國 立 交 通 大 學

管理學院（資訊管理學程）碩士班

碩 士 論 文

動態協同式過濾推薦之系統實做

A System Implementation of Dynamic Collaborative Filtering for Recommendation

研 究 生：廖學毅

指導教授：劉敦仁　教授

中 華 民 國 九 十 六 年 七 月

動態協同式過濾推薦之系統實做

A System Implementation of Dynamic Collaborative Filtering for Recommendation

研 究 生：廖學毅　　Student：Roger Hsueh-Yi Liao

指導教授：劉敦仁 教授　Advisor：Dr. Duen-Ren Liu

國 立 交 通 大 學

管理學院（資訊管理學程）碩士班

碩 士 論 文

A Thesis

Submitted to Institute of Information Management

College of Management

National Chiao Tung University

In Partial Fulfillment of the Requirements

For the Degree of

Master of Science

in

Information Management

July 2007

Hsinchu, Taiwan, the Republic of China

中　華　民　國　九　十　六　年　七　月

動態協同式過濾推薦之系統實做

研究生：廖學毅　　指導教授：劉敦仁 教授

國立交通大學管理學院（資訊管理學程）碩士班

摘要

在現今廣受應用的資料探勘領域裡，推薦系統廣泛被應用於建議相關商品給使用者。在眾多新興電子商務中，線上電影推薦系統如"IMDb(Internet Movie Database)"， "MovieFinder.com"，"MovieLens"等，皆已廣為大眾喜愛和接受。這些電影推薦系統都是使用「協同過濾」(Collaborative Filtering) 技術來推薦相關影片給使用者。在這篇研究中，我們嘗試根據使用者線上瀏覽歷史和影片評價紀錄，運用「使用者相關性」(UserCorrelation)、「商品相關性」(ItemCorrelation)，以及「斜率性預測」(SlopeOne Predictor)，來建立一個「動態協同式過濾推薦系統」。此外所實做之系統運用「點閱流樹」(ClickStream Tree)技術，來預測使用者下一步將瀏覽的網頁。研究方法發現「商品相關性」(ItemCorrelation)是協同過濾演算法中較有效率的，而「斜率性預測」(SlopeOne Predictor)則是較精確的方法。此動態推薦系統以「服務性架構」(Service Oriented Architecture)及「商業流程執行語言」(Business Process Execution Language)為實做基礎，能有效地根據資料的特性進行相關商品推薦。

A System Implementation of Dynamic Collaborative Filtering for Recommendation

Student：Roger Hsueh-Yi Liao　　Advisor：Dr. Duen-Ren Liu

Institue of Information Management

National Chiao Tung University

Hsinchu, Taiwan, Republic of China

Abstract

As a popular application of data mining, recommender systems attempt to predict items that a user may be interested in, given some information about the user's profile. With the gradually increasing use of IMDb (Internet Movie Database), MovieFinder.com, MovieLens and the likes, recommendation systems are gaining more popularity and acceptance by the public. Such film recommendation systems make use of collaborative filtering technology to recommend films to users. In this research, we propose a dynamic collaborative filtering system that makes use of UserCorrelation, ItemCorrelation, SlopeOne Predictor, and Clickstream collaborative filtering model to make use of both user navigation patterns along side historical purchased items for users with similar buying behavior. With the proposed dynamic model, we predicted the potential next page (movie title) of interest with higher confidence via the help of clickstream tree. We observed that ItemCorrelation is the faster recommendation scheme, and SlopeOne Predictor is the more accurate scheme. Our dynamic recommendation system based on SOA, orchestrated by BPEL dynamically switches among the schemes to generate the more accurate recommendation within a timely fashion in a scalable manner.

誌 謝

感謝我的恩師劉敦仁教授多次的提供方向與建議，在劉老師的悉心指導下，讓我能對推薦系統的演算法和實做有了非常深入的了解。雖然來回改了很多次，但老師總能清楚的指引讓我能從模糊地帶慢慢的抓住重點。在交大的兩年我從結婚到有小孩，特別要感謝的是我老婆在我背後給予支持幫我寫作業、繪圖、改論文、帶小孩等，沒有她的加持我看我得延畢了。

也特別感謝爸爸跟媽媽能毫無怨言的照顧我們家的小歐寶貝，讓我能有充裕的時間來完成我的論文。其次要感謝交大資管所裡的老師和實驗室的同學們，他們的指導與鼓勵和適當的給予壓力，讓我能順利的完成學業。最後要感謝的是浩然圖書館提供了我一個舒適的環境讓我能在最後的一兩個月靜下心來衝刺我的研究。

Table of Contents

Index of Figures

Index of Tables

Index of Charts

1. Introduction

1.1. Background

The data that one collects about their customers is one of the greatest assets of that business. Buried within this vast amount of data are all sorts of valuable information that could make a significant difference to the way in which any business organization run their business, interact with their current and prospective customers and gaining the competitive edge on their competitors. Data mining is a set of automated techniques used to extract buried or previously unknown pieces of information from large databases, using different criteria, which makes it possible to discover patterns and relationships. The derived information can be utilized in the areas such as decision support, prediction, forecasting and estimation to make important business decisions. Data mining uses the business data as raw material using a predefined algorithm to search through the vast quantities of raw data, and group the data according to the desired criteria that can be useful for the future target marketing. Creating a picture of what is happening relies on the collection, storage, processing and continuous analysis of large amounts of data to provide the information that the particular business will need [1].

As a popular application of data mining, recommender system attempt to predict items that a user may be interested in, given some information about the user's profile. Recommendation systems work by collecting data from users, using a combination of explicit and implicit methods.

Examples of explicit data collection include the following:

- Asking a user to rate an item on a sliding scale.

- Asking a user to rank a collection of items from favorite to least favorite.
- Presenting two items to a user and asking him/her to choose the best one.
- Asking a user to create a list of items that he/she likes.

Examples of implicit data collection include the following:

- Observing the items that a user views in an online store.
- Keeping a record of the items that a user purchases online.
- Obtaining a list of items that a user has listened to or watched on his/her computer.

The recommendation system compares the collected data to similar data collected from others and calculates a list of recommended items for the user. Recommendation systems are a useful alternative to search algorithms since they help users discover items they might not have found by themselves [20]. Collaborative filtering techniques are well known in enabling the prediction of user preferences in the recommendation systems. There are three major processes in the recommendation systems: object data collections and representations, similarity decisions, and recommendation computations. Collaborative filtering aims at finding the relationships among the new individual and the existing data in order to further determine the similarity and provide recommendations.

Defining the similarity is an important issue. How similar should two objects be in order to finalize the preference prediction? Similarity decisions are concluded differently by collaborative filtering techniques. For example, people that like and dislike movies in the same categories would be considered as the ones with similar behavior. The concept of the nearest-neighbor algorithm has been included in the

implementation of the recommendation systems. The challenge of conventional collaborative filtering algorithms is the scalability issue. Conventional algorithms explore the relationships among system users in large datasets. User data are dynamic, which means the data vary within a short time period. Current users may change their behavior patterns, and new users may enter the system at any moment. Millions of user data, which are called neighbors, are to be examined in real time in order to provide recommendations [3].

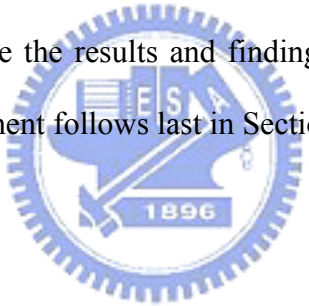
1.2. Motivation

With the gradually increasing use of IMDb (Internet Movie Database), MovieFinder.com, MovieLens and the likes, movie recommendation system are gaining more popularity and acceptance by the public. Such film recommendation systems make use of collaborative filtering technique to recommend films to users. The predictions are personalized to individual user's tastes, requiring users to rate films they have seen and generating recommendations based on patterns of similarity discovered in the user base. Traditional movie rental stores in Taiwan often make use of POS (Point of Sale) systems to take orders and perform simple queries. Such stores often query customer data, record an order, and forget it. With this business model, customers would usually walk into the store, browse through piles of unassociated titles and check out the items that he had in mind before coming to the store. Little or no interactions made with the customer often result in poor cross selling, lower customer satisfaction, decreases the customer’s loyalty on the company and the likelihood of the customer attaching to the company in a long run. What is unknown to the store owners are the hidden relations between their users and items on the shelf. With the aid of collaborative filtering, it is our aim in this research to build a movie recommendation system to mine these forgotten data so that similar items rented by

users with similar preference in the past can be processed to come up with other title recommendation to increase the overall sale.

1.3. Organization

The sections that follow are organized in the following fashion. A literature review of related researches and journals are touched. Collaborative filtering, clickstream collaborative filtering, and SOA architecture serve as the main focus subjects. Section 3 discusses our proposed methodology with regards to ER model design, UI design, collaborative filtering algorithm implemented, and the dynamic system that seamlessly incorporate all of the above. Section 4 details the experiment evaluation and analysis. In this section we go over the initial data and application setup, evaluation metrics and analyze the results and findings. A concluding remark along with future work for improvement follows last in Section 5.



2. Literature Review

2.1. Collaborative Filtering

Recommender systems have been developed to automate the recommendation process. Large-scale commercial applications of the recommender systems can be found at many ecommerce sites, such as Amazon, CDNow, Drugstore, and MovieFinder. These commercial systems recommend products to potential consumers based on previous transactions and feedback. They can enhance e-commerce sales by converting browsers to buyers, increasing cross-sales, and building customer loyalty. One of the most commonly-used and successful recommendation approaches is the collaborative filtering approach. Such approach works by first identifies a set of similar consumers based on past transaction and product feedback information and then makes a prediction based on the observed behavior of these similar consumers.

Collaborative filtering generates personalized recommendations by aggregating the experiences of similar users in the system. The key aspect of collaborative filtering lies in identification of consumers or users similar to the one who needs a recommendation. Cluster models, Bayesian Network models, and specialized association-rule algorithms, among other techniques, have been used for this identification purpose [7]. However, there remain important research questions in overcoming two fundamental challenges for collaborative filtering recommender systems: (1) scalability and (2) accuracy. The first challenge is to improve the scalability of the collaborative filtering algorithms. These algorithms are able to search tens of thousands of potential neighbors in real-time, but the demands of modern systems are to search tens of millions of potential neighbors. Further, existing algorithms have performance problems with individual users for whom the site has large amounts of information. The second challenge is to improve the quality of the

recommendations for the users. Users need recommendations they can trust to help them find items they will like. Users will "vote with their eyes closed" by refusing to use recommender systems that are not consistently accurate for them. In some ways these two challenges are in contrary, since the less time an algorithm spends searching for neighbors, the more scalable it will be, and the worse its quality. For this reason, it is important to treat the two challenges simultaneously so the solutions discovered are both useful and practical [11].

