Hybrid recommendation engine based on anonymous users

Çubuk, M.

Award date:
2009

Eindhoven, August 2009

Hybrid Recommendation Engine Based on Anonymous Users

by
Mehmet ÇUBUK

MSc. Computer Science and Engineering
Student identity number 0641901

in partial fulfilment of the requirements for the degree of

Master of Science
in Computer Science and Engineering

Supervisor:
Dr. Mykola Pechenizkiy TU/e, CSE
Mentor at the Company:
Business Consultant Guido Budziak, Adversitement
Company:
Adversitement, Uden

Acknowledgments

I would like to thank Dr. Mykola Pechenizkiy and Mr. Guido Budziak for their guidance in this master project. Especially Dr. Mykola's comments and suggestions were particularly helpful for improving the application and both Mr. Guido's and Dr. Mykola's comments and patience during the development of the project were really helpful and ease my burden during the project.

I also wish to thank to Bizztravel for cooperating with Adversitement and providing me the data for doing this master project. And of course I am thankful to Adversitement and all my colleagues in the company for creating a nice working environment and for the support during the project.

Special thanks to my dear flat mates Derya Sever and Firat Gelbal who were always kind and supportive during this master project and their presence in the house helped me to overcome the stress of the master project. And I am thankful and grateful to all my friends in Eindhoven for all the great memories and support.

Last but not least at all; I am most grateful to my lovely family who never once stopped believing in me and whose love helped me come to this position.

Abstract

In this thesis we developed a hybrid recommendation engine for a travel company and web mining is used to gather data for making recommendations. Also we gave a general architecture for recommendation engines and tried to map our implemented recommendation engine to this architecture. Specialty of our recommender is that it only depends on the implicit feedbacks of the users which are gathered through the web logs. Besides that it incorporated contextual information into the recommendation process. Many of the traditional recommenders based on the explicit feedback of the users; however in many cases users are reluctant against giving explicit feedbacks such as rating; therefore we want to solve this problem by making recommendations based on the navigation behavior of the users which is learned through the web logs. This information consists of the users' search keywords, items the users clicked on etc. Also we believe that contextual information such as type of a day has a role while making recommendations; but many of the traditional recommenders didn't consider context; so in this thesis one of our goal is to see the affect of context information for the recommendations. We implemented our product for the online travel company Bizztravel and we evaluated our system by means of hit accuracy and step reduction metrics and gave our results, presented our conclusions in the end of the thesis.

Table of Contents

1 Introduction

In the last two decades internet became the main knowledge base for any kind of information ranging from academic work to industrial applications. It is still growing with a very high pace which results in a new problem for the users of the internet. Because of the high volume of information resides in the web, users need assistance in finding what they are seeking for. Especially in the last decade to solve the problem of finding relevant information; many research have been done and still going on in the areas of information retrieval, recommender systems, adaptive systems, web and data mining. Recommender systems started to be widely used by emerging e-commerce applications for assisting their users to get use of their ever increasing resources. For increasing the customer satisfaction even more personalization techniques began to gain much *a*ttention from academic and industrial environments.

Recommender systems are started to be used for fulfilling each user's needs by taking several issues into considerations; such as user's demographic information and history of transactions, content of the items and resources such as web pages, news, songs, movies etc. web logs which keeps the hidden trails that the users left while navigating in the web. Recently on top of these used information contextual information such as time, place are incorporated into the recommendation process.

In the past, according to the information that is used in the recommendation process, recommendation systems are separated into several categories. The most famous ones were, content-based filtering, collaborative filtering, hybrid-approaches. Most of these approaches were relying on the explicit feedback of the users. However in many situations either from trust issues or just because of laziness, users are not prone to give feedback. Even more importantly, most of the time, the users of the systems are anonymous. This requires modifications in the methods mentioned above such as addition of implicit feedback. Therefore recommenders started to be separated according to which type of data they base their recommendations i.e. whether implicit, explicit or both of it. Otherwise there are more or less well-established recommendation techniques for explicit-feedback based recommendation systems such as Item-to-Item, User-to-User based methods; but with the usage of implicit feedback in the recommendation process these algorithms needs modification such as determination of user interest from the implicit feedback and overcoming the unreliable nature of implicit feedbacks.

Evaluation of the recommendations is another important issue. This is quite a difficult task especially because of the nature of the recommendation process. Even though it is a very subjective concept, for applying it in computer science objective methodologies are needed as it is said that "If you can't measure it don't do it". This mentality leads to using different metric dimensions for evaluating the effectiveness of the recommendations. The most emphasized metric dimensions are accuracy and coverage. These two were the ones used widely in the early researches on this field.

