses an approximation of the ratings based on regression model. If an active user voted for items from I_a, there are $|I_a|$ available experts for predicting the ratings of each item from$\frac{I}{I_a}$. The recommendation problem can now be approached as the identification of an optimal combination of experts. To make the solution computationally efficient, Vucetic and Obradovic (2005), model the experts $f_{j,i}$ as linear functions:

$$F_{j,i}(x) = X\alpha_{j,i} + \beta_{j,i}$$

Where $\alpha_{j,i}$ (eq-1)and $\beta_{j,i}$ (eq-2) are the only two parameters to be estimated for each expert. The two parameters could be estimated by using ordinary least squares as follow:

$$\propto_{j,i} = \frac{\sum_{u \in U_i \cap U_j}(r_{ui} - r_{*i})(r_{uj} - r_{*j})}{\sum_{u \in U_i \cap U_j}(r_{ui} - r_{*i})^2} \text{ (Eq-1)}$$

$$\beta_{j,i} = r_{*i} - \alpha_{j,i} r_{*j} \text{ (Eq-2)}$$

2.5 CLICKSTREAM BASED RECOMMENDATION ALGORITHMS

We studied click stream-based CF (CCF) algorithms to avoid pitfalls they fall and tried to design better algorithm to make better quality recommendations for specific dataset. We are not going to give details of each algorithm since the algorithms used in this section far from scope of this thesis. However we try to summarize each well known method's weakness and strength over the others. CCF recommendation has many applications. Some examples of the applications are: customizing web site interfaces by predicting next relevant pages, documents, page categories, (such as iGoogle) or products (Intelligent Offers[tm]). CCF a kind of item-based CF algorithm that adopts prediction models for efficient and effective recommendation of items: it trains the models offline and uses them in online recommendation (Deshpande & Karypis, 2001).

Unlike in other item-based CF recommendations the sequence of items viewed, user session time that spent in the page and repeat visits are important in CCF for its recommendation quality. The common prediction models for CCF recommendation are the Markov model, Association rule and clustering (Kim et al, 2004).

The Markov Model (MM) has been well positioned as a prediction model in CCF because of its high precision. However, this high precision comes at the cost of low recall because the click stream data is usually sparse and MM cannot cover the sparse data well. MM generally has higher precision and lower coverage than association rule and clustering.

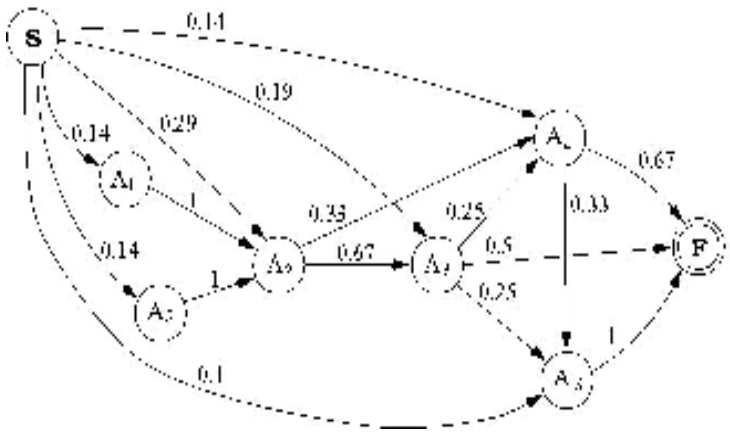

Figure 2.3: Markov Chain data, each node represents pages/items. Probabilities between Start (S) and Final (F) are obtained from counting click-troughs.

Association Rule (AR) is based on the relationship of co-occurrences of pages / items viewed, without considering the sequence of them. AR has been applied for finding frequent product that is used for recommending relevant products. This makes AR generally produce higher coverage than MM at the expense of lower precision in CCF recommendation.

Clustering may have a performance issue because its recommendation process may require the real-time calculation of finding the closest page in the closest cluster for an active click stream. So its online operation could be relatively expensive compared to the MM and AR whose online real-time operation is just to refer to lookup tables that are built offline.

Clustering shows higher coverage than other models because it can cover a current path with the closest cluster (Kim et al, 2004).

2.6 LIMITATIONS OF COLLABORATIVE FILTERING ALGORITHMS

There are several technical challenges for collaborative filtering algorithms. One of the most important problems is data sparsity (Balabanovic & Shoham, 1997). We can describe the data sparsity as if the number of people who have rated items is relatively small compared to the number of item set in the database, there won't be significant similarity between users and items. This means that neighbors really won't be enough near to describe user or recommendations, thus recommendations won't be all that good. These problems become more urgent as the number of items increases versus number of rates.

Another problem is new users and new items in the system (Good et al, 1999). Since there is no purchase or rating for the new items, creating clusters or similarity metrics for these items are almost impossible. Same issue is valid for new users. Users with no profiles or having few ratings/ranking would not able to get accurate recommendations.

Scalability is the most common problem for the most of the recommendation algorithm. Memory based algorithms are usually use nearest neighborhood algorithms which require computations that grows with both number of users and items. Nowadays e-commerce sites have millions of users and products. Computing user-item matrix for such systems causes serious performance problems and requires large storage areas.

A final important issue concerns the notion of "serendipity". Stated informally, I want a recommender system to "tell me something I don't already know." Many current systems fail this test. For example, we used Amazon.com's recommendation engine for similar items. After we rated a number of books, the system recommended Dan Brown's Deception Point. At this point, the system began to recommend more Dan Brown books, such as Angels & Daemons, Da Vinci Code, and so on. It seems unlikely that someone who is familiar with any of Dan Brown's book will be unaware of the rest of his books. Thus, these recommendations carried no new information. Such situations are common.

An analogous argument can be made for CDs and artists. In fact, the argument can even be strengthened. If someone rates CDs by Nirvana highly, that person is highly likely to already have an opinion about CDs by Hole and Foo Fighters (because of overlap and/or relationships between members of these groups) (Terveen & Hill, 2001).

3- NEW ALGORITHM FOR ITEM-ITEM RECOMMENDATIONS

In this section, we present new item-item based CF algorithm to recommend Top-N item to users which it is called as RE (stands for Recommendation Engine). We designed this algorithm upon request from our clients and implemented for one of our major client who sells home furnishings, contemporary apparel and unique gifts online. Initial design of the algorithm was implemented for contemporary apparel products. During our study, the key measure for our client was to improve the quality of the recommendations to increase the sales coming from recommended items tab. The system, we have applied RE algorithm is able to keep track of sales coming from web site and more specifically sales coming from recommended items tab in the web site. That is actually helped us a lot to analyze performance of our recommendation system. This information will be given in details on the following sections of this thesis.

3.1 RECOMMENDATION ENGINE (RE) ALGORITHM

Our algorithm concentrates on creating Top-N recommendation for items purchased and viewed by users to other similar items rather than matching the user to similar users who have already purchased similar products with given user. We determined the most-similar items for a given item I_i by creating item to item matrix where each corresponding item I_j in this matrix purchased within specific correlation time. We could build an item-to-item matrix by iterating through all item pairs and computing a similarity metric for each pair. However, many product pairs have no common customers or in other word most of the customers have enough purchases to correlate with other items. Therefore calculating similarities for each product pair approach is inefficient in terms of processing time and memory usage. In order the increase performance, we build our item-to-item matrix for products which have related products within predefined correlation time period.