Most collaborative filtering methods fall into two categories: memory based algorithms and model based algorithms [2]. Memory based algorithms store users' rating in a training set. In the predication phase, they predict the ratings of an active user based on the corresponding ratings of the users in the training set that are similar to the active user. In contrast, model-based algorithms construct models in a precompiled manner that capture items with similar ratings from the training set and apply the precompiled model to predict the ratings for active users. Both types of approaches have been shown to be effective for collaborative filtering. In the subsections that follow, we elaborate the two approaches as well as an emerging simple, yet efficient deviation from model based scheme, slope one predictor.

2.1.1. Memory Based Collaborative Filtering

Early recommender systems were pure collaborative filters that computed pair wise similarities among users and recommended items according to a similarity weighted average. Breese et al. [2] refer to this class of algorithms as memory based algorithms. Memory based collaborative filtering algorithms are deterministic by nature. They rely on a database of previous users' preferences and perform similarity calculations on the database each time a new prediction is required [2]. The most

common representatives are nearest neighbor-based algorithms where a subset of users most similar to an active user is chosen and a weighted average of their preference ratings is used to estimate preferences of an active user on other items.

Memory based algorithms utilize the entire user-item database to generate predictions. These systems employ statistical techniques to find a set of users, known as neighbors, that have a history of agreeing with the target user (i.e., they either rate different items similarly or they tend to buy similar set of items). Once a neighborhood of users is formed, these systems use different algorithms to combine the preferences of neighbors to produce a prediction or top-N recommendation for the active user. The techniques, also known as K nearest neighbor or user-based collaborative filtering, are more popular and widely used in practice. Figure 1 shows the schematic diagram of the collaborative filtering process. Collaborative filtering algorithms represent the entire M x N user-item data as a ratings matrix, A. Each entry a_{ij} in A represents the rating of the i_{th} user on the j_{th} item. Each individual rating is within a numerical scale (e.g. 1 to 5) and it can as well be 0 indicating that the user has not yet rated that item.

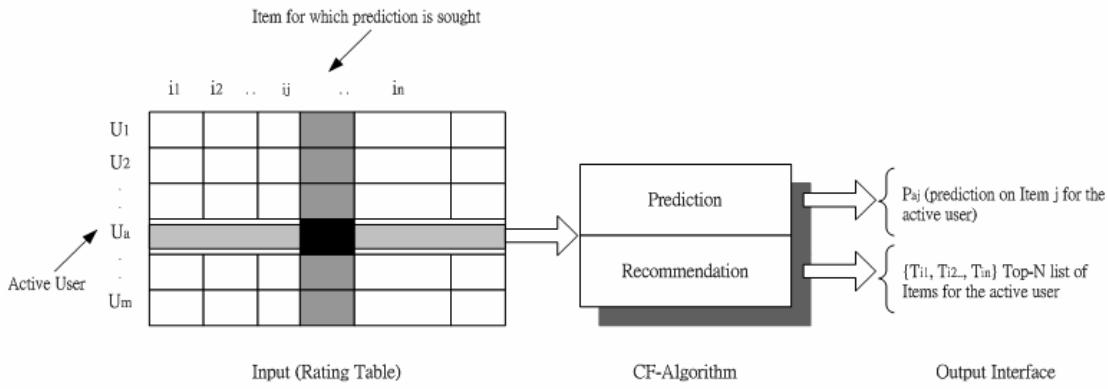
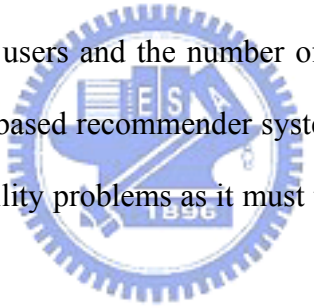


Figure 1: The Collaborative Filtering Process

User-based collaborative filtering systems have been very successful in past, but their widespread use has uncovered some potential challenges such as:

- Sparsity: In practice, many commercial recommender systems are used to evaluate large item sets (e.g., Amazon.com recommends books and CDnow.com recommends music albums). In these systems, even active users may have purchased well under 1% of the items (1% of 2 million books is 20,000 books). Accordingly, a recommender system based on nearest neighbor algorithms may be unable to make any item recommendations for a particular user. As a result the accuracy of recommendations may be poor.
- Scalability: Nearest neighbor algorithms require computation that grows linearly with both the number of users and the number of items. With millions of users and items, a typical web based recommender system running existing algorithms will suffer serious scalability problems as it must traverse through the entire data set.



2.1.2. Model Based Collaborative Filtering

Model based collaborative filtering algorithms provide item recommendation by first developing a model of user ratings. Algorithms in this category take a probabilistic approach and envision the collaborative filtering process as computing the expected value of a user prediction, given his/her ratings on other items. The model building process is performed by different machine learning algorithms such as Bayesian network, clustering, and rule-based approaches. The main idea here is to analyze the user-item representation matrix to identify relations between different items and then to use these relations to compute the prediction score for a given user-item pair. The intuition behind this approach is that a user would be interested in

purchasing items that are similar to the items the user liked earlier and would tend to avoid items that are similar to the items the user didn't like earlier. These techniques need not require identifying the neighborhood of similar users when a recommendation is requested; as a result they tend to produce much faster recommendations [11].

One critical step in the item-based collaborative filtering algorithm is to compute the similarity between items and then to select the most similar items. The basic idea in similarity computation between two items i and j is to first isolate the users who have rated both of these items and then to apply a similarity computation technique to determine the similarity $s_{i,j}$. Figure 2 illustrates this process. Here the matrix rows represent users and the columns represent items.

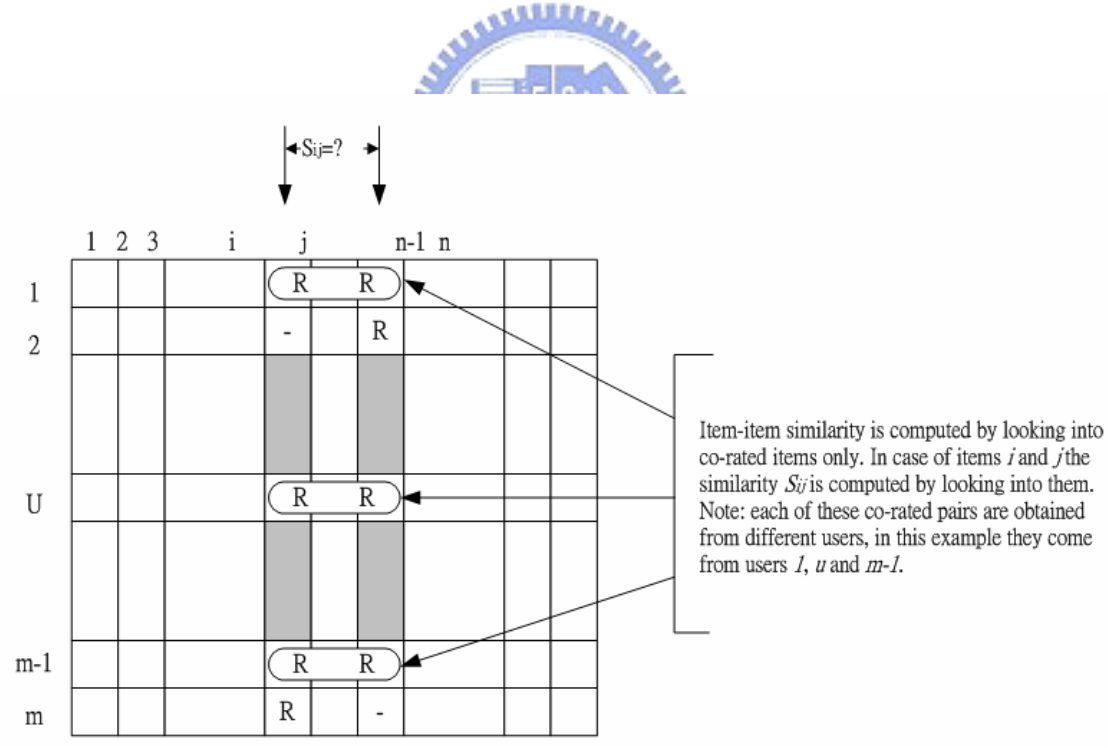


Figure 2: Isolation of the co-rated items and similarity computation

One fundamental difference between the similarity computation in user-based collaborative filtering and item-based collaborative filtering is that in case of

user-based collaborative filtering the similarity is computed along the rows of the matrix but in case of the item-based collaborative filtering, the similarity is computed along the columns (e.g. each pair in the co-rated set corresponds to a different user, see Figure 2). The similarity computation scheme is still correlation-based but the computation is performed on the item space rather than the user space. Typically, one would usually have a set of items that is static compared to the number of users that changes more often. The static nature of items has lead to the idea of precomputing the item similarities. In this scheme, one would retain only a small number of similar items. For each item j one compute the k most similar items, where k << n and record these item numbers and their similarities with j. K is termed as the model size. Based on this model building process, the prediction generation algorithm works as follows. For generating predictions for a user u on item i, our algorithm first retrieves the precompiled k most similar items corresponding to the target item i. Then it searches how many of those k items were purchased by the user u, based on this intersection, the prediction is computed using basic item-based collaborative filtering algorithm.

2.1.3. Slope One Predictor

Slope one predictor algorithm works on the intuitive principle of a "popularity differential" between items for users. In a pair wise fashion, one determines how much better one item is liked than another [9]. One way to measure this differential is simply to subtract the average rating of the two items. In turn, this difference can be used to predict another user's rating of one of those items, given their rating of the other. Consider two users A and B, two items I and J and Figure 3. User A gave item I a rating of 1, whereas user B gave it a rating of 2, while user A gave item J a rating of 1.5. We observe that item J is rated more than item I by 1.5−1 = 0.5 points, thus one could predict that user B will give item J a rating of 2+0.5 = 2.5. User B is called the

predictee user and item J the predictee item. Many such differentials exist in a training set for each unknown rating and one can take an average of these differentials.