In principal these systems try to predict whether the user will like or dislike an item they didn't see, and this prediction based on the previous transactions of users. Therefore in theory the evaluations of these techniques are quite simple; such as in hit accuracy technique; the data set is separated into training and test data set. Then the recommender is trained with the training set; after that recommendation engine tries to predict the ratings of users for the test set and compare the predictions with the real results of the users. In the end number of correct prediction divided by the number of total prediction gives the hit accuracy of the recommendation engine.

However soon enough, it is seen that these metrics were missing crucial points such as the evaluation of recommending new items. User studies showed that users favor receiving unexpected items from recommendation engines [8, 9]. Therefore new metric dimensions are started to be used such as serendipity and diversity [7].

Also the usability of popular evaluation techniques such as hit accuracy for implicit-feedback based recommendations is in doubt. Because this evaluation technique needs to know the real ratings of items given by users to compare the predictions; but in implicit-based feedbacks ratings are not known. Therefore for these techniques to be applicable; the measure of user interest on an item should be determined precisely in these systems. Some examples of this approach could be using number of clicks on an item or time spent on a particular item for measuring how much the user liked the item.

As a result, researchers come up with a solution to the problem of measuring user ratings without getting explicit feedback and it is web mining. Web mining is the most popular technique for collecting implicit feedbacks by analyzing the web logs which reveals the users' navigational behavior. From these data; how much the user spend on an item, how many times the user clicked on an item, what kind of a route lead the user to this item can be computed and these information is used in determination of the users' interest on items.

Web mining can be separated into three categories. These are web content mining, web structure mining and web usage mining. All of this three techniques have its own application areas, but in aggregation they can be used to discover useful information from the content of the web pages, build underlying link structure of the web site which can be used for calculating similarity between different web sites [1] and for extracting useful navigational patterns which would be used for modeling the users' navigational behaviors. Web mining gets use of many machine learning techniques such as clustering, association rule mining etc. for extracting these information.

In this thesis our goal is seeing the affects of using different kind of recommendation techniques such as content-based, collaborative-filtering, context-based recommendations in an application where the users are anonymous and we retrieve all the information about the users through web mining. We want to see how the anonymous user nature and implicit feedback affects the performance of these techniques and which technique is most suitable for making recommendations to anonymous user based applications.

1.1 Project Definition

In this master project we developed a hybrid recommender engine for the company named Adversitement. Main goal of our work is developing a recommendation engine which is based on usage of implicit feedback and anonymous users. This product will be used by Bizztravel which is a customer of company Adversitement that helped me in my project by letting me use their applications to do web mining. Apart from this implementation a generic architecture for a recommender engine to be used in e-commerce applications is given. In our implementation we focused on improving recommendations based on implicit feedback. Details of the project, architecture, motivation behind this work and overview of the company will be given in the rest of this part.

1.1.1 Company Information: Adversitement

Adversitement is an e-marketing consultancy specialized in web analytics. Web analytics is the reporting and analysis of the usage data of web sites to understand the customers, visitors of these web sites better. Then according to the outcome of web analytics, web sites optimization considering the business goal of the web sites can be made. For doing web analytics, huge amount of usage data is collected which tracks the user actions in the web sites such as which pages they visited, how did they access to this site, etc. Then statistical analysis is applied to this data for understanding the behavior of users, server traffics, and this knowledge could be used to optimize server traffic, web site navigation, etc.

So far Adversitement is only making statistical analysis on the web logs for using it in web analytics. However, these data can be even more useful by applying advanced techniques. There are many machine learning techniques such as association rules, clustering, pattern analysis which could use this data for extracting important knowledge about the user behavior, interests, etc. This information could be easily used in recommender engines for adapting the websites according to the needs and preferences of the users.

With this project, Adversitement exactly plans to do that, extending their business coverage by developing recommendation engines for their customers. As they already have huge amount of web usage data and a developed web mining framework to use these data, they are ready for working on a project like this. We will try to use this framework for collecting web data and will try to improve it by applying more advanced techniques such as item-to-item filtering, pattern analysis beside statistical analysis. Then we will develop our recommender system on top of this framework.

1.1.2 Customer Information: Bizztravel

Bizztravel is one of the customers of Adversitement which uses their experience in web analytics for improving their website. Bizztravel is one of the many travel companies in Netherlands which tries to sell holiday packages to their customer through internet. They have different web sites for different type of holidays and one of these applications is BeachMaster.nl which is a web site for selling summer holiday packages. Their main customer segment consists of students and young people and they sell holiday packages from all over the countries located near Mediterranean.