Each step to create item-item matrix is described as follows:

1. Every time a customer purchases a item A, record transactions of other item B they purchased within the correlation time of buying A.

2. Find P(A → B):
    i. Find number of orders where B was bought after A was bought by the *same* customer C within the correlation time period of buying A we called this number as 'α'
    ii. Find number of orders where a customer bought A then bought *anything else* within the correlation time period. We called this number as 'β'
    iii. Divide 'α' by 'β' to get P(A → B). This is the probability which B will be bought if A is bought first.

3. Find P(X → B), where X refers to 'any' product:
    i. Find all transactions where a customer bought *any* product X and then bought B within the correlation time period. Call the number of such orders 'θ'
    ii. Find all transactions where a customer bought a product X and then any other product Y within the correlation time period. Call this number of such orders 'δ'.
    iii. Divide 'θ' by 'δ' to get P(X → B). This is the probability of B being bought after a purchase.

4. Divide P(A → B) by P(X → B) and subtract 1 to get the similarity which is actually equals to how much more likely are you to buy B after buying A than buying B after buying any random product X.

In more generalized way, below pseudo code (Figure 3.1) describes the overall algorithm to calculate similarities between each item. First loop iterates through the customers who purchased item I_i and purchased item Ij after Item I_i within predefined correlation period. If corr(I_i,I_j) satisfies the condition, customer purchase for given item pair is recorded into the set Λ which we called joint purchases of customers.

By looping through the joint purchases set Λ, algorithm calculates similarities between each pair by use of the $sim(I_i,I_j)$ function which is defined at the below equation.

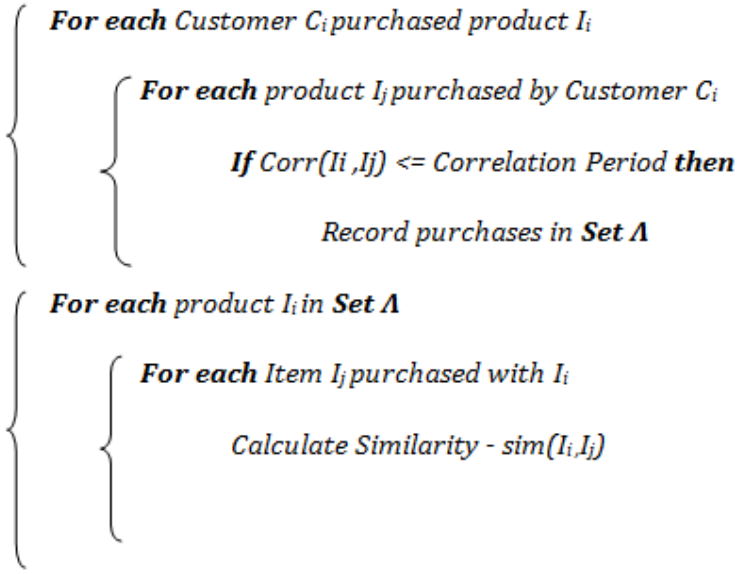

Figure 3.1: Pseudo Code for the RE

The algorithm can select recommendations from the similar customers' items using various methods, common techniques are to rank each item according to how many customers purchased it or how many purchase are done by customers. Using collaborative filtering to generate recommendations is computationally expensive. It requires scanning N items for each customers U which can be notated as *O(UN)* in the worst case, where U is the number of customers and N is the number of items.

For e-commerce data, we can say that an average customer who purchases distinct items is extremely sparse. Below table 3.1 shows the number of distinct customers and number of distinct items purchased by these customers. As you see, almost 55% of customers have purchased less than or equal to 5 different items.

Table 3.1: Customer distribution for different products buckets.

# of distinct Item=1	1 < # of distinct Item <=4	4 < # of distinct Item <= 7	7 < # of distinct Item <= 11	11 < # of distinct Item <=1 5	# of distinct Item > 15
18%	37%	17%	12%	6%	10%
Total # of Customers			815104		

Therefore algorithm's performance much closer to *O(U + N)* where cost of scanning every customer is approximately *O(U)* rather than *O(UN)* since most of the customer vectors contain a small number of items, regardless of the size of the items. For customers who have purchased a large amount of items from the catalog requires *O(N)* processing time. So, we can evaluate final performance of the algorithm as *O(U + N)*.

For very large data sets where customer vectors contain large number of items the algorithm encounters scalability issues. We tried to solve scalability issues by use of time period parameter where time period decrease the size of sample. We recorded execution[1] time of the RE algorithm for 68746 items which has purchased buy 774478 active users out of 257595 items on the production environment. Execution time of the algorithm for 14 days is given at table 3.2. Average execution time is about 10 minutes and 31 seconds with standard deviation of 0.001998.

Table 3.2: Execution time of the RE

Day	Date Executed	Duration
12	1/19/2009 3:00	0:08:50
11	1/18/2009 3:00	0:10:26
10	1/17/2009 3:00	0:08:31
9	1/16/2009 3:00	0:09:01
8	1/15/2009 3:00	0:14:36
7	1/14/2009 3:00	0:09:08
6	1/13/2009 3:00	0:09:06
5	1/12/2009 3:00	0:08:52
4	1/11/2009 3:00	0:12:01
3	1/10/2009 3:00	0:08:46
2	1/9/2009 3:00	0:09:12
1	1/8/2009 3:00	0:17:42

However, reducing sampling size may cause low quality recommendations. In order to overcome this issue, we introduce correlation time period variable which is amount of time period that Item B was bought after Item A was bought by the *same* customer C. We saw that limiting our item pairs to correlation period gives better results. Correlation time period must be greater or equal to 0 where 0 represents items bought in the same cart.

[1] : Production environment has eight X86 family 6 Model 23 Stepping 6 GeniuneIntel ~2493 Mhz processors, 16.387,64 MB physical memory and 293 GB available hard disk space (total 865 GB hard drive)

We introduced new similarity method which can be evaluated as extended probability based similarity. In usual probability based similarity method calculates conditional probability for given item i and j pair $P(j\,|i\,)$ is enough to calculate similarity between items. Instead of using this method we calculated our similarity index as follows:

$$sim(\,i,j) = \frac{p(j\,|i)}{p(j|X)} - 1 = \left(\frac{Freq(i,j)}{Freq(i)} \Big/ \frac{Freq(j,X)}{Freq(X)} \right) - 1$$

Where i is the item that user has already purchased or viewed, j is the item which customers in the system was purchased j after they purchased item i within the correlation time period and X represents any item in the product catalog that have occurrences within the correlation time period with item j.

Top-N recommendations are created by ordering N item in descending order for given product i by their likelihood ratio (similarity index). Bigger the likelihood ratios better we get recommendation qualities. We suppressed all items which has negative likelihood ratio since negative likelihood ratio value for i → j implies j is less likely a candidate to be a subsequent purchase than other products and they are not a good candidate for cross-sells.

We identify clearance items in the system within last three months and suppressed those items since they usually have bigger likelihood ratios over the other items.

3.2 FUTURE IMPROVEMENTS

Our implementation for item-item collaborative recommendation system still has some weaknesses. One of them is the cold-start problem which is common for most of the item-item CF type recommender systems where recommendations are required for items that no one has yet rated or purchased (new products). We found two solutions to this problem.