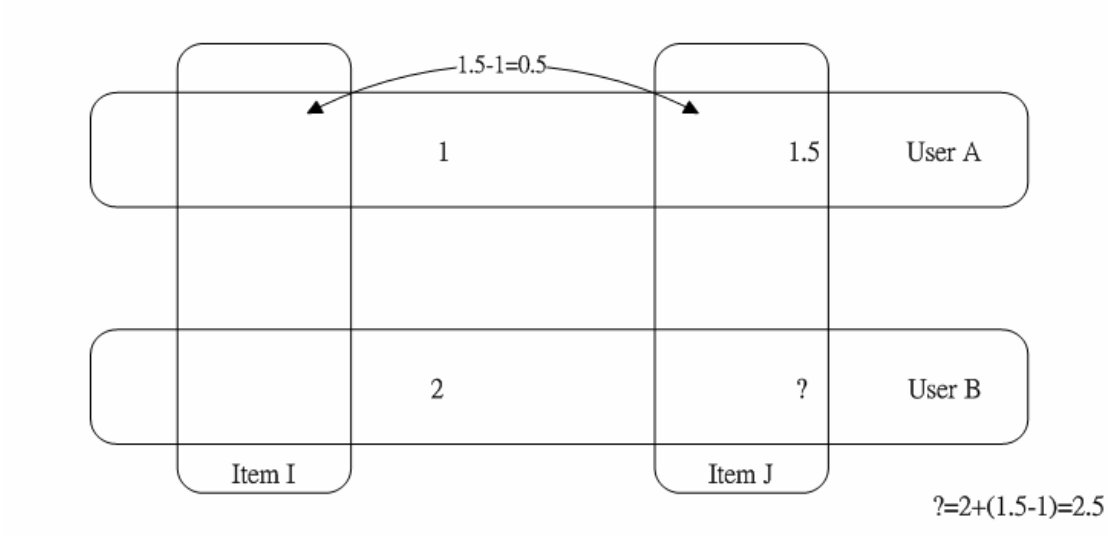


Figure 3: Basis of Slope One

2.2. Clickstream Collaborative Filtering

Clickstream based collaborative filtering (CCF), a kind of item-based collaborative filtering, has received much attention as a way of doing collaborative filtering recommendation in web navigation because user data is generally not available. Like other item-based collaborative filtering, CCF needs to adopt prediction models for efficient and effective recommendations of pages (or products) due to its vast amount of data stream to process: it trains the models offline and uses them in online recommendations [8]. Unlike other item-based CF recommendations (i.e., market basket data in EC), the sequence of pages is important for its recommendation quality due to the fact that the sequential structure is embedded through the hyperlinks in web pages. From browsing patterns user's product preference can be captured and then relevant pages or products can be recommended. The common models for CCF

recommendation are Markov models, sequential association rules, association rules and clustering. Markov models have been well positioned as a CCF recommendation model because of its high precision coming from the consideration of consecutive orders of preceding pages. In the subsections that follow, we examine Markov model and the clickstream concepts more in detail.

2.2.1. Markov Models

The problem of predicting a user's behavior on a web site has gained importance due to the rapid growth of the World Wide Web and the need to personalize and influence a user's browsing experience. Markov models and their variations have been found to be well suited for addressing this problem. Markov models have been used for studying and understanding stochastic processes and shown to be well suited for modeling and predicting a user's browsing behavior on a web site. In general, the input for these problems is the sequence of web pages accessed by a user and the goal is to build Markov models that can be used to predict the web page that the user will most likely access next [4]. Lower-order Markov models are not very accurate in predicting the user's browsing behavior, since these models do not look far into the past to correctly discriminate the different observed patterns. As a result, higher-order models are often used. Unfortunately, these higher-order models have a number of limitations associated with high state-space complexity, reduced coverage, and sometimes, even worse, overall prediction accuracy. One simple method to overcome some of these problems is to train varying order Markov models and use all of them during the prediction phase. Unfortunately, even though this approach was able to reduce the state-space complexity by up to an order of magnitude, it also reduced the prediction accuracy of the resulting models.

The act of a user browsing a web site is commonly modeled by observing the set of pages that he or she visits. This set of pages is referred to as a web session (S), and is represented by the sequence of pages $S = (P_1, P_2, \ldots, P_i)$ that were accessed. In this sequence, P_1 represents the first page that was accessed, P_2 the second, and so on. Given such a web session, the next-page prediction problem is that of predicting the web page that will be accessed by the user next. That is, given S, predict the next page P_{l+1} of the user's web session. This formulation can be used to solve many prediction problems that often arise in the web and e-commerce domain, such as whether or not a user will further explore the information associated with a particular topic, buy a particular product, view certain advertisements, or leave the web site, providing valuable clues about the user's interests and behavioral patterns. Such formulations are possible because most pages in a web site have a certain meaning associated with them and a user, by visiting a particular page, indicates his or her interest in that page. For example, e-commerce sites contain pages related to the products they sell (e.g., product description/specification pages, customer ratings and reviews, comparative pricing, etc.), pages related to order processing (e.g., shopping cart management, payment information, etc.), and pages related to various policies (e.g., return/exchange policy, privacy policy, etc.) [4].

2.2.2. Click-Stream Tree

A clickstream is the recording of what a computer user clicks on while web browsing or using a personal computer. As the user clicks anywhere in the tool, application or the webpage, the action is logged on a client or inside the web server. Clickstream analysis is generally used for analyzing employee productivity, software testing, market research and web activity analysis. Since the business world is quickly evolving into a state of e-commerce, analyzing the data of clients that visit a company

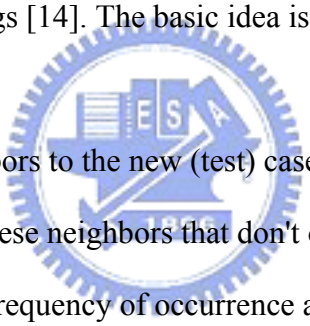
website is becoming a necessity in order to remain competitive [18]. This analysis can be used to generate an analysis of a user's clickstream while using a website to reveal usage patterns, which in turn gives a heightened understanding of customer behavior. This use of the analysis creates a user profile that aids in understanding the types of people that visit a company's website. Clickstream analysis can be used to predict whether a customer is likely to purchase from an e-commerce website. Clickstream analysis can also be used to improve customer satisfaction with the website and with the company itself. Both of these uses entail a huge business advantage. With the growing corporate knowledge of the importance of clickstream, the way that they are being monitored and used to build Business Intelligence is evolving.

The study of modeling and predicting a user's access on a web site has become more important. There are three steps in this process [6]. Since the data source is web server log data, the first step is to clean the data and prepare for mining the usage patterns. The second step is to extract usage patterns, and the third step is to build a predictive model based on the extracted usage patterns. The prediction step is the real-time processing of the model, which considers the active user session and makes recommendations based on the discovered patterns. Overall approach can be summarized as follows. The user sessions are clustered based on the similarity of the user sessions. When a request is received from an active user, a recommendation set consisting of three different pages that the user has not yet visited, is produced using the best matching user session. For the first two requests of an active user session all clusters are explored to and the one that best matches the active user session. For the remaining requests, the best matching user session is found by exploring the top-N clusters that have the highest N similarity values computed using the first two

requests of the active user session. The rest of the recommendations for the same active user session are made by using the top-N clusters.

2.2.3. Light Collaborative Filtering

Unlike explicit movie ratings, user browsing behavior logs are in the form of either a true visit or no-visit. Because the data are binary (1 or 0) encoded, and not ranked preferences on a numerical scale, efficient and lightweight schemes are described for compactly storing data, computing similarities between new and stored records, and making recommendations tailored to an individual. In the simplest scoring scheme, recommendations might be made based on a linear weighted combination of other people's browsing behavior logs [14]. The basic idea is as follows:



1. find the K nearest neighbors to the new (test) case
2. collect all attributes of these neighbors that don't occur in the test case
3. rank these attributes by frequency of occurrence among the K neighbors.

In measuring distance between cases, a score that measures similarity is computed; the higher the score, the greater the similarity. The pseudo code is shown below. It follows the 3 steps listed earlier and computes as the aforementioned steps.

```
Input: C {new case represented by M attributes C(1), ... C(m)},
       D {Historical data of n cases, D1 ... Dn}
Output: A {Ranked List of attributes}

Begin
  for j = 1 .. m do
    df = Number of case in D where attribute j appears;
    pv(j) = 1 + 1/df;
```

```
    done
    score(Di) = 0 for i=1,n
    rank(j) = 0 for j=1,m
    for j = 1 .. m do
      if (C(j) == 0) continue;
      //examine only attributes that are positive for the new case
      for i = 1 .. n do
        if (Di(j) == 0) continue;
        //score a case only if it shares an attribute with new case
        score(Di) += pv(j);
      done
    T = select K cases with highest scores in D;
    for j = 1 to m do
      if (C(j) == 1) continue;
      //examine only attributes that are not positive for new case
      for i = 1 to k do
        //increase count of those attributes that are in top-K cases
        if (Ti(j) == 1)
          rank(j) += 1;
      done
    done
    Output = small subset of attributes with highest rank(j);
End
```

2.3. Dynamic Data Mining

A Dynamic Data Mining Process (DDMP) system based on Service-Oriented Architecture (SOA) is especially useful when different activity of data mining is applied. Each activity in data mining process is viewed as a web service operated on internet. Depending on recommendation outlined above, the web service can be dynamically linked using Business Process Execution Language (BPEL). If the recommendation functions can be selected, combined, and interchanged dynamically, it will be much more flexible for small and medium enterprises to adopt it [13]. For an

effective knowledge discovery in database process (KDD), several technologies have to work together. First, data preprocessing including data cleaning, integration, transformation, and reduction should be applied. The quality of data preprocessing will significantly affect the mining result. Second, statistical analysis and machine-learning techniques are applied to those quality data to extract patterns and to predict trends. A typical KDD process including data preprocessing phase, data mining phase, and analysis phase is illustrated as Figure 4.

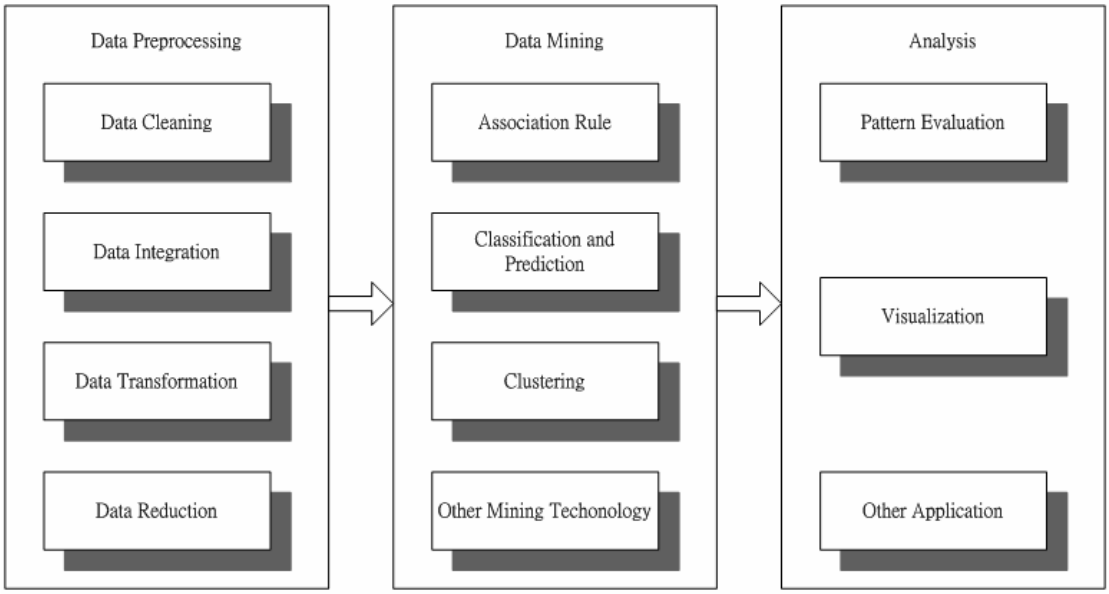


Figure 4: Knowledge Discover in Database Process

Dynamic Data Mining Process (DDMP) system based on SOA [13] proposed that each activity in data mining process be viewed as a web service operated on internet. The Web Services provide functions of data preprocessing, algorithm calculations, data mining tasks, or visualization analysis. Those Web Services are dynamically linked using BPEL to construct a desired data mining process. The building blocks of DDMP, SOA, BPEL are examined next.