As there are many other web site which sells travel packages Bizztravel wants to become better than its rivals. They believe that by using a recommendation engine they can improve their sales and can help their customers in searching a holiday. By using recommendation engines they can show the best results to their customers and get advantage over their competitors.

Also as their application based on anonymous user; we decided to implement a recommender engine suited to the anonymous users. Even though recommendation engines become very popular in internet right now thanks to big applications such as Amazon; Last.fm these applications relies on identified users but there aren't many applications based on anonymous user and uses recommendation engines. Therefore this application will give us a chance to see how to apply recommendation strategies for anonymous users.

The current application based on search mechanism. In the web site there are several search facets which the users can use for making searches on several features such as Destination, Price, Departure Date, Number of Travelers, etc. Then the results are displayed to the user in an information retrieval fashion where items are filtered out according to the given search keywords. Bizztravel wants to improve its application by displaying recommended items

beside these search results. With this engine they want to increase the number of customers who stays in the website until they reach the order step instead of leaving the site after search step. This goal is illustrated in Figure 1

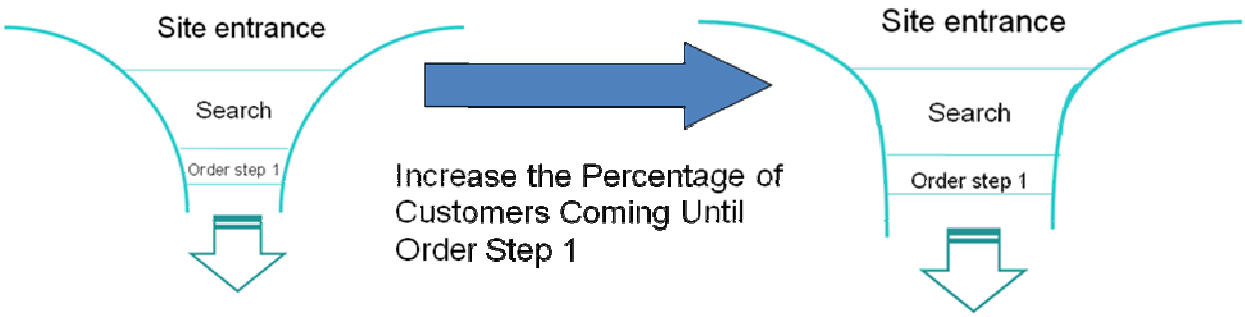

Figure 1: Improving the Conversion of Customers From Search Step to Order Step

1.1.3 Details of Recommendation Engine

Our recommender engine used a hybrid model. It is composed of content-based, collaborative-based filtering with a knowledge-based component. In addition to these components, it also uses contextual information such as week day type. Instead of relying on explicit feedback of the users which is the standard way in traditional recommendation engines, we tried to exploit implicit feedback from the web usage data that resides in web logs. As the application we worked with is an anonymous user based application; we made modifications in the techniques that are used for explicit-feedback based applications.

To illustrate how our hybrid model works we will give an example of it here. First of all according to the actions of users which could be either clicking on a holiday package or making a search for holiday we choose the most similar items according to the current user model. So let's assume the user clicked on a holiday in Greece; our recommender will first recommend an item to user which is the most similar item to this holiday according to our item-to-item algorithm. We do this to give more importance to the last action of the user. Then the rest of the recommendations will be based on the history of the users. Some of these recommendations will be the one most similar to the user model according to our content based filtering and others will be the one most similar to collaborative model. Our last recommendation will be chosen from the knowledge base component. For example if in our knowledge component algorithm there is a rule such as "if the user is searching for holiday in Greece recommend the item GR2009"; GR2009 will be displayed as the last recommendation to user. The main idea of knowledge based component is to give a chance to Bizztravel to control the recommendation process by giving more importance to particular holiday packages.

As mentioned earlier, the main method for learning user behavior and preferences is done through web mining. We did user modeling based on the items user clicked, the search keywords the user used, and from where the user came to the application (from Google, by bookmark, etc.). Then according to the user model, proper recommendation algorithms are used to improve the qualities of recommendations.

1.1.4 Methodology of our Project

In our implementation we applied goal-oriented approach, rather than data-driven approach [54]. In goal-oriented approach, first the metrics that will be used in the evaluation of the system should be chosen. Then according to these metrics, items (web page, news, images, travel packages, etc.) that will be recommended to users should be determined. User modeling

and recommendation strategies that will be used in the system should be selected for delivering these items. Finally the relevant data for building the user profiles and making recommendations should be collected. So here is our approach in the implementation of the recommendation engine for the travel web sites.