One possible solution can be create list of new products and ask for merchants to identify an existing product in the database closest to new products which already have some ratings/purchase history. We called those products as "proxy product" and recommendations for new products can be done using proxy products until the new product has enough history to generate recommendations. This method can produce good results since the experienced human interaction is involved the process. However maintaining such system might be expensive and even impossible when the number of new products in the system is increasing rapidly. Alternative solution to this problem can be combining our model with Content Based Collaborative Filtering approach to identify products similar to the new products. Content information can help bridge the gap from existing items to new items, by inferring similarities among them (Schein et al, 2002). So, we can make recommendations for new items that appear similar to other recommended items. In order to achieve good results for the content based collaborative filtering method, item catalogs must be well-formed.

Another improvement that we plan to implement is addressing recommendation quality problem. We plan to add seasonality analysis to identify product's seasonality index to increase recommendation quality. Seasonality[1] can be a good indicator to suppress bad recommendations. Our logic is: compute seasonality for product has been selling more than 12 months and generate seasonality index[2] for each season (Dec-Feb, Mar-May, Jun-Aug, Sept-Nov). If product has been selling for less than 12 months, its seasonality index is 0. Once the seasonality index of the product is computed, recommendations can be suppressed depend on the season and seasonality index of the product.

[1]: Seasonality is pattern in a time series that repeats itself at least once a year.

[2]: We haven't decided which method to calculate seasonality index yet.

4- PERFOMANCE EVALUATION METRICS FOR RECOMMENDER SYSTEMS

In this section, we introduce different types of evaluation metrics that we used to analyze performance of our recommender system. Performance of the recommender systems can be measured under two main categories Efficiency and Effectiveness (Rijsbergen, 1979). Efficiency is measured by the ratio of output over input, and effectiveness is measured by output quality. In a recommendation system, possible measures for its efficiency are response time of the algorithm to make recommendations and storage space used during the off-line and online computations. Effectiveness is simply a prediction quality of the algorithm or in other word recommendation quality. For the comparison of the recommender systems, we focus on only effectiveness since we don't know much about algorithm used by the other system.

There are several popular evaluation metrics available to measure performance of the recommender systems. But it is a challenging task to proof which algorithm is the best for given purpose. Because researchers disagree on which attributes should be measured, and on which metrics should be used for each attribute. Researchers who survey the literature will find different quantitative and qualitative metrics for performance evaluation of RS. Evaluating recommender systems and their algorithms is relatively hard for several reasons. First of all each algorithm has different behavior on different data sets. Thus, using same algorithm on different dataset may give better or worse results. Herlocker (2004) draw attention to good point which many collaborative filtering algorithms have been designed specifically for data sets where there are many more users than items. Such algorithms may be entirely inappropriate in a domain where there are many more items than users. Similar differences exist for ratings density, ratings scale, and other properties of data sets.

Another reason makes the evaluation difficult is that the goals for which an evaluation is performed may differ. Most of the article that we studied during our works focused specifically on the accuracy of collaborative filtering algorithms. However, accuracy alone may not be good performance indicator. Some of the works has speculated that there are other metrics that have larger effect on user satisfaction and performance. Because of that other metrics might be needed to achieve performance goals of the systems.

To analyze performance of our system, we used metrics described below. Sometimes different methods gave different results and algorithm that we compared has gained advantage on our algorithms. We will introduce these details in the experimental work section.

The most popular and widely used techniques that we discuss in this section are Precision & Recall, Receiver Operating Characteristic (ROC), Coverage, Hit Rate (HR) and statistical accuracy metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

4.1 PRECISION AND RECALL

In order to evaluate top-N recommender systems two metrics are widely used in Information Retrieval (IR): These are recall and precision. Precision is defined as the ratio of relevant recommended items to the total number of items recommended:

$$Precision = {}^{n_{rs}}/_{n_s}$$

Where n_{rs} is the number of relevant items selected and n_s is the number of items selected.
Recall is defined as the ratio of relevant items selected to the total number of relevant items;

$$Recall = {}^{n_{rs}}/_{n_r}$$

Where n_{rs} is the number of relevant items selected and n_r is the number of relevant items.
Precision and Recall are computed from a 2x2 matrix, shown in table 4.1. Item set must be separated into two classes which are relevant and not relevant. Also item set must be separated into two different sets where one is selected which represent recommended items and other is not selected represents all items (Herlocker et al, 2004).

Table 4.1: Generalized Precision & Recall matrix

	Selected	Not Selected	Total
Relevant	N_{rs}	N_{rn}	N_r
Irrelevant	N_{is}	N_{in}	N_i
Total	N_s	N_n	N

4.2 HIT RATE

Hit Rate (HR) is a metric which can measure recommendation quality by looking at the number of hits within the top-N items that were recommended. The number of hits is the number of items in the test set that was also present in the *top-N* recommended items returned for each item. Hit rate is defined as follows:

$$Hit\ Rate = \frac{Number\ of\ hits}{n}$$

Where *n* is the total number of items. HR equals to 1.0 indicates that the algorithm is always able to a recommend the hidden item, whereas HR equals to 0.0 indicates that the algorithm is not able to recommend any of the hidden items.

4.3 COVERAGE

Coverage is a measure of the percentage of items which CF algorithm can provide predictions. Recommender systems are usually not able to generate a prediction for some of the items because of the data sparsity or because of other factors such as low similarity values which are computed during the item-item matrix generation. Such cases lead to low coverage values. A low coverage value indicates that the recommender system is not able to recommend items for most of the items available in the system. High coverage value indicates that the recommender system is be able to provide recommend for most of the items that user interested. Coverage is defined as follows:

$$Coverage = \frac{\sum_{i=1}^{m} np_i}{\sum_{i=1}^{m} n_i}$$

Where n_i are the items available in the system and np_i is the number of items which the recommender system is able to generate a prediction.

4.4 STATISTICAL ACCURACY METRICS

Statistical recommendation accuracy metric measures the closeness between the numerical recommendations provided by the system and the numerical ratings provided by the user for the same items. Common metrics used for this purpose are Mean Absolute Error, Mean Square Error and Root Mean Squared Error.

4.4.1 MEAN ABSOLUTE ERROR

The MAE measures the average absolute deviation between a predicted rating and the user's true rating.MAE is defined as follows:

$$MAE = |\bar{E}| = \frac{\sum_{i=1}^{N}|p_i - r_i|}{N}$$

Where p_i is the predicted rating and the r_i is the actual user ratings and N is the total number of the item's rated by the users.

4.4.2 MEAN SQUARE ERROR

The MSE measure is a variation of the MAE where the square the error before summing it instead of taking absolute value of the error. The result is more emphasis on large errors. For example, an error of one point increases the sum of error by one, but an error of two points increases the sum by four (Herlocker et al, 2004). MSE is defined as follows:

$$MSE = E^2 = \frac{\sum_{i=1}^{N}(p_i - r_i)^2}{N}$$

Where p_i is the predicted rating and the r_i is the actual user ratings and N is the total number of the item's rated by the users.

4.4.3 ROOT MEAN SQUARED ERROR

RMSE is a measure of error that is biased to weigh large errors disproportionately more heavily than small errors. In most of the cases RMSE indicates better accuracy than MAE.

$$RMSE = \sqrt{E} = \sqrt{\frac{\sum_{i=1}^{N}(p_i - r_i)^2}{N}}$$

Where p_i is the predicted rating and the r_i is the actual user ratings and N is the total number of the item's rated by the users.