2.3.1. SOA

Service oriented architecture (SOA) is an evolution of distributed computing based on the request/reply design paradigm for synchronous and asynchronous applications. An application's business logic or individual functions are modularized and presented as services for consumer/client applications. What's key to these services is their loosely coupled nature, e.g., the service interface is independent of the implementation. Application developers or system integrators can build applications by integrating one or more services without knowing the services' underlying implementations or physical locations [16]. For example, a service can be implemented either in .Net in the states or J2EE in Taiwan, and the application consuming the service can be on a different platform or language.

Service oriented architectures have the following key characteristics:

- SOA services have self-describing interfaces in platform independent XML documents. Web Services Description Language (WSDL) is the standard used to describe the services.
- SOA services communicate with messages formally defined via XML Schema (also called XSD). Communication among consumers and providers or services typically happens in heterogeneous environments, with little or no knowledge about the provider. Messages between services can be viewed as key business documents processed in an enterprise.
- SOA services are maintained in the enterprise by a registry that acts as a directory listing. Applications can look up the services in the registry and invoke the service. Universal Description, Definition, and Integration (UDDI) is the standard used for service registry.

- Each SOA service has a quality of service (QoS) associated with it. Some of the key QoS elements are security requirements, such as authentication and authorization, reliable messaging, and policies regarding who can invoke services.

2.3.2. BPEL

The Business Process Execution Language (BPEL) is a programming language for specifying business processes that involve Web Services. BPEL is especially good at supporting long running conversations with business partners. Even before the standard is formally released, it is becoming clear that BPEL will be the more widely-adopted standard for business processes involving Web Services. BPEL is geared towards programming in the large, which supports the logic of business processes. These business processes are self-contained applications that use Web Services as activities that implement business functions [17]. BPEL does not try to be a general-purpose programming language. Instead, it is assumed that BPEL will be combined with other languages which are used to implement business functions (programming in the small).

Programming in the large generally refers to the high-level state transition interactions of a process—BPEL refers to this concept as an Abstract Process. A BPEL Abstract Process represents a set of publicly observable behaviors in a standardized fashion. An Abstract Process includes information such as when to wait for messages, when to send messages, when to compensate for failed transactions, etc. Programming in the small, in contrast, deals with short-lived programmatic behavior, often executed as a single transaction and involving access to local logic and

resources such as files, databases, etc. BPEL's development came out of the notion that programming in the large and programming in the small required different types of languages.

3. Methodology

In this research, we propose a dynamic collaborative filtering system that makes use of UserCorrelation, ItemCorrelation, SlopeOne Predictor, and Clickstream collaborative filtering model to make use of both user navigation patterns along side historical purchased items for users with similar buying behavior. Clickstream collaborative filtering is taken into consideration in that often user may just appear to be browsing through the library of movie titles without target intent of purchase set in mind. He or she may tend to browse a site randomly on a routine basis similar to what most people would do everyday by checking eBay, Yahoo, or Amazon and the likes just for fun. Without a pre-determined mind set of purchasing any item, items recommended via traditional collaborative filtering could sometimes prove ambiguous to the user. We believe that in this scenario, a CCF approach would have been more accurate and up to the point. As indicated by Deshpande [4] the probability of visiting a page p_i does not depend on all the pages in the web session, but only on a small set of k preceding pages, where k << total number of pages in the active session. Using the clickstream process assumption, being able to predict the potential interests (or disinterests) of a user while he or she is still undecided, can help in taking actions to affect their behavior.

In cases where users have committed on buying a particular item (e.g. added an item to shopping cart), item to item CF will come into the play. This is similar to Amazon's book recommendation when one has committed on purchasing a book. At this stage, user has already indicated interest in buying, thus the turn out ratio for he or she to buy a closely related item (e.g. other items purchased by users when they

purchased this item) would have been higher. Because the algorithm recommends highly correlated similar items, recommendation quality is excellent [11]. Item to item CF has also proved to be highly scalable and up to the performance challenge. The key to item-to-item collaborative filtering's scalability and performance is that it creates the expensive similar-items table offline. The algorithm's online component looking up similar items for the user's purchases and ratings scales independently of the catalog size or the total number of customers; it is dependent only on how many titles the user has purchased or rated. Thus, the algorithm is fast even for extremely large data sets [10].

To adopt the aforementioned techniques, Dynamic Collaborative Filtering system based on SOA [13] will be approached. With this approach, each recommendation technique will be viewed as a web service. The data recommendation service will provide Clickstream Collaborative Filtering along with User-Based Collaborative Filtering, Item-Based Collaborative Filtering, and SlopeOne Predictor. The goal of using Web Services in this is to achieve universal interoperability between applications by using web standards. Web Services use a loosely coupled integration model to allow flexible integration of heterogeneous systems in a variety of domains including business-to-consumer, business-to-business and enterprise application integration [21]. The basic specifications defined the Web Services operations: Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI). SOAP defines an XML messaging protocol for basic service interoperability. WSDL introduces a common grammar for describing services. UDDI provides the infrastructure required to publish and discover services in a systematic way. In metaphor, UDDI acts as directory listing service like yellow pages, WSDL

acts as an entry in the listing describing access information such as expected inputs and outputs, and SOAP acts as telephone line connecting the end points. Together, these specifications allow data mining applications to find each other and interact following a loosely coupled, platform independent model [11]. The framework of the proposed Dynamic Collaborative Filtering system can be depicted as Figure 5.

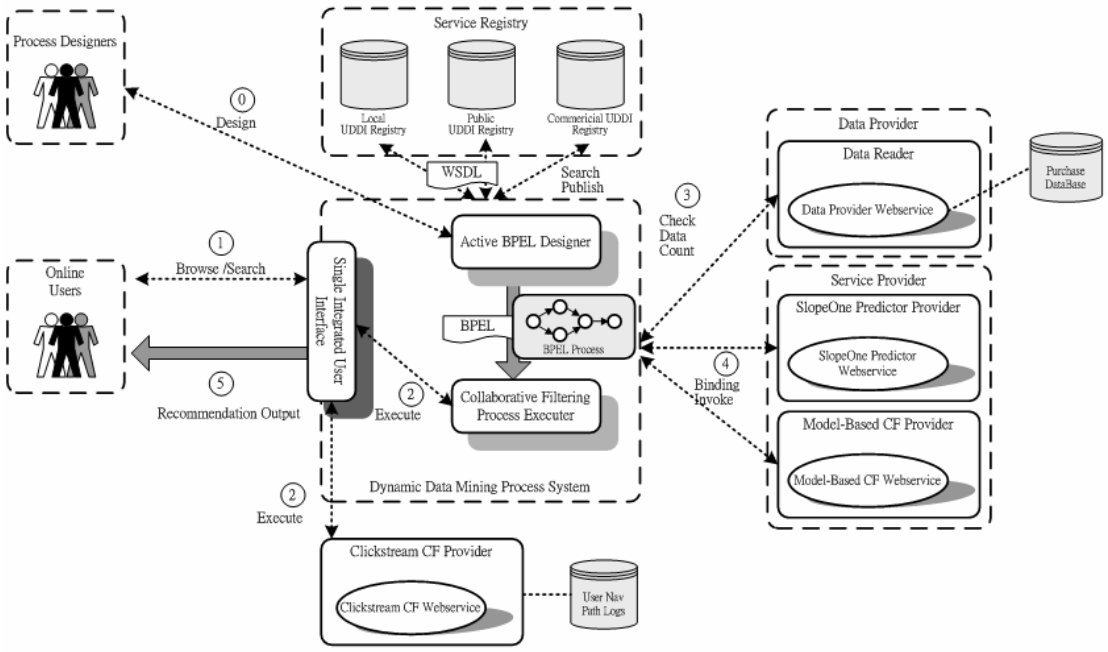
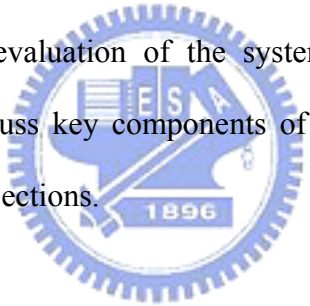


Figure 5: Dynamic Collaborative Filtering Architecture

The data source of our dynamic system will be that of a video store currently running a POS system for its daily rental operations. The legacy system keeps its transaction records in a DOS-based application. Since the legacy system doesn't require customers to explicitly give a rating on the movie each time he or she rents one, we will not be able to approach legacy data with the prescribed item-based CF. A. Schein [12] have termed this, the cold-start problem, where recommendations are required for items that no one (in our data set) has yet rated. Pure collaborative filtering cannot help in a cold-start setting, since no user preference information is

available to form any basis for recommendations. A work around is to approach this via an implicit rating deviation. Implicit rating prediction refers to prediction of data such as purchase history; a purchase is not necessarily an indication of satisfaction, but a purchase can be treated as an indication of some implicit need or desire for an item. For legacy data we assign users and movies that had been associated in the past an implicit value of 1, a task that is analogous to predicting a customer purchase. Implicit rating prediction is more appropriate for domains where explicit rating information is not available. To evaluate our dynamic model, we will be reviewing the key decisions in evaluating collaborative filtering recommender systems: the user tasks being evaluated, the types of analysis and datasets being used, the ways in which prediction quality is measured, the evaluation of prediction attributes other than quality, and the user-based evaluation of the system as a whole as outlined by Deshpande [4]. We now discuss key components of the proposed dynamic system approach in the following subsections.



3.1. ER Model Design

The proposed ER model for the persistent layer is shown in Figure 6. Since we do not have explicit ratings from the migrated legacy data, an "IMPLICITRATING" column is used to associate the past rental records where 1 indicates a rent. As our dynamic system is built and evolve, the system will be asking future users to explicitly give a rating on a scale of 1 to 5. When enough explicit ratings have been accumulated passed a preset threshold (e.g. 5 explicit ratings for the particular movie), the dynamic system can switch to the explicit rating scheme that makes use of traditional item-based collaborative filtering methods. The implicit rating scheme

serves a great aid in situations of a “cold start” and in places where explicit ratings have not yet met the threshold.