- **Metrics to be used:** The main goal of the companies are increasing their sale revenues, so therefore the main metric for our recommendation engine will be number of visits that ends with a purchase. However for evaluating this we need to run our engine in live environment but this is not possible because of time limitations. Therefore hit accuracy metric will be used for determining how well our engine is predicting items that users will be interested in. In the future when the application starts to be used in Bizztravel, sales happening through recommendations would be a better performance indicator for our engine or measuring the increase in the conversion from search step to order step might be another good indicator.
- **Recommended Items:** We recommend holiday packages to our users which would be most interesting for them according to their user model.
- **User Modeling Techniques:** User modeling is based on the implicit feedbacks of the users. Implicit feedbacks could be search keywords, destination of holiday, and number of travelers, referrer page, items the user viewed, etc.
- **Recommendation Strategies:** Combination of Content-based, collaborative filtering, web mining and knowledge-based reasoning is applied as a recommendation strategy. In addition the contextual information is integrated into the recommendations. Content-based is based on item similarities according to the item features. Collaborative filtering techniques are used for holiday recommendations according to the items user clicked on and keywords they used in their searches. Web mining techniques are used for learning the actions of users in the web site. Knowledge-based component is composed of campaigns and promotions that are entered by Bizztravel. Apart from these techniques, contextual information such as weather conditions, price of holiday packages, number of travelers, etc. are used to improve the quality of recommendations. However because of time limitations only day type is directly used in the recommendation process whereas other contextual information is used implicitly through the algorithm based on search terms.
- **Data Collection:** Web usage data of the users are collected. This data apart from standard fields such as IP-Address, referrer-field, browser, request time, etc. consist of user feedback such as search keywords, holiday destination. Adversitement has already advanced data pre-processing techniques for identifying sessions and this will helped us a lot in the web mining phase.

1.1.5 Motivation for the Project and Methods

After explaining our methodology, we want to show why we applied these methods and worked on this project. In the rest of this section our main motivation behind this project will be given.

In the last decade it is seen that e-commerce applications, news portals, basically web-based applications need an efficient mechanism for providing all of their services, products and contents to users. Because of the huge number of content, users of these applications need assistance while using them. Every user only wants to see content that they are interested in or even worse they even themselves don't know what they are looking for exactly and they want to see items (pages, products, services, news etc.) which might be interesting for them. This requires the contents of the applications to be adapted to the needs of the users. Web recommenders are the solution for this problem, and started to be used widely in many

applications such as Amazon, Netflix and Last.fm which are the leading companies in their markets. Even these examples show that success of e-commerce applications heavily depends on the personalization capabilities.

Because of the large variety of products, and increasing number of content in internet, finding something interesting or relevant turned into a difficult task. Especially this is the case for new released items. Recommender systems have been applied in various applications but most of these systems couldn't succeed particularly well [11, 16]. This might be because of the fact that they rely on traditional recommendation methods such as content-base, collaborative filtering. Both of these techniques have their advantages and disadvantages; but their common disadvantage is that they can't handle cold start problem i.e. not able to handle recommendation of new items in the systems or recommend items to new users.

Another problem with the old recommender systems is that, they don't take contextual information into account. However contextual information such as weather, companions, time should have a crucial part in recommender engines. Recommendation engines should imitate what people do in their practical life while making suggestions. People take context into account implicitly such as considering the companions of the recommended person or the importance of the day like the Valentine day while making suggestions and this kind of information need to be reflected on to the recommender systems for retrieving better results.

The biggest problem of the traditional recommendation techniques is that they rely only on the explicit feedbacks users are giving. However many users are reluctant to give explicit feedback. In this case, systems have deficiencies in making recommendations because of the sparse rating data and missing user information. Especially collaborative filtering but also content-based filtering needs many ratings for every item even though the similarities between items can be computed with content based filtering but it also needs ratings for determining whether the user liked an item or not. This requires systems to have much more users than items so that items might receive more ratings and similarity computations between items would be more reliable. However in most of the applications this is not the case. Therefore pure usage of these techniques is not scalable to handle huge amount of content; because of performance issues.

In this master project deficiencies encountered while working with recommendation systems such as cold-start problem, over-specialization, lack of feedback from users, working with sparse data-set is tried to be resolved. We proposed to solve these problems by developing a hybrid recommender system which takes contextual information into account while combining it with implicit feedback from users to create user models. Context information might solve these problems by making recommendations only based on context information such as recommending the most famous movies that are watched on Friday midnight to a new user or to a user who doesn't give any feedback

Demographic filtering can be used for solving the new user issue of the cold start problem. This information can be either retrieved by personal forms or again implicitly by analyzing the IP-address and understanding where the user came from. This information can be used to localize the application such as recommending items corresponding to the place user came from or automatically translating the language. For new users who started to use the system recently, there won't be much data to make recommendations, however demographic information such as age, nationality could be a starting point for making recommendations to the new users. In our application as all the users will be anonymous and the customers are mainly from Netherlands and Belgium, demographic information was not applicable. However contextual information

such as referrer is used to make initial recommendations. Also by using implicit feedback; our system is able to determine interest of users automatically without waiting for explicit feedback of users; which partially helped us to deal with the cold start problem for new users.