4.5 RECEIVER OPERATING CHARACTERISTIC

Recently, ROC analysis became the most common accuracy metric to analyze performance of the recommender systems. ROC graph is a technique for visualizing, organizing and selecting classifiers based on their performance. One of the earliest adopters of ROC graphs in machine learning was Spackman (1989), who demonstrated the value of ROC curves in evaluating and comparing algorithms.

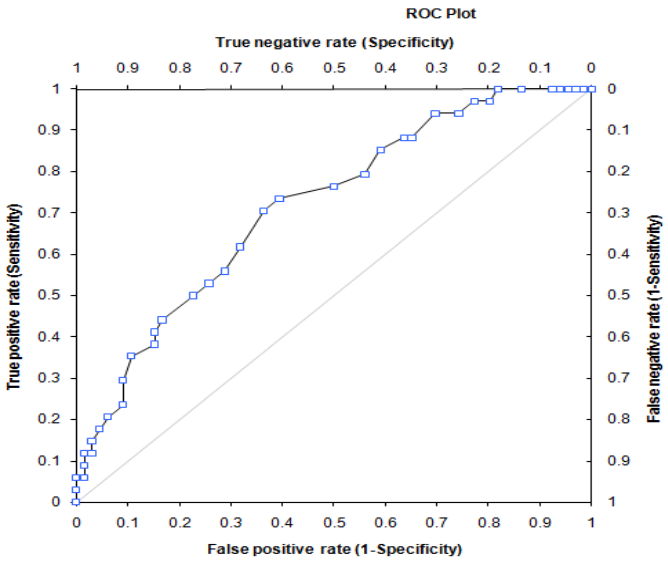

Figure 4.1: Sample ROC curve

ROC sensitivity can be a good metric for measuring the system's ability to discriminate between good and bad recommendations, independent of the search length (Herlocker et al, 2004). ROC sensitivity denotes the Area Under the Curve (AUC), commonly referred to as the ROC curve, is a measure of the diagnostic power of a filtering algorithm. A ROC curve plots the sensitivity and the specificity of the filtering test. More accurately, it plots sensitivity and 1-specificity. The ROC curve ranges from 0 to 1.

Sensitivity (eq-1) is the probability of a randomly selected good recommendation actually being rated as good, and as a result being accepted by the filtering algorithm. Specificity (eq-2) is the probability of a randomly selected bad recommendations actually being rated as bad.

	p	**n**
Y	**T**rue **P**ositives	**F**alse **P**ostives
N	**F**alse **N**egatives	**T**rue **N**egatives
Totals	**P**	**N**

$$Sensitivity = Recall = \frac{TP}{P}, \quad Specificity = \frac{TN}{FP+TN}$$

It is important to note that there exists a trade-off between sensitivity and specificity. Any increase in sensitivity will be accompanied by a decrease in specificity. The area under the ROC curve, corresponding to ROC sensitivity, and increases as the filter is able to detect more good recommendations and at the same time decline more bad recommendations. An area of 1 represents the perfect filtering algorithms, while an area of 0.5 represents a random filter (Vozalis & Margaritis, 2003).

Herlocker et al (2004) list advantages and disadvantages of the using AUC metric as follows:

1. It provides a single number representing the overall performance of an information filtering system.
2. It is developed from solid statistical decision theory designed for measuring the performance of tasks such as recommender system performs
3. It covers the performance of the system over all different recommendation list lengths.

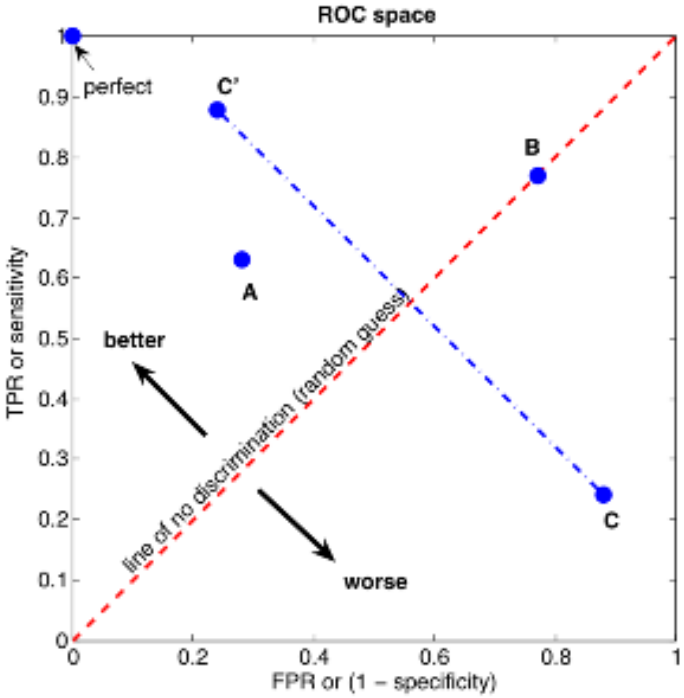

Figure 4.2: ROC space and plots of the four prediction examples.

The disadvantages of the AUC metric are as follows:

1- A large set of potentially relevant items is needed for each query.
2- For some tasks, such as *Find Good Items* users are only interested in performance at one setting, not all possible settings.
3- Equally distant swaps in rankings have the same effect no matter where in the ranking they occur.
4- It may need a large number of data points to ensure good statistical power for differentiating between two areas.

5-EXPERIMENTAL RESULTS

In this section, we experimentally evaluate our item-item based CF recommendation algorithm and compare their performance against the performance of the click stream based CF recommendation algorithm. Sub sections explain data set used in this experiment, our implementation for evaluation metrics explained above to analyze effectiveness of our novel recommendation engine algorithm. We don't give any comparative result for efficiency of the algorithms since we don't know the underlying algorithm used in click stream based CF algorithm. All experiments were performed on same environment: Microsoft Windows Server 2003^{tm} operating system, Intel® Xeon® CPU E5420 @ 2.50 GHZ (eight processors), 15.9 GB of RAM and 293 GB available hard disk space (total 865 GB hard drive).

5.1 DATASET

Dataset used in this experiment was collected from e-commerce site which sells contemporary apparels. We used three main tables to test and generate recommendation results. These tables given below figure 5.1 are: Item table which has definition for items in the product catalog, Order Item table that stores customers purchase history and Endeca table that stores currently available product list on the e-commerce site.

Order Item				
Column Name	**Data Type**	**Length**	**Precision**	**Scale**
CUST_ID	numeric	9	10	0
ORDER_ID	nvarchar	16		
ITEM_ID	nvarchar	40		
QUANTITY	numeric	5	4	0
PROCESS_DATE	datetime	8		

Endeca				
Column Name	**Data Type**	**Length**	**Precision**	**Scale**
TransType	varchar	255		
Product	varchar	255		
Parent_id	varchar	255		
name	varchar	255		
item_no	varchar	255		

Item				
Column Name	**Data Type**	**Length**	**Precision**	**Scale**
ITEM_ID	nvarchar	40		
DIV_CODE	nvarchar	4		
ITEM_DESC	nvarchar	100		
STYLE	nvarchar	24		
GALVIN_DIV	nvarchar	4		
GALVIN_DEPT	nvarchar	4		
GALVIN_CLASS	nvarchar	4		
DEPT	nvarchar	8		
FIRST_ADVERT_DATE	datetime	8		
LAST_ADVERT_DATE	datetime	8		
LAST_CHANGE_DATE	datetime	8		
STYLE_DESC	nvarchar	60		
DW_PROCESS_DATE	datetime	8		
DW_LAST_CHANGE_DATE	datetime	8		

Figure 5.1: Dataset Structure

Like most of the e-commerce site, there were more users than the items available on the web site during our experiment. We have used customer orders made between January 1st 2006 for the train set where it had 442512 customers and 544 different items and March 18th 2008 and for test set, we used orders made between March 18th 2008 and May 7th 2008 where it had 82868 customers and 528 items. There were 2174835 joint purchases (purchases of customers for each pairs of item A and B where item B purchased after item A) for train set and 691822 joint purchases for train set to generate recommendations and cross validation respectively. Recommendations were generated by using train set and cross validation was done by using of test set. We also suppressed some of the items from the train set according to business rule supplied by the data owner (i.e. we suppressed recommendations within same department).