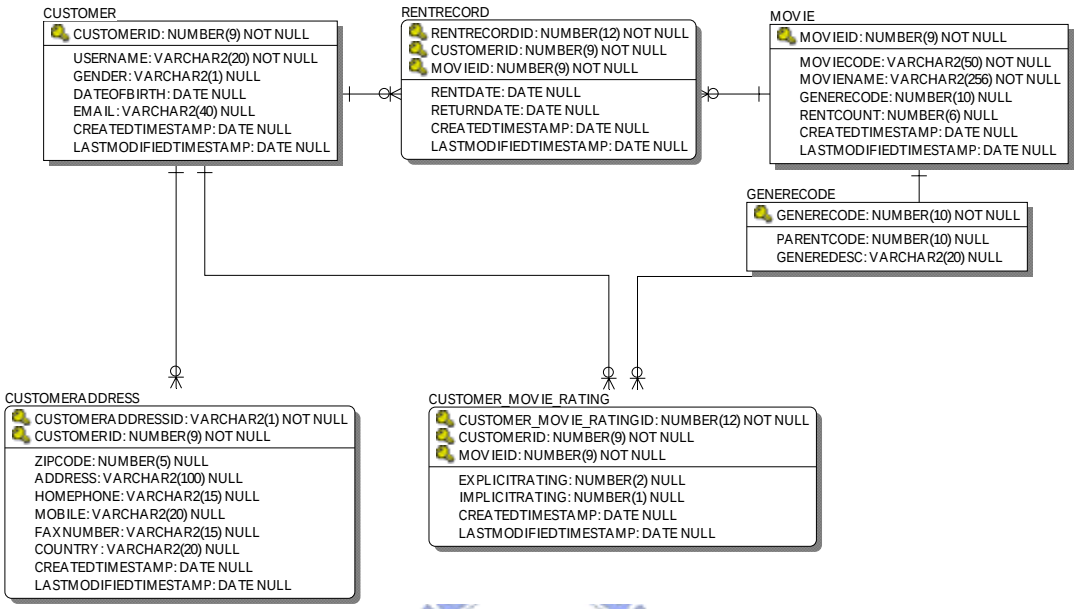


Figure 6: ER Model

3.2. Class Diagram

The implementation involve the components as follow: (1) Recommender, (2) DataModel, (3) UserCorrelation, (4) ItemCorrelation, (5) Userneighbourhood. A Recommender is the core abstraction in our implementation. Given a DataModel, it can produce recommendations. Applications will either use the GenericUserBasedRecommender implementation or GenericItemBasedRecommender. A DataModel is an encapsulation over user preferences. Our implementation draws this data from a database. An alternative to read the data from flat files is also provided. Along with DataModel, we use User, Item and Preference abstractions to represent the users, items, and preferences for those items in the recommendation

engine. A UserCorrelation defines a notion of similarity between two Users. These are attached to a Neighborhood implementation for finding the closest neighbor's similarity scores. In a user-based recommender, recommendations are produced by finding a "neighborhood" of similar users near a given user. A UserNeighborhood defines a means of determining that neighborhood — for example, nearest 10 users. ItemCorrelations are analogous, but find similarity between Items. As depicted in Figure 2 previously, ItemCorrelations compute the similarity scores along item columns in contrast to UserCorrelation where the computation takes place along user columns. The DataModel class provides function calls to retrieve such information as users, preference for the item, and items. A parent DataModel interface defines the common functions, whereas the implementing FileDataModel and AbstractJDBCDataModel provide functions to encapsulate data from flat text file and database. The Item, User, Preference encapsulate the retrieved items, users, and preference for the item respectively. The overall DataModel class diagram is shown below in Figure 7.

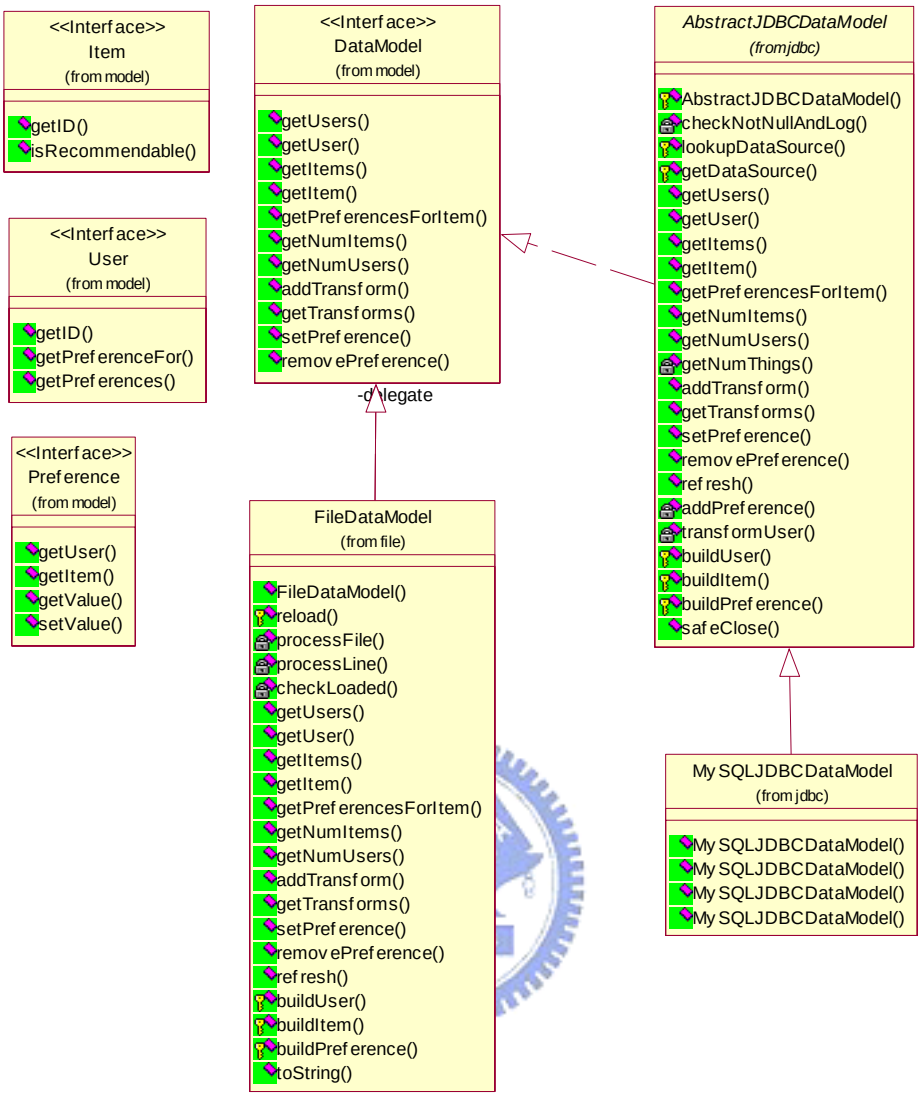


Figure 7: DataModel Class Diagram

UserCorrelation and ItemCorrelation are similar as they compute similarity scores based on the ratings given by users. Both approaches use scoring algorithms to compute similarity. In our approach, we will use the two widely-adopted approaches, Cosine-based Similarity and Correlation-based Similarity (Pearson Correlation) to score the similarity between items and users. Similarities between items are calculated along the data columns, whereas similarities between users are calculated along data

rows as discussed in section 2.1.2. The overall Correlation class diagram is shown below in Figure 8.

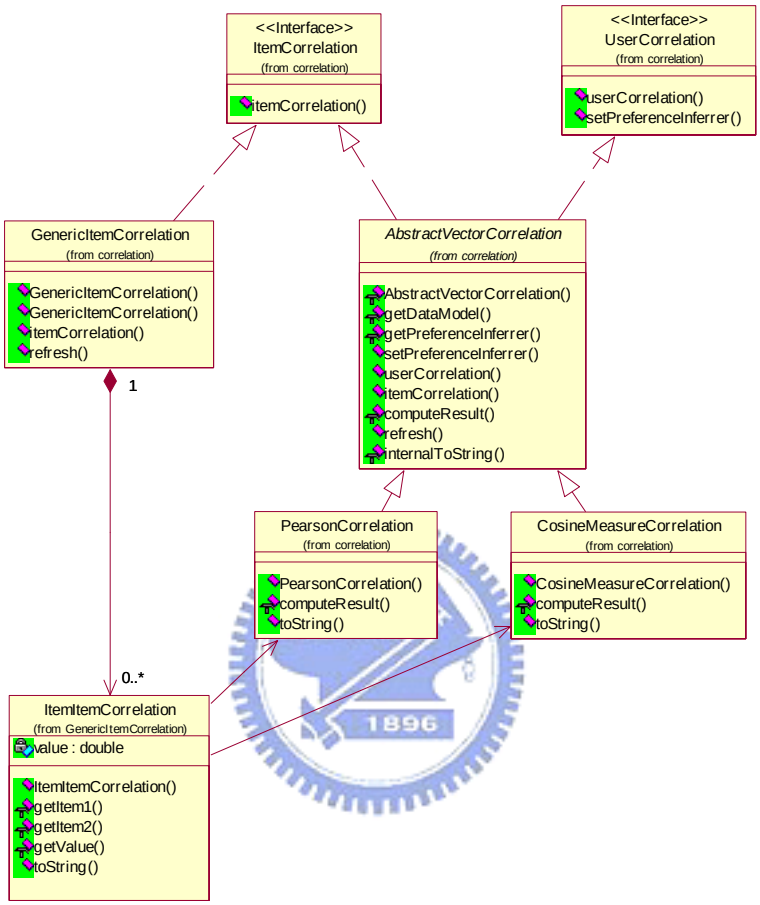


Figure 8: Correlation Class Diagram

Recommender implementations make use the aforementioned correlation implementations to aggregate the similarity scores calculated and select the top score items or closest "neighbors" for generating recommendation based on UserCorrelation or ItemCorrelation. GenericUserBasedRecommender does not reference UserCorrelation directly, instead it deters reference to NearestNUserNeighborhood that caches the calculated UserCorrelation as a measure of neighborhood similarity. The higher the UserCorrelation score, the closer the

neighbor. In contrast, GenericItemBasedRecommender makes direct reference to ItemCorrelation. Since ItemCorrelation captures the ItemItemCorrelation between two items in a static manner, the collection of ItemItemCorrelations are pre-computed offline to decouple the recommendation process and computation process. Due to this static behavior, we deter the ItemItemCorrelations computation in an offline batch. The overall Recommender class diagram is shown below in Figure 9.