Using contextual information while making recommendations will be a cure against making unusable recommendations to users such as recommending a single person holiday to someone looking for a family holiday. This information will be used implicitly in our recommendation model thanks to our recommendation algorithms which based on search keywords that includes information about price, number of travelers which implicitly use contextual information.

Knowledge-based component and content-based filtering are used for making associations between new items and old items in the system which will be a remedy for new item issue of the cold start problem. Domain experts, company employees will enter if-then rules to system for adding more value to certain products. These rules will be added to system through a GUI where the domain experts will enter the items and keywords that are related to these items such as Destination or Price Range. Then in the application when the user search a holiday with some destination which is related to a product in the knowledge-based component; the related item will be recommended to user. This information could also be used to recommend items which don't have any user feedback. The screenshot of the GUI for adding campaigns and products can be seen in Figure 2.

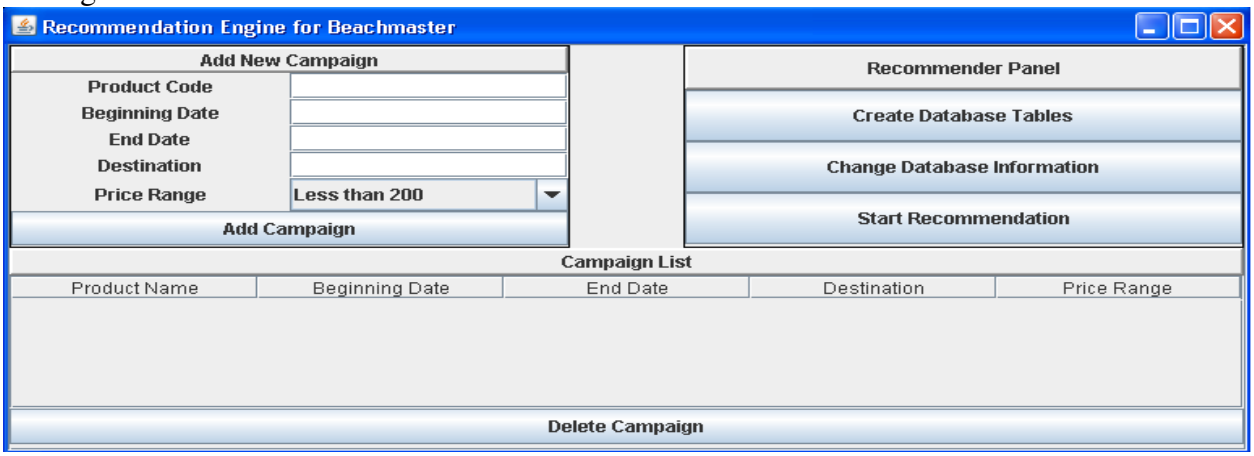

Figure 2: GUI for adding Campaigns

Researches show that item based filtering techniques give better results than user-based recommenders. Therefore item-based techniques are used in the project. One of the well established methods for computation of the similarities of items is cosine metric, and we used it in this project. Actually we applied adjusted cosine similarity for considering the differences in user's rating, i.e. 7 out of 10 for one user might be high and for other user it can be a low rating; therefore each user's average rating should be considered in computing item similarities.

As many of the recommender systems, our system is also composed of two phases that are; offline phase and online phase. In the offline phase analysis of the web logs, extraction of important information from these logs will be done by using web usage mining. In the online phase, user models will be made or will be updated according to the users' current session, and according to the model, recommendations will be given to the users.

By doing web usage mining, implicit feedback is gathered which contains valuable hidden information about user preferences. This information is our only source for user modeling; therefore web usage mining was an essential part in our project.

To help readers, summary of the motivation of this project will be given in the next section by showing the problems of recommender engines and proposed solutions to these problems.