Recommendation set generated by Intelligent Offer[tm] (IO) for the same data set had recommendations at the item level where our recommendation engine had recommendation at the parent level. We converted item level recommendations to parent level by using same transition algorithm from item to parent level.

Since data sparsity has a huge impact on the recommendation results, we have made analysis for last 4 years data to identify data sparsity level. We have created six buckets for the number of distinct items purchased by customer and looked at the customer orders made for last 4 years (Table 5.1). We saw that customers have been following same buying trend where approximately 65% of the customers have purchased less than or equal to the 4 distinct items within the same year (Figure 5.2). We saw the same trend for the train and the test sets (Table 5.2) which are subset of the all orders between 2006 and 2008. We concluded that expecting high quality recommendation for each item is relatively impossible because of the data sparsity level.

Table 5.1: Number of the customers belongs to number of distinct item bucket for last 4 years

Years	1	1-4	5-7	8-11	12-15	>15	Total Customer
2007	86248	172770	83984	25690	16612	17582	404893
2006	82401	164515	82503	25650	16682	18665	392422
2008	65517	135111	65835	19326	11902	11768	311467
2009	5185	8335	2093	267	95	58	18042

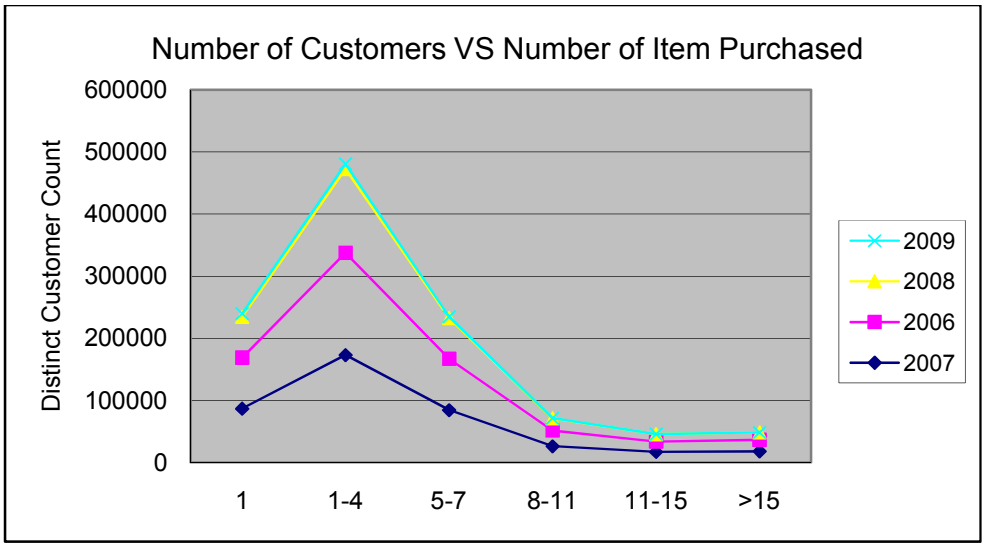

Figure 5.2: Customer purchase trend for last four years

Table 5.2: Number of the customers belongs to number of distinct item bucket for the Train and the Test sets.

Set Name	1	1-4	5-7	8-11	11-15	>15	Total Customer
Train Set	26995	41923	10111	2941	625	273	82868
Test Set	178791	189899	46414	18369	5584	3455	442512

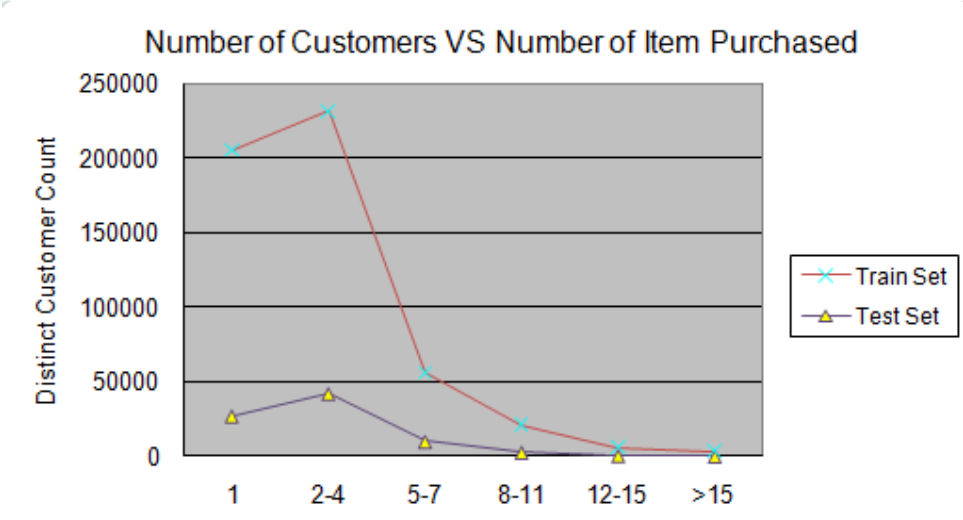

Figure 5.3: Customer purchase trend for the Train and the Test sets

5.2 EVALUATION METRICS

We applied different measures (please look at section 4) in order to evaluate the performance of the different types of CF algorithms employed by recommender systems. We will compare item-item based CF based recommendation engine results with the click stream based CF recommendation engine where each system produces same output that is Top-N recommendations for given item A (or product A) . Each evaluation scheme was adjusted to dataset and algorithm result used in this experiment. Here in this section, evaluation metrics which refer to the effectiveness of the output will be presented. Metrics that evaluate the efficiency of the recommender systems in terms of response time and space requirements are not in scope of this experiment. For some of the performance test, we have compared two systems for sample recommendation output size N=538 which is the all available item size and N=313 selected randomly to reduce difference between the available recommendation set.

5.2.1 PRECISION AND RECALL TEST

We implemented precision and recall test as follows:

1. If the system able to recommend at least two successful items out of 10 recommendations, we considered it as relative class, otherwise we consider it as not relative class.
2. For positive and negative classification, we looked at whether the user purchased the recommended item or not.