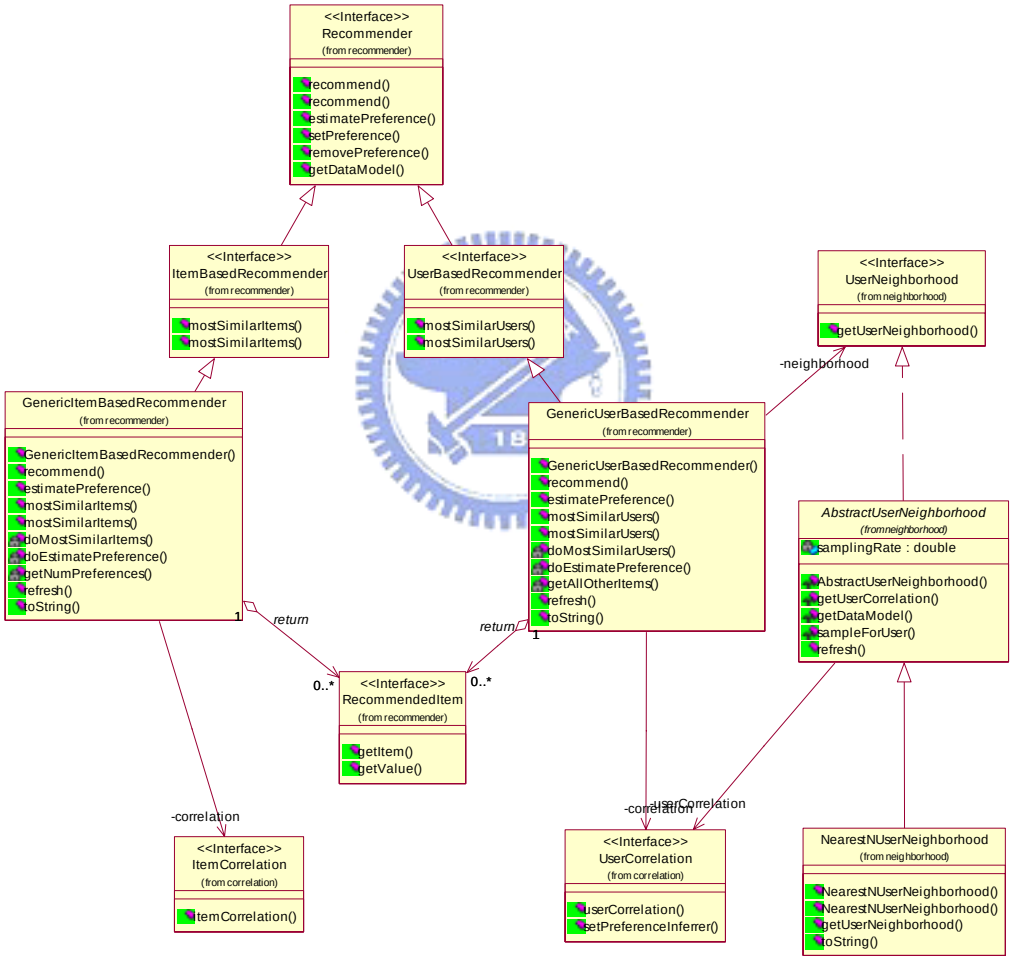


Figure 9: Recommender Class Diagram

The overall package interaction abstraction is illustrated in Figure 10. In short, recommender package makes use of neighborhood package to generate

recommendations based on the applied model based on UserCorrelation; whereas ItemCorrelation is referenced directly by the recommender package.

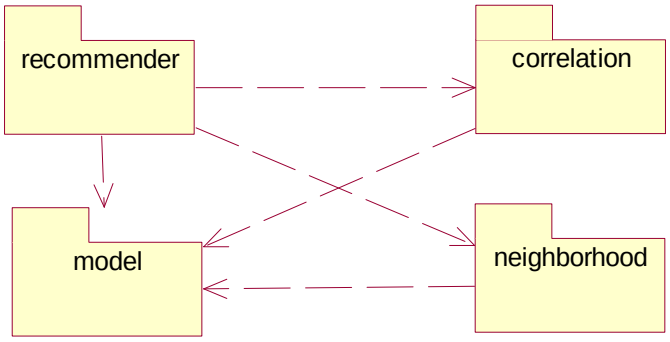


Figure 10: Package Interaction Diagram

3.3. Collaborative Filtering

3.3.1. Item-Based Collaborative Filtering

There are a number of different ways to compute the similarity between items. B. Sarwar [11] presented three such methods. They are cosine-based similarity, correlation-based similarity and adjusted-cosine similarity.

Cosine-based Similarity

In this case, two items are thought of as two vectors in the m dimensional user-space. The similarity between them is measured by computing the cosine of the angle between these two vectors. Formally, in the M x N ratings matrix in Figure 2, similarity between items i and j, denoted by $sim(i, j)$ is given by

$$sim(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\left\|\vec{i}\right\|_2 * \left\|\vec{j}\right\|_2}$$

Correlation-based Similarity

In this case, similarity between two items *i* and *j* is measured by computing the Pearson-r correlation $corr_{i,j}$. To make the correlation computation accurate we must first isolate the co-rated cases (i.e., cases where the users rated both *i* and *j*) as shown in Figure 2. Let the set of users who both rated *i* and *j* are denoted by *U* then the correlation similarity is given by

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

Weighted Sum

The next step of our collaborative filtering system is to generate the output in terms of prediction. Once we isolate the set of most similar items based on the aforementioned similarity measures, the next step is to look into the target user's ratings and use a technique to obtain predictions. Here we consider the Weight Sum approach. This technique computes the prediction on an item *i* for a user *u* by computing the sum of the ratings given by the user on the items similar to *i*. Each ratings is weighted by the corresponding similarity *sim(i, j)* between items *i* and *j*. The prediction $P_{u,i}$ is simply in the form of

$$P_{u,i} = \frac{\sum_{allsimilaritems} (s_{i,N} * R_{u,N})}{\sum_{allsimilaritems} (|s_{i,N}|)}$$

Basically this approach tries to capture how the active user rates the similar items. The weighted sum is scaled by the sum of the similarity terms to make sure the prediction is within the predefined range.

3.3.2. Slope One Predictor

The Slope One scheme takes into account both information from other users who rated the same item (like the Adjusted Cosine Similarity) and from the other items rated by the same user (like the Per User Average). However, the schemes also rely on data points that fall neither in the user array nor in the item array (e.g. user *A*'s rating of item *I* in Figure 3), but are nevertheless important information for rating prediction. Much of the strength of the approach comes from data that is not factored in. Specifically, only those ratings by users who have rated some common item with the predictee user and only those ratings of items that the predictee user has also rated enter into the prediction of ratings under slope one schemes. We now elaborate the slope one derivation.

The notation denoting the schemes are as follows. The ratings from a given user, called an evaluation, are represented as an incomplete array u, where u_i is the rating of the user gives to item i. The subset of the set of items consisting of all those items which are rated in u is $S(u)$. The set of all evaluations in the training set is x. The number of elements in a set S is $card(S)$. The average of ratings in an evaluation u is denoted $\bar{u}$. The set $S_i(x)$ is the set of all evaluations $u \in x$ such that they contain item i ($i \in S(u)$). Given a training set x, and any two items j and i with ratings u_j and u_i respectively in some user evaluation u (annotated as $u \in S_{j,i}(x)$), we consider the average deviation of item i with respect to item j as:

$$dev_{j,i} = \sum_{u \in S_{j,i}(x)} \frac{u_j - u_i}{card(S_{j,i}(x))}$$

Given that $dev_{j,i} + u_i$ is a prediction for u_j given u_i, a reasonable predictor might be the average of all such predictions:

$$P(u)_j = \frac{1}{card(R_j)} \sum_{i \in R_j} (dev_{j,i} + u_i)$$

We can simplify the prediction formula for the SLOPE ONE scheme to:

$$P^{S1}(u)_j = \bar{u} + \frac{1}{card(R_j)} \sum_{i \in R_j} dev_{j,i}$$

Note that the implementation of Slope One doesn't depend on how the user rated individual items, but only on the user's average rating and crucially on which items the user has rated. This will act as another item-based CF provider as depicted in Figure 5.

3.3.3. Light Collaborative Filtering

Here we define the target active user session as a collection of page ids associated with Web pages. The page ids are analogous to the new case's attributes as discussed in Section 2.2.3. Each new case represented by m page ids. For each training case, count the number of positive page ids in common with the new case. The new case's collection of page ids is represented by C(1) to C(m). Historical cases are resented by D1 to Dn. For each new case's page ids (e.g 1 to m), with reference to the pseudo code defined in section 2.2.3 we compute the apriori predictive value as 1 plus the inverse frequency of the total attribute occurrences. This function measures the apriori predictive value of the particular page id and is computed once at the start. We then traverse through each of the historical cases to check if the historical case's page ids

are positive with the corresponding new case's page ids. For each historical case's page ids conforming to test page id, the computed apriori predictive value is accumulated for that historical case. The top K cases are then selected to further rank the page ids that have not occurred in the new case. The rank is based on the total occurrences of the page id in the returned top K cases.

3.4. Clickstream Tree

The novelty of this approach proposed by Ş. Gündüz [5] lies in the method by which the similarity of user sessions are computed and how they are clustered. Each user session is a sequence of Web pages visited by a single user with a unique session number. Each clickstream tree has a root node, which is labeled as "null". Each node except the root node of the clickstream tree consists of three fields: data, count and next node. Data field consists of page number and the normalized time information of that page. Count field registers the number of sessions represented by the portion of the path arriving to that node. Next node links to the next node in the clickstream tree that has the same data field or null if there is any node with the same data field. Each clickstream tree has a data table, which consists of two fields: data field and first node that links to the first node in the clickstream tree that has the data field. The tree for each cluster is constructed by applying the algorithm given in below.

Create a root node of a clickstream tree, and label it as null
index ← 0
while index ≤ number of sessions in the cluster **do**
  active_session ← t_{index}
  m ← 0
  current_node ← root node of the clickstream tree
  while m ≤ active_session.length **do**
    active_date ← { $p^{m}_{t_{index}}$ } – { $T^{m}_{t_{index}}$ }

```
      if there is a child of current_node with the data field then
        child.count++
        current_node ← child
      else
        create a child node of the current_node
        child.data = active_data
        child.count = 1
            current_noode ← child
      end if
      m++
    end while
    index++
  end while
```

The children of each node in the clickstream tree are ordered in the count descending order such that a child node with bigger count is closer to its parent node. Upon constructing the clickstream tree, we tweaked the original idea of returning the most frequent visited path by feeding the frequent path to a binary data recommendation engine, Light Collaborative Filtering as discussed previously in section 3.3.3. We make use of Light CF's scoring function to score each case (e.g. clickstream tree's computed frequent paths) with their apriori predictive value respectively. The higher the apriori score, the higher the rank. The recommendation is then based on the top-most ranked cases.

3.5. Dynamic Collaborative Filtering

In this section we discuss the dynamic collaborative filtering approach under Service Oriented Architecture (SOA). The core to SOA lies in Business Process Execution Language (BPEL). BPEL orchestrates at the time of execution, which of the aforementioned collaborative filtering algorithm is to be processed. The business rules defined in BPEL and the actual implementations are loosely coupled in that

changes in BPEL do not affect the actual implementations. As business rules evolve, we can change the orchestration defined in BPEL to reflect such. Should there be new algorithms developed in the future, we can plug-in the new implementation into the "enterprise service bus." We will use ActiveBPEL Designer [15] to design our dynamic model. The proposed dynamic collaborative filtering process is designed to distinguish logged in users to check if enough explicit ratings have been observed so that a choice among the collaborative filtering algorithms can be made. Concurrently, "ClickStreamTree" implementation for recommending next "most likely to access" Web pages associated with each movie based on matching the current access path to the most similar stored access path as defined in clickstream tree is also being processed. In cases where users have logged in, BPEL first checks if explicit ratings have met the predefine threshold. If rating counts exceeded a predefined threshold (e.g. 100,000) where the real time computation as those defined in SlopeOne Predictor could significantly affect the recommendation efficiency, Item-Based Collaborative Filtering recommendation service will be called upon to alleviate the recommendation task and balance the scalability and prediction accuracy. Note that ClickStreamTree service is independent of BPEL execution in that it computes the next most likely to be viewed Web pages in a parallel process. The BPEL orchestrated process with the aforementioned rule sets designed by ActiveBPEL Designer is given below in Figure 11.