1.1.6 Problems of Current Recommendation Engines

1. Pure recommendation techniques have several deficiencies such as:
    a. **Over-Specialization:** In content-based filtering users only receive recommendations that are similar to the items that they rated before. Novel item recommendation cannot be made because, recommendations only based on user profiles and social relations of users are not considered while making recommendations.
    b. **Cold-Start Problem:** New users can not receive good recommendations because they don't give enough feedback for getting recommendations, similarly new items can't be recommended as they are not rated by enough users.
    c. **Computation-time Deficiencies:** Most of the techniques are not scalable enough to handle large quantity of items.
    d. **Explicit Feedback:** Heavily relies on explicit user-feedback which users are reluctant to give. Therefore they can't make proper recommendations with sparse data set.
2. Contextual information such as weather, time etc. is not considered in recommendation process; however it is a fact that recommenders need to consider contextual information while making recommendations.

1.1.7 Solutions

1. Develop a hybrid recommender for compensating the deficiencies of pure techniques by mixing them. This recommender will use techniques such as collaborative, content-based, knowledge-based component beside contextual information. This approach might solve the following problems:
    a. **Over specialization:** By using collaborative filtering, this problem will be overcome by considering other users' interests who have similar tastes to current user.
    b. Cold-Start Problem: As our system based on anonymous users, every user will be treated as new user and we will try to overcome new user problem by using previous transactions of users and trying to recommend items to new users according to the items they are interested in and search keywords that they used. So as soon as the new user clicked on an item or made a search; our system will be able to come up with recommendations by using the item similarity matrix or again by using the mapping between items and search keywords. New item problem will be solved by knowledge-based component and content-based filtering; but because the item space in our application doesn't change frequently, this issue won't be a serious problem in our project.
    c. Computation-time Deficiencies: Recommendation process is separated into two phases for solving this problem. It will consist of offline and online phases. In offline phase the data collection and processing will be made where as in the online phase user modeling will be done. Then according to the current user model and past transactions of the users, recommendations will be made based upon the recommendation model that is built in offline phase.
    d. Explicit Feedback: Implicit feedbacks from the users will be collected for user modeling by web usage mining. Web logs contain secret usage behaviors which could be used in user modeling. Apart from these logs, by analyzing the click-

stream and session information, navigational behavior of the users will be extracted.

2. While making recommendations, contextual information is incorporated into process, so that different recommendations will be made according to the context. For example some search parameters such as price, number of travelers will reveal items that are more suitable according to these parameters. So from the search queries of the users; we will get contextual and content information; whereas from the clicked items we will gain the content information. Then content and contextual information will be combined and recommendations will be presented to users. Also we will try to see how the user behavior changes in week and weekend days, and try to determine whether it makes sense to make recommendations based on day type or not.

In Figure 3, the data flow of our recommendation engine and the mapping of search query to content and context information can be visualized. In the figure green objects represents resources related to users; blue object represent contextual information, red object represents the results shown to user and orange objects represents the modules related to recommendation process. Because of the time limitations; we only checked the affect of using week day type; but not able to check for the other context information such as price range.