We have selected recommendation outputs for two dataset where Dataset-1 N = 313 and Dataset-2 N=538. We have generated 2x2 confusion matrix (table 5.2 and table 5.3 respectively) for RE and IO (click stream based algorithm) as follows:

Table 5.3: Confusion matrix for RE

	N=538		N=313	
	positive	negative	positive	negative
Relative (Y)	2841	1689	1631	969
Not Relative (N)	37	657	22	420
Totals	2878	2346	1653	1389

Table 5.4: Confusion matrix for IO

	N=538		N=313	
	positive	negative	positive	negative
Relative (Y)	1511	1509	883	843
Not Relative (N)	64	750	37	467
Totals	1575	2259	920	1310

We have calculated false positive rates, true positive rate (Recall), precision, accuracy and F-Score in table 5.4 and compared the results. Among all these values, we focused on two metrics that are precision and recall. Results for all sample sizes have shown us our algorithm gave better results than IO. However recall value for both systems was very close to each other.

Table 5.5: Precision and Recall calculations

	RE	IO		RE	IO	
	N=538	N=538	RE>IO	N=313	N=313	RE>IO
FP Rate	0.719948849	0.667994688	TRUE	0.69762419	0.64351145	TRUE
TP Rate (Recall)	0.98714385	0.959365079	TRUE	0.986690865	0.959782609	TRUE
Precision	0.627152318	0.500331126	TRUE	0.627307692	0.511587486	TRUE
Accuracy	0.669601838	0.589723526	TRUE	0.674227482	0.605381166	TRUE
F-Score	0.619089554	0.48000021	TRUE	0.61895877	0.491012771	TRUE

5.2.2 HIT RATE TEST

We measured hit rate for two different aspects. First one is purchase based where hit rate is number of purchases made by recommendations results divided by number of all purchases made. And the second one is based on number of hit in the Top-N recommendation divided by number of all recommendations made by the algorithm. Table 5.5 shows the HR values for each system and aspects where purchase based HR is very small for both systems. Because the number of orders usually much more than number of items in the system. For both tests, we recorded better performance for RE.

Table 5.6: Hit Rate Values for RE and IO

	Total Purchases	Rec. Purchases	Hit Rate (Purchase Based)	Number of Hits	Number of All Recommendations	Hit Rate
IO	630211	24572	0.038990116	1575	3770	0.417772
RE	630211	48986	0.077729522	2851	5187	0.549643

5.2.3 COVERAGE

As we discussed before, coverage is a measure of the percentage of items which CF algorithm can provide predictions. We interpreted prediction for item as if a system is able to recommend at least 1 or 6 out of Top 10 recommendation for given item A. Test set we used has 538 items for purchases made between March 18th 2008 and May 7th 2008. We calculated coverage value for RE as 96% and 86% for IO for condition-1 where at least one prediction for item. Coverage values for condition – 2 where at least 5 prediction for item: were 78% for IO and 96% for RE. We observed better results for RE for both conditions (Table 5.6).

Table 5.7: Coverage values for RE and IO.

	Total Item	Total Prediction -1	Total Prediction - 2	Coverage - 1	Coverage - 2
IO	538	460	424	0.855018587	0.788104089
RE	538	520	519	0.966542751	0.964684015

5.2.4 STATISTICAL ACCURACY METRICS

Most of the research we reviewed during our works had used statistical accuracy metrics to measure the error between a predicted rating and the user's true rating. However in our case we didn't predict rating for item. Instead we recommend 10 items which we thought customers most likely to buy with given Item A. Thus, we needed to modify statistical accuracy metrics to adapt it for our test. We decided to measure error rates between all recommendations and recommendations which were successful within the Top-N recommendations for each product. We calculated Mean Absolute Error, Mean Square Error and Root Mean Squared Error and found below results shown in table 5.7.

Table 5.8: Statistical accuracy metric results for RE and IO.

	RE Result	IO Result	RE>IO
MAE	4.342007435	4.079925651	FALSE
MSE	27.59776536	23.29422719	FALSE
RMSE	5.248472959	4.821921751	FALSE

For MAE and RMSE values both systems had very close results. However IO performance seems little bit better than RE where it had less error rate.

As a complementary analysis to statistical accuracy metrics, we also used Opportunity Success Rate. The opportunity success rate provides information about won and lost opportunities. The OSR helps the user to assess the sales performance. Opportunity success rate is defined as follows:

$$Opportunity\ Success\ Rate = \frac{Number\ of\ Opportunites - Status\ Won}{Number\ of\ Opportunities - Status\ Won\ and\ Lost}\ X\ 100$$

In our problem, we defined Number of Opportunites − Status Won as number of recommendation bought by the customer in the test set. Number of Opportunites − Status Won *and Lost* corresponds to number of recommendation made by the system.

Table 5.9: OSR rates for RE and IO.

	Matching Recommendations	Total Recommendations	OSR
IO	1575	3770	41.77%
RE	2851	5187	54.96%

5.2.5 ROC Graph and AUC

For our last test, we used sample data size N=538 and plot our ROC curves according to following rule:

1- We categorized recommendation engine results as N>=5 and N<5. If a recommender system able to recommend at least five items, we categorized it as N>=5 , otherwise we categorized it as N<5
2- We sorted the recommendation results by N values in descending order and for each recommendation result; we calculated true positives and false positive frequencies. True positive frequency is number of total successful recommendations[1] at given level divided by number of the total unsuccessful recommendations[2]. And for the false positive frequency is the number of total not relevant recommendations at the level divided by the number of total not relative recommendations.
3- Using TPF and FPF, we calculated Area for each item. The AUC is the summation of the areas calculated at each item.

Below Figure 5.3 and 5.4 shows ROC graph for each system and the AUC values.

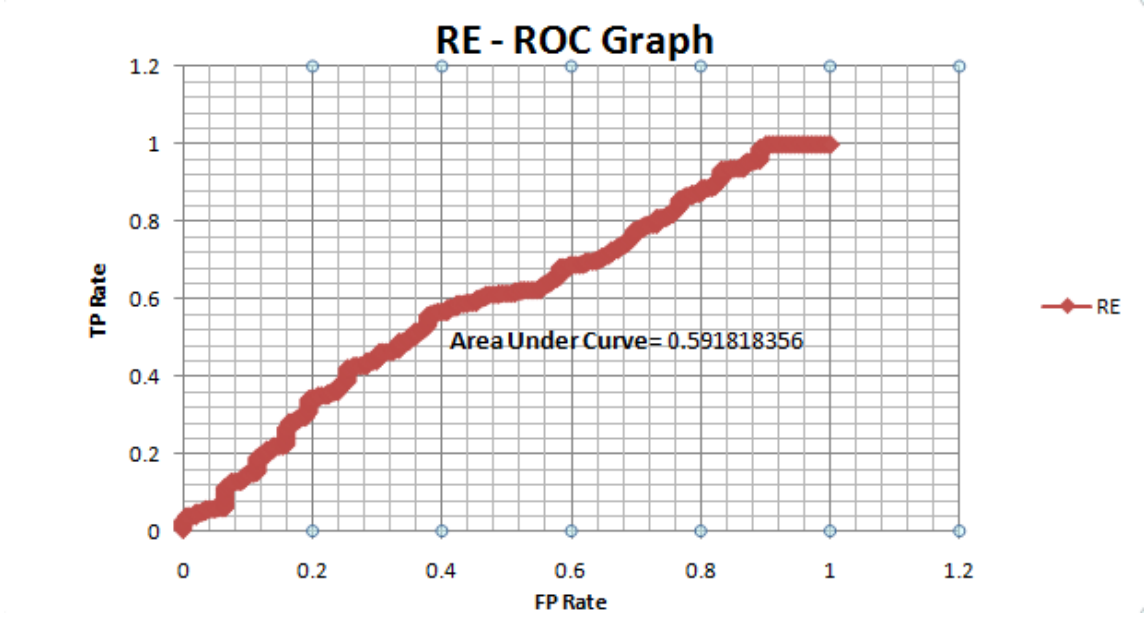

Figure 5.4: ROC Curve for RE

[1] : In order to evaluate a recommendation as a successful, there must be hit with given set.
[2] : In order to evaluate a recommendation as an unsuccessful, there must be no hit within given set.