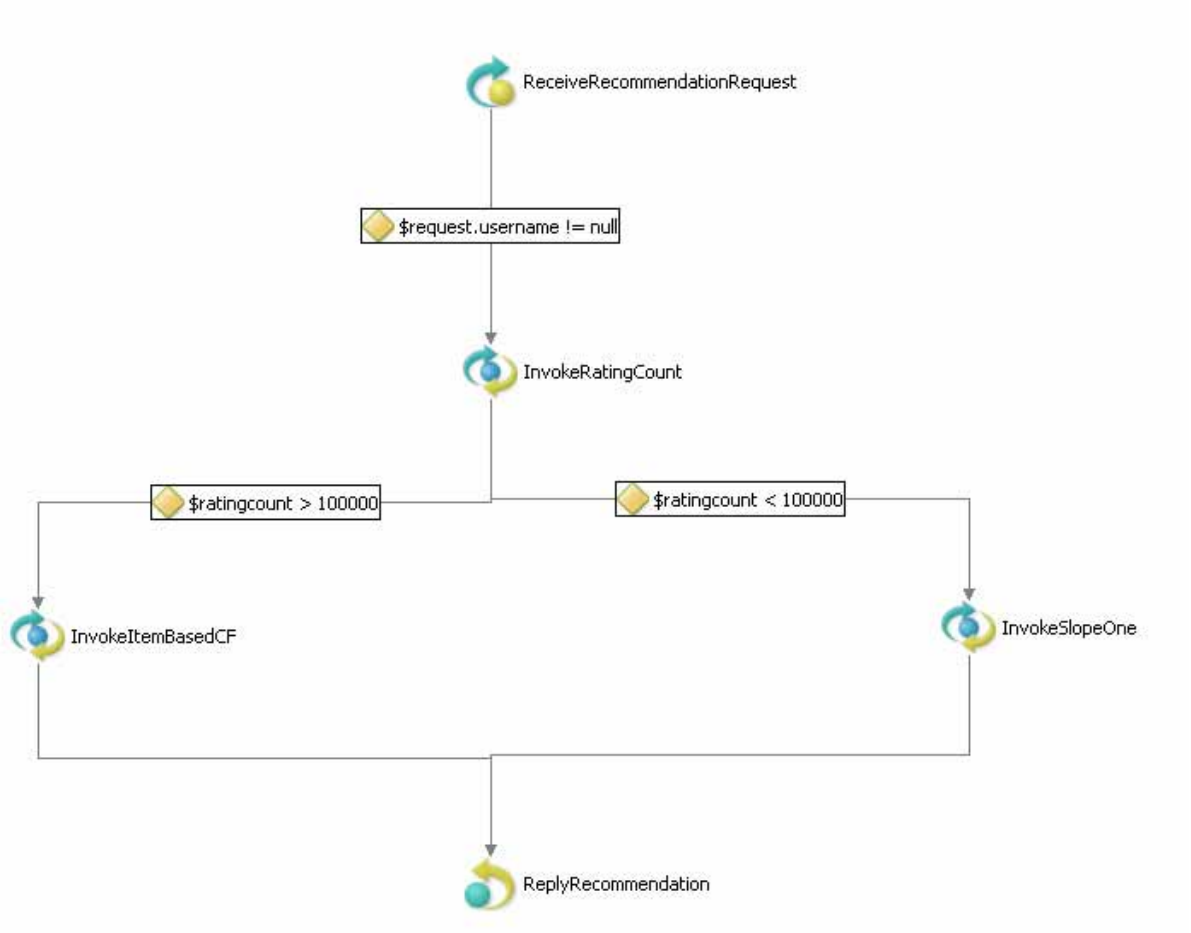


Figure 11: Dynamic Collaborative Filtering Process

4. Experimental Evaluation

Recommender systems research has used several types of measures for evaluating the quality of a recommender system. They can be mainly categorized into two classes:

- *Statistical accuracy metrics* evaluate the accuracy of a system by comparing the numerical recommendation scores against the actual user ratings for the user-item pairs in the test dataset. Mean Absolute Error (MAE) between ratings and predictions is a widely used metric. MAE is a measure of the deviation of recommendations from their true user-specified values. For each ratings prediction and actual pair $<p_i, q_i>$, this metric treats the absolute error between them, e.g., $|p_i - q_i|$ equally. The MAE is computed by first summing these absolute errors of the N corresponding ratings-prediction pairs and then computing the average. Formally,

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N}$$

  The lower the MAE, the more accurately the recommendation engine predicts user ratings.

- Decision support accuracy metrics evaluate how effective a prediction engine is at helping a user select high quality items from the set of all items. These metrics assume the prediction process as a binary operation either items are predicted (good) or not (bad). With this observation, whether an item has a prediction

score of 1:5 or 2:5 on a five-point scale is irrelevant if the user only chooses to consider predictions of 4 or higher. The most commonly used decision support accuracy metrics are reversal rate, weighted errors and ROC sensitivity.

We will use MAE as our choice of evaluation metric to report prediction experiments because it is most commonly used and easiest to interpret directly.

4.1. Data Source

We will be drawing customer rating data from CUSTOMER_MOVIE_RATING table as depicted in Figure 6. Since we do not have explicit ratings available from our legacy exported data, we will be obtaining our alternative "1 Million MovieLens Dataset" from GroupLens Research [19]. Note the 2 columns of EXPLICITRATING and IMPLICITRATING in the table. Since a rent record from legacy data indicates a purchase, in hindsight, we will be assigning a value of 1 in the IMPLICITRATING column and leave the EXPLICITRATING empty. For the MovieLens data, the actual ratings will be inserted into the EXPLICITRATING column, and since an explicit rating is most likely associated with a purchase, we'll be defaulting the IMPLICITRATING column for such data a value of 1. A snapshot of the imported data in the database is depicted in the Table 1.

Table 1: Imported Data Snapshot

CUSTOMER_MOVIE_RATINGID	CUSTOMERID	MOVIEID	EXPLICITRATING	IMPLICITRATING
238	4	1036	4	1
2115	18	1036	3	1
2404	19	1036	5	1
2713	22	1036	4	1
3008	23	1036	5	1
4001	29	1036	5	1
5213	36	1036	4	1
5832	42	1036	4	1
6971	48	1036	4	1

4.2. Application Setup

The application requires J2SE 5.0 or above to run. The bundled java web archive file (WAR) requires Servlet 2.3 or above containers such as Apache Tomcat. Copy the WAR file to Tomcat's webapps directory, and the start Tomcat by executing the startup.bat command in terminal window. Recommendations are automatically retrieved with reference to three controlling attributes: "userID", "movieID" and "howMany". UserID and movieID attributes are automatically checked by the system, whereas howMany attribute is preset in a global web context fashion. The "userID" denotes which user id one is seeking recommendation for, and "howMany" denotes how many recommendations the application should return from the computation. The movieID associated with the web page is then passed along with the userID and howMany attributes to the BPEL engine. BPEL engine takes charge in checking the rating counts for the particular movie. Depending on the rating counts for the particular movie, the dependent CF scheme as illustrated in Figure 11 will be chosen to generate recommendations. Upon receiving the web server renders the recommendation at the lower part of the target page.

4.3. Experiment Result

The data were randomly divided as 90% for training, and 10% for testing purposes. With the trained dataset, the correlation scores thereby generated were used to predict the ratings in the test dataset. The actual rating is compared with the estimated rating generated by the recommendation engine. MAE is then calculated to be the average of the actual and estimated differentials. Experiments were run with the different collaborative filtering scheme aforementioned in section 3.3. We've divided the evaluation into 3 parts: (1) initial run time consumption, (2) subsequent runs time consumption, and (3) MAE (Mean Absolute Error). The Time consumptions were divided into initial run and subsequent runs to illustrate Item-Based Collaborative Filtering is magnitudes higher in Time consumption during startup (see Chart 1), but is more efficient in subsequent runs (see Chart 2). The reason for this is that during startup phase, Item-Based CF scans through the entire database and compute each item pair's correlation score. Nevertheless, since relationships between item pairs are rather static, this calculation can be pre-computed in a separate offline batch Process. The computed similarity scores can then be stored in cache for later online Item-Based CF's quick reference. The initial run of compared collaborative filtering scheme in milliseconds is shown next in Chart 1, the associated data sheet is shown in Table 2.

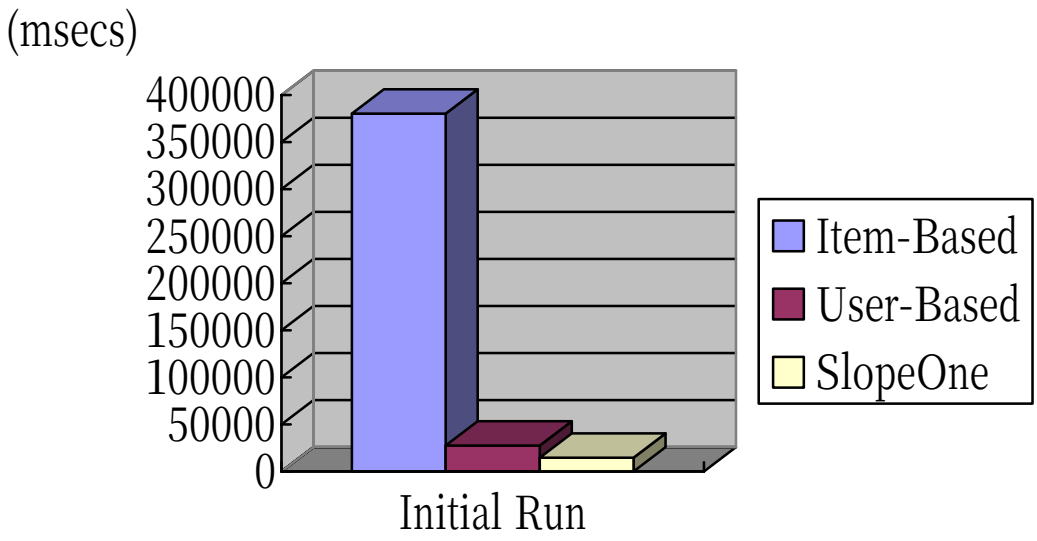


Chart 1: Initial Run Time Consumed

Table 2: Initial Run Time Consumed Data Sheet

Collaboration Scheme	Time (milli-seconds)
ItemCorrelation(Pearson) Initial Run	380328
UserCorrelation(Cosine) Initial Run	27453
SlopeOne Initial Run	14610

The subsequent runs of compared collaborative filtering scheme in milliseconds is shown next in Chart 2, the associated data sheet is shown in Table 3.