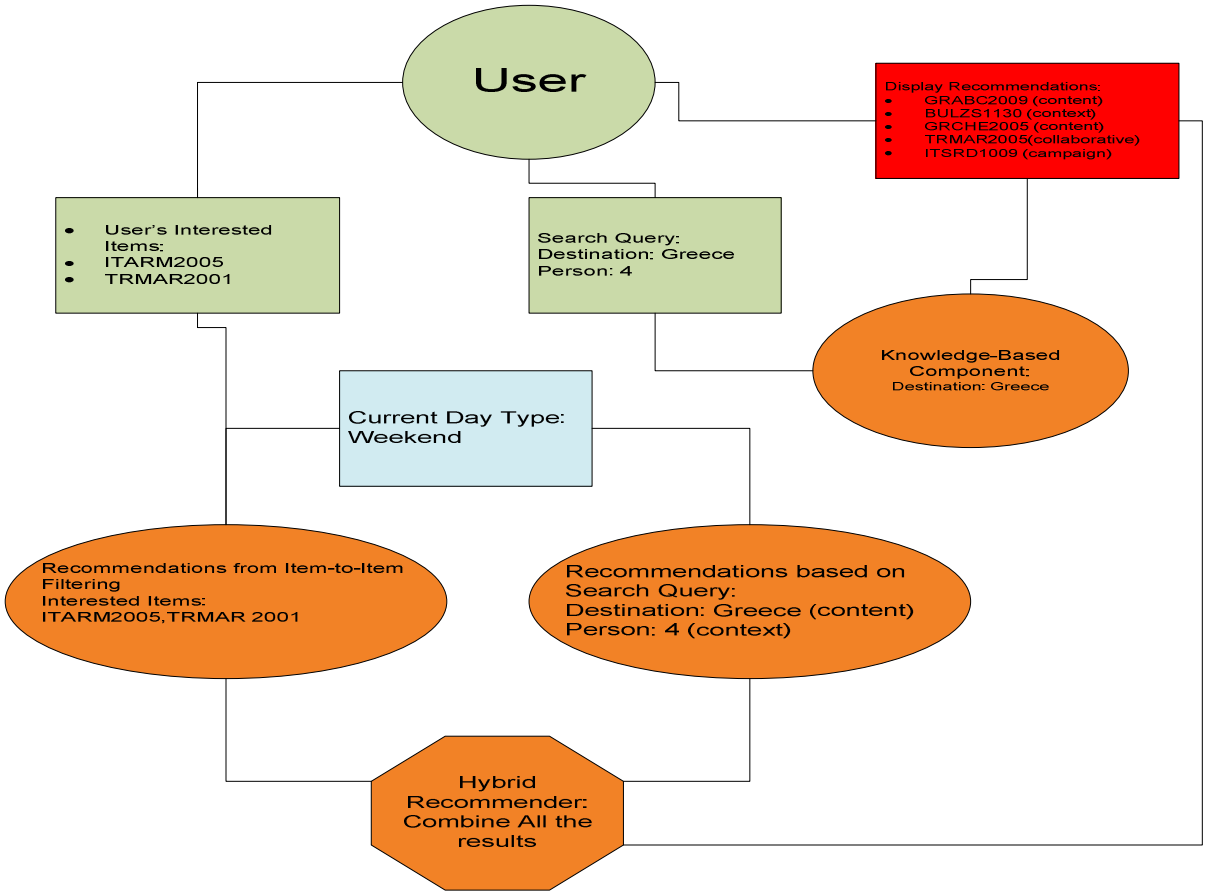

Figure 3: Data Flow of the Recommendation Engine (Mapping of User Model and Context to Algorithms)

In the next part we will give our research focus in this project.

1.2 Research Focus

1. Development of hybrid recommender which consists of collaborative, content-based and knowledge-based component that uses contextual information in the recommendation process while using implicit feedback in user modeling.
2. Applying web usage mining for retrieving information and implicit feedback from usage data.
3. Making recommendations purely based on anonymous-users and seeing how the performance of known techniques such as item-to-item filtering is affected by this.
4. Determining new evaluation techniques for recommendation techniques that are based on anonymous users such as measuring the number of reduced step in search path of purchases.
5. Incorporating contextual and content information into recommendation process and see how it affects the performance. Testing how the performance is affected by making recommendations based on day type.
6. Determining a method for evaluating users' interest level on items by using implicit feedback such as number of clicks, user events.

1.3 Specialty of Project

Much research has been done on recommendation engines. These researches covered algorithms to be used, how to improve recommendations, performance issues, etc. But common point of most of these researches is that they are based on explicit-feedback applications. However in our project we tried to see how the recommendations engines perform by depending only on implicit feedback and anonymous users.

Anonymous users created many difficulties in recommendations as the main idea of recommendations is that recommending the best items to users according to their profile; but in our application the user profiles only depend on current session of the users and therefore profiles would be deficient to make proper recommendations. To avoid these difficulties we tried to apply techniques which determine item similarities and relations between search keywords and items.

Also instead of depending on ratings we tried to find out what is the best indicator for understanding users' interests on items. According to these indicators our engine determined which items users liked and didn't like, then it recommends new items that are similar to liked items of the current user and items that are related to the search keywords of users.

Another interesting thing about our project is the characteristic of our application. As it is a travel site where holiday packages are sold for summer vacations, it has a very different marketing strategy than famous applications such as Amazon, EBay. It is because in Bizztravel; customers don't make transactions very frequently on the contrary most of the time they only make one purchase per season, as they don't go on a holiday more than one time a season. However in famous applications; customers make much more transactions and they have a much larger item space than our application. Therefore for these famous applications making novel recommendations to users, showing interesting items to them is more important than recommending items that are obvious that the user would like.

However for our application, the most important thing is to recommend items that would be most interesting for the users rather than surprising them. As a result of this observation in our engine, we tried to recommend items that are most suitable to current profile and hope that we would capture the attention of customers so that they won't go to another website to search for

holiday packages. We chose this approach because there are many similar applications to Bizztravel and it is known that customers are searching holidays in several web sites and most important thing for Bizztravel is to not lose potential customers just because not showing an item in time. However for not falling into the trap of overspecialization; we used collaborative filtering algorithm to recommend items that are not corresponding to the exact search query. If we didn't apply this mechanism; there wouldn't be any point of using our system; because without the collaborative filtering algorithm; our system wouldn't have much difference than a search engine which is currently employed by the application itself.

1.4 Proposed Architecture

Also we want to give a general architecture which is not only suited for anonymous users but could be applicable to any kind of application. Here is the generic architecture for recommendation engines which covers almost every aspect of recommendation strategies.