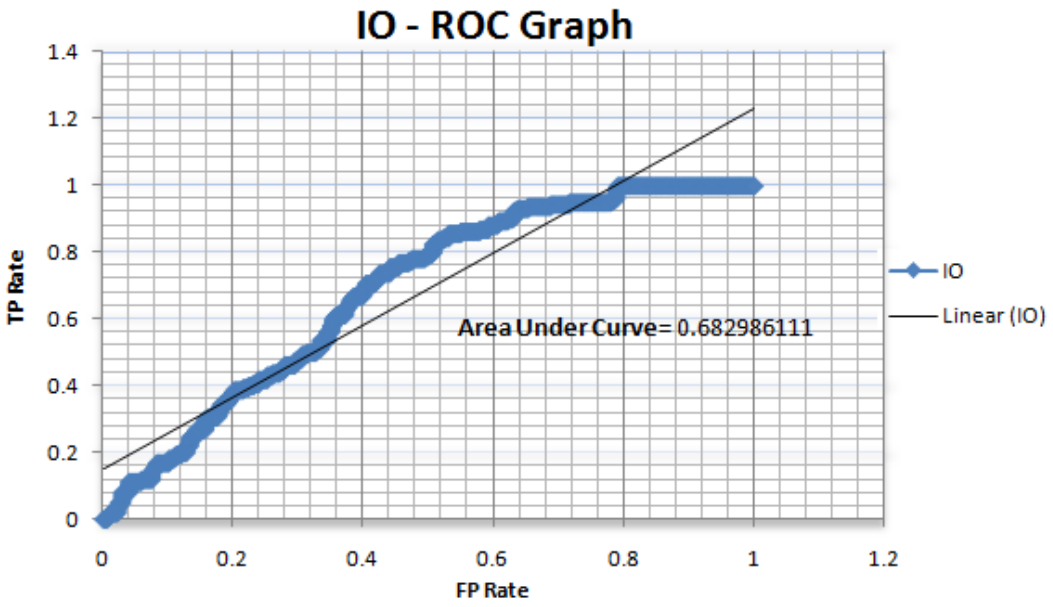

Figure 5.5: ROC Curve for IO

According to experiment results, the AUC for RE was calculated as ≈60% and the AUC for IO was calculated as ≈68%. There were only ≈8% percent different between the two systems. ROC test didn't give us expected results for RE. We might need to redefine class definition to get better test results since ROC for Top-N recommendation result is relatively hard task. Another reason was, at the time we did the experiment; coverage value of the RE was bigger than the IO. We think this might lead more false positive results.

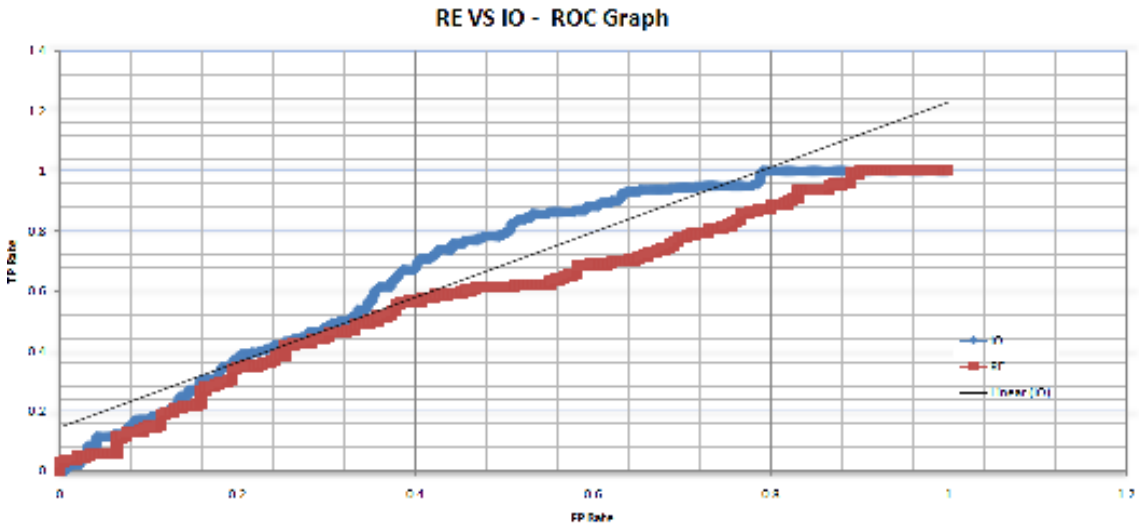

Figure 5.6: Comparative ROC Curve RE vs. IO

6-CONCLUSION AND FUTURE PLANS

Recommender systems are software applications that help users in their decision-making while interacting with large information spaces especially in the World Wide Web. They also help business owners who are trying to sell their products online. As we discussed in previous sections of this thesis, there exist many popular and successful implementations such as Amazon.com and Netflix. Proven success of such systems influenced other business owners and use of the recommender systems increases rapidly as a part of the E-commerce sites. Before we started this thesis our aim was to design a recommender system that was going to be a part of the E-commerce site. Our initial research has shown us designing both efficient and effective recommender system is a challenging task. Our works and experience during the design and implementation of the recommender system has constructed this thesis. The system, presented in this thesis is currently working as part of the e-commerce site and results shown us our implementation for the real-world data is working fine.

In this thesis, we presented new algorithm which was an alternative to the existing click stream based CF algorithm. We also empirically evaluated our recommender system performance against the existing recommender system. However, there isn't any systematical way to evaluate performance of the recommender systems for different tasks and different contexts. In order to achieve our goal, we gave brief information about well known collaborative filtering algorithms and present metrics to compare different types of recommender systems. During our performance analysis, we focused on effectiveness of the recommender system. Experimental results has shown us for Top-N type of recommender systems and the data used in context of this thesis, our algorithm has shown better performance than the click stream based algorithm in terms of recommendation quality. In experimental results section, we evaluated results of our recommender system with our interpretation to well known techniques. For the precision and recall tests with different sample sizes our algorithm worked better than click stream one. Also RE gave much better results for coverage and hit rate tests. For statistical accuracy metrics those are MAE, MSE, RMSE and OSR, click stream based algorithm gave small error values, however OSR value of the click stream based algorithm was less than our algorithm. ROC curve and AUC tests worked better for the clicked stream based algorithm but there were two points that made us uncertain about this test.

First one was our interpretation might causes these results. We might need to go back and redo this test again with different samples and classifications. Second thing was ROC curve might not be the suitable way to compare Top-N recommender systems quality.

Besides the experimental results, there is one point we would like to underline. Our algorithm is enough scalable to process large size of data set within acceptable time period approximately 15-20 minutes in average. We have already added analyzing efficiency performance analysis task to our future plans by comparing our algorithm with the well known item-item CF based algorithms.