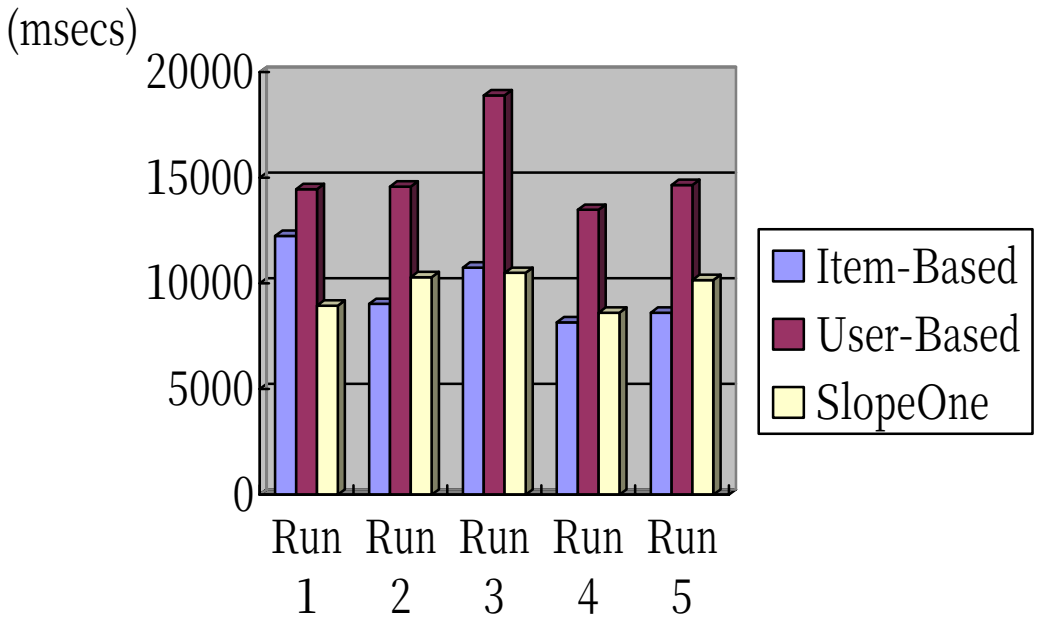


Chart 2: Subsequent Runs Time Consumed

Table 3: Subsequent Runs Time Consumed Data Sheet

Collaboration Scheme	Run 1	Run 2	Run 3	Run 4	Run 5
ItemCorrelation(Pearson)	12250	9031	10750	8157	8609
UserCorrelation(Cosine)	14468	14579	18906	13484	14656
SlopeOne	8953	10297	10500	8609	10156

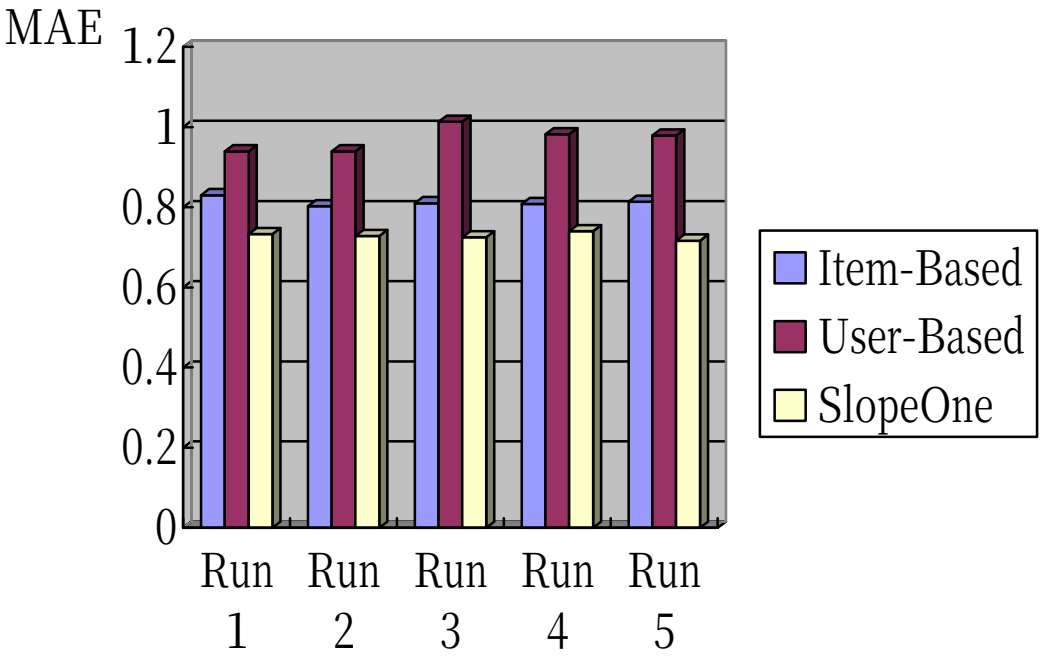


Chart 3: Various Runs MAE (Mean Absolute Error)

Table 4: Various Runs MAE (Mean Absolute Error) Data Sheet

Collaboration Scheme	Run 1	Run 2	Run 3	Run 4	Run 5
ItemCorrelation(Pearson	0.8300	0.8023	0.8100	0.8081	0.8134
UserCorrelation(Cosine)	0.9393	0.9393	1.0132	0.9821	0.9789
SlopeOne	0.7332	0.7284	0.7248	0.7402	0.7166

For the clickstream tree evaluation, since we do not have access to user navigation logs with our current application, we make use of the msweb data courtesy of Microsoft.com covering the web pages each user has navigated in a one-week time frame in February 1998. We evaluated the clickstream tree by first generating the clickstream via the frequent visited navigation paths. Upon completing the clickstream tree, the tree elements (e.g. frequent navigation path) will be scored via

the light collaborative filtering case scoring scheme where the apriori score is calculated for each case. The top cases are then tested to check against a purposely hidden path id (e.g. web page id) to verify if it's among one of the top cases. If any of the case matches, it is considered an accurate path recommendation. The observed accuracy scores are listed in the following Table 5.

Table 5: Clickstream Tree Accuracy

Recommendation Length	3	5	8
Accuracy	0.3333	0.4833	0.5883

4.4. Experiment Analysis

In Table 2, the MAE for among all collaboration schemes are comparable. The runs were divided in two runs: initial run and next run. As expected ItemCorrelation takes the longest time in the initial run as it has to scan through the entire database to calculate the ItemItemCorrelation scores for all items, though subsequent computing time toped all other schemes. SlopeOne scheme ranked first in lowering the MAE, and thus is observed to be the more accurate scheme. UserCorrelation ranked last in MAE and time consumed. It's interesting to see that accuracy actually decreases with greater count of data processed. This is likely to be the result of over-fitting. As a result of this, our Dynamic Collaborative Filtering model efficiently makes use of BPEL engine to dynamically choose a scheme that is more accurate but requires more processing time for smaller data counts and switch to a more scalable scheme that cuts the processing time for larger data counts to balance the prediction accuracy and processing time.

5. Conclusions and Future Work

With the proposed dynamic model, we predicted the potential next page (movie title) of interest with higher confidence via the help of clickstream tree. We observed that ItemCorrelation is the faster recommendation scheme, and SlopeOne predictor is the more accurate scheme. Our dynamic recommendation system based on SOA, orchestrated by BPEL dynamically switches among the schemes to generate more accurate recommendation within a timely fashion in a scalable manner. We expect that for users with committed buying will rent even more movies through the recommendation computed by the dynamically binded collaborative filtering. The ultimate goal of this research is to turn traditional video rental stores into an e-commerce capable business through Knowledge Discovery in Database (KDD) techniques such as product recommendation via collaborative filtering approach. Having the framework built in a service oriented architecture (SOA), we leave the room for improvement with a very scalable and yet adaptable infrastructure. To sum it up, what we achieved in this research is to turn a traditional business into a e-Business by KDD techniques to mine the useful knowledge buried within legacy data in hope that data can some day be formalized into information, information be turned into knowledge, and eventually be transformed into intelligence to not only increase customer loyalty but also maximize the net profit. The data source from Movie Lens, albeit useful in proving our concept will be much more practical when we tailor our design to capture that of a real video store. We've only made use of SOA to orchestrate the collaborative filtering Web Services with our local implementation, the service can greatly be enhanced when external collaborative filtering or data mining schemes can be integrated and orchestrated.

References

1. S.R. Ahmed, “Applications of data mining in retail business,” Information Technology: Coding and Computing, 2004, Proceedings, ITCC 2004, IEEE, pp. 455-459 Vol.2.
2. J. S. Breese, D. Heckerman, and C. Kadie, “Empirical Analysis of Predictive Algorithms for Collaborative Filtering,” Proc. 14th Conf. Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1998, pp. 43-52.
3. A.Y. Chen and D. McLeod, “Collaborative Filtering for Information Recommendation Systems,” Department of Computer Science and Integrated Media System Center.
4. M. Deshpande, G. Karypis, “Selective Markov models for predicting Web page accesses,” ACM Transactions on Internet Technology (TOIT) 2004, pp. 163-184.
5. Ş Gündüz, MT Özsu, “A Web Page Prediction Model Based on Click-Stream Tree Representation of User Behavior,” Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 535-540.
6. J.L. Herlocker, J.A. Konstan, J. Riedl, “Explaining collaborative filtering recommendations,” Proceedings of the 2000 ACM conference on Computer supported cooperative work, 2000, pp.241-250.
7. Z. Huang, D. Zeng, H. Chen, “A Link Analysis Approach to Recommendation under Sparse Data,” Proceedings of the Tenth Americas Conference on Information Systems, New York, New York, August 2004.
8. Dong-Ho Kim, Il Im, Atluri, V., “A clickstream-based collaborative filtering recommendation model for e-commerce,” Seventh IEEE International Conference, E-Commerce Technology, 2005. CEC 2005, pp. 84-91.
9. D. Lemire, A. Maclachlan, “Slope one predictors for online rating-based collaborative filtering,” Proceedings of SIAM Data Mining (SDM’05), 2005.
10. Greg Linden, Brent Smith, and Jeremy York, “Amazon.com recommendations: item-to-item collaborative filtering,” Internet Computing, IEEE, 2003, pp. 76-80.
11. B. Sarwar, G. Karypis, J. Konstan, J. Reidl, “Item-based collaborative filtering recommendation algorithms,” Proceedings of the 10th international conference on World Wide Web, ACM, pp. 285-295.
12. A. Schein, A. Popescul, L. Ungar, and D. Pennock, “Methods and Metrics for Cold-Start Recommendations,” Proceedings of the 25th International ACM

Conference on Research and Development in Information Retrieval, 2002, pp.253-260.

13. Chieh-Yuan Tsai, Min-Hong Tsai, “A dynamic Web service based data mining process system,” The Fifth International Conference on Computer and Information Technology (CIT’05), IEEE, 2005, pp. 1033-1039.
14. Sholom M. Weiss and Nitin Indurkhya, “Lightweight Collaborative Filtering Method for Binary Encoded Data,” Proceedings of PKDD Freiburg, Germany, September 2001.
15. ActiveBPEL Designer, http://www.active-endpoints.com/active-bpel-designer.htm
16. An introduction to SOA, http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa.html
17. BPELJ: BPEL for Java technology, http://www-128.ibm.com/developerworks/library/specification/ws-bpelj/
18. Clickstream, http://www.active-endpoints.com/active-bpel-designer.htm
19. MovieLens Data Sets, http://www.grouplens.org/taxonomy/term/14
20. Recommendation System, http://en.wikipedia.org/wiki/Recommender_system
21. W3C, Web Service Architecture, http://www.w3.org/TR/ws-arch/