Our other future plans consist of following topics:

- Adding new suppression rules to generate more accurate recommendations: We plan to add seasonality analysis to our recommender system to avoid effect of seasonality on the purchases. We are still trying to find out how to define similarity index (some details has given in section 3.2)

- Using Content Based Filtering algorithms to overcome cold start problem of the recommender system: We described cold start problem in previous section of this thesis. Our aim is to use content based collaborative filtering algorithm to identify similar products for new merchandises and make new merchandises available for recommendations.

- Creating hybrid systems: We plan to use Response Model and Recommendation Engine together to increase marketing abilities of business owner. Response model is used to define customers who have intention to purchase goods. If we can design a model that can create custom recommendations (by use of recommendation algorithm presented here) to target customers, this may decrease the marketing cost and increase the sales.

- Creating more generic recommender system: The algorithm we presented in this thesis was designed for specific business. We plan to improve our item-item based algorithm to work in a more generic way so that it can be implemented like all other plug and play systems.

REFERENCES

Adriaans, P., & Zantinge, D. (1996). *Data Mining*. Addison-Wesley Longman.

Balabanovic, M. and Shoham, Y. Fab (1997). *Content-Based, Collaborative Recommendation*, in Resnick and Varian, p 66-72

Berry M.J. and Linoff G. S. (2000). *Mastering data mining, the Art and Science of Customer Relationship Management*. Wiley Computer Publishing, John Wiley & Sons, Inc.

Breese, J., Heckerman, D., & Kadie, C. (1998). *Empirical Analysis of Predictive Algorithms for Collaborative Filtering,* In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98), p43-52.

Chan, P. (1999). *A non-invasive learning approach to building web user profiles,* In Proceedings of ACM SIGKDD International Conference. ACM, New York

Claypool M., Gokhale A.,Miranda T.,Murnikov P., Netes D., and Sartin M. (1999), *Combining Content-based and Collaborative Filters in an Online Newspaper*. In Proceedings of ACM SIGIR Workshop on Recommender

Deshpande M. & Karypis G.(2004), *Item-Based Top-N Recommendation Algorithms*, ACM Transactions on Information Systems (TOIS) Volume 22 , Issue 1 ISSN:1046-8188, p143

Deshpande M. and Karypis G. (2001), *Item-Based Top-N Recommendation Algorithms*, Technical Paper, University of Minnesota

Deshpande M. and Karypis G. (2001). *Selective Markov models for predicting Web-page accesses*, First SIAM International Conference on Data Mining

Good, N., Schafer, J.B., Konstan, J., Borchers, A., Sarwar, B., Herlocker, J., and Riedl, J., (1999), *Combining Collaborative Filtering with Personal Agents for Better Recommendations*, in Proceedings of AAAI'99

Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; Riedl, J. T. (2004), *Evaluating collaborative filtering recommender systems*, ACM Transactions on Information Systems (TOIS), Volume 22 , Issue 1, ISSN:1046-8188 , p5 - 53

Hofmann T. (2003), *Gaussian Latent Semantic Models for Collaborative Filtering*, In the Proceedings of the 26th Annual International ACM SIGIR Conference

Jin R., Joyce Y. and Si L. (2004*): Content-based filtering & collaborative filtering*, Annual ACM Conference on Research and Development in Information Retrieval Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval Sheffield, United Kingdom, p. 337 - 344

Kim D., Atluri V., Bieber M.,Adam N., Yesha Y. (2004), *A Clickstream-Based Collaborative Filtering Personalization Model: Towards A Better Performance*, Proceedings of the 6th annual ACM international workshop on Web information and data management, ISBN:1-58113-978-0 , p. 88-95.

Kitts, b., Freed, d., and Vrieze, m. (2000), *Cross-sell: A fast promotion-tunable customer–item recommendation method based on conditional independent probabilities*, In Proceedings of ACM SIGKDD International Conference. , ACM, New York, 437–446.

Linden G.,Smith B., and York J.(2003), *Amazon.com Recommendations: Item-to-Item Collaborative Filtering*, IEEE Internet Computing Volume 7, Issue 1, ISSN:1089-7801, p. 76-80

Mild A. And Natter M. (2001), *A Critical View on Recommendation System*, Working Paper No. 82

Mobasher B., Dai H., Luo T., Nakagawa N. (2002). *Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks*, In Proceedings of the IEEE International Conference on Data Mining (ICDM2002), Maebashi City, Japan, p. 669-672.

Pennock D. M., Horvitz E., Lawrence E. and Giles C. L. (2000), *Collaborative Filtering by Personality Diagnosis: A HybridMemory- and Model-Based Approach*, in Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI). pp. 473 - 480

Resnick, P. & Varian, H. R. (1997). *Recommender systems*, Commun. *ACM 40*, 56–58

Rijsbergen C. J. (1979), *Information Retrieval*, Butterworths, London, second edition

Sarwar B., Karypis G., Konstan J., and Riedl J. (2002), *Recommender Systems for Large-Scale E-Commerce: Scalable Neighborhood Formation Using Clustering*, Minneapolis, Minnesota, United States

Sarwar B., Karypis G., Konstan J., and Riedl J. (2001), *Item-based collaborative filtering recommendation algorithms* , In WWW '01: Proceedings of the 10th International *c*onference on World Wide Web, Hong Kong. ACM Press, pages 285–295

Sarwar B., Karypis G., Konstan J., and Riedl J. (2000) *Analysis of Recommendation Algorithms for Ecommerce,* Minneapolis, Minnesota, United States Pages:158 - 167 ISBN:1-58113-272-7

Sarwar B., Konstan J., Borchers A., Herlocker J., Miller B and Riedl J. (1998), *Using Filtering Agents to Improve Prediction Quality*, in the GroupLens Research Collaborative Filtering System

Schein A. I.,Popescul R., Ungar L. H., Pennock D. M., (2002), *Methods and metrics for cold-start recommendations*, In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval

Shardanand, U. and Maes, P. (1995), *Social Information Filtering:Algorithms for Automating - Word of Mouth*, In Proceedings of CHI '95. Denver, CO.

Spackman, K. A. (1989). *Signal detection theory: Valuable tools for evaluating inductive learning*, In Proceedings of the Sixth International Workshop on Machine Learning, San Mateo, CA. Morgan Kaufman, p60-63

Terveen and Hill W. (2001), *Beyond Recommender Systems: Helping People Help Each Other*, AT&T Labs – Research

Terveen L.,Amento B.,Hill W. (1999), *ACM Transactions on Computer-Human Interaction (TOCHI)*,Volume 6 , Issue 1 ,ISSN:1073-0516, p. 67 - 94

Ungar L.H. and Foster D.P, (1998), *Clustering methods for collaborative filtering*. In Workshop on Recommendation Systems at the Fifteenth National Conference on Artificial Intelligence, July 1998.

Vozalis E. and Margaritis K.G. (2003), *Analysis of recommender systems algorithms*, In Proceedings of the Sixth Hellenic-European Conference on Computer Mathematics and its Applications - HERCMA

Vucetic S. and Abrodovic Z, (2005), *Collaborative Filtering Using a Regression-Based Approach, Knowledge and Information Systems*, Volume 7, Issue 1, ISSN: 0219-1377
, p.1 - 22

VITA

Ergin DEMİREL was born in Ankara. He received his under graduate degree in Computer Engineering from Eastern Mediterranean University, TRNC, in 2005. He has been working in IT Company (Agilone LLC.) as a programmer and BI analyst. His areas of interests are Data warehouses, data mining applications and enterprise web programming.