Figure 4 can be separated into three parts; namely they are data sources, web mining and online modules. Data Sources consists of user and item databases which keeps information about these assets, and web logs which keep information about the usage data of the web site.

Web usage mining module will use these web logs to extract useful information about users such as determining users' interested items. This module will be executed offline as the computations in this phase cannot be done in real time and needs some preprocessing. The extracted information from the web logs will be combined with other data sources for user modeling.

Lastly in the online phase which is shown at the bottom of the Figure 4 will make the real recommendations by using the outcomes of offline phase and data sources besides using the knowledge based component, contextual information and the current user model. Knowledge based component will contain the business rules such as promotions, campaigns that are entered by the domain experts. Also recommendations for the anonymous users and identified users will be separated. Actually recommendations for the identified users will be extension of recommendations for anonymous users; as in the recommendations for identified users besides the resources that are used for anonymous users; user database will also be included in the process.

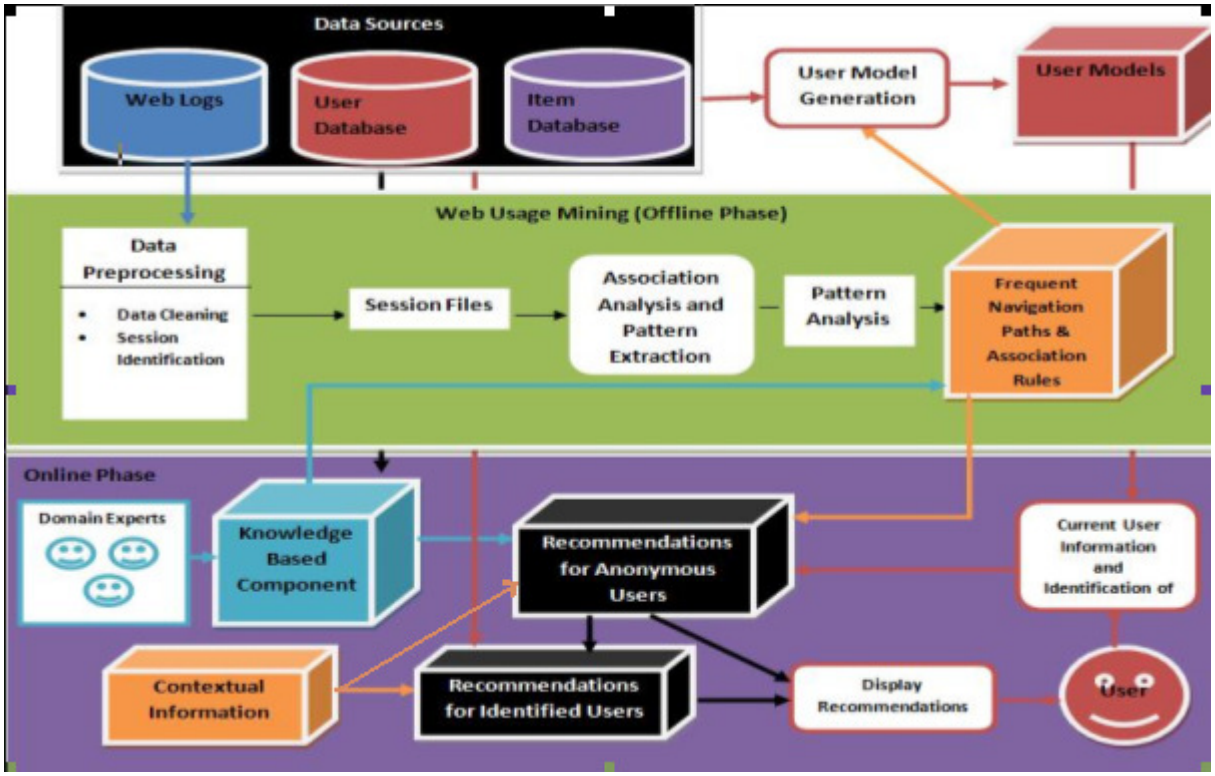

Figure 4: Architecture for the Recommender Systems

1.5 Results

In our research we come up with several hit accuracy results trying different kind of algorithms under different context. In our trials we wanted to see how the performance changes according to the type of the algorithms used and we wanted to determine whether pure recommendation algorithms perform better than hybrid or not. We also wanted to see how the contextual information such as type of day whether it is a week day or weekend day, affects the recommendation process.

We applied a hit accuracy evaluation technique that we will be described in details in the section 5. Results for the pure collaborative filtering algorithm were around 40% whereas the result for the hybrid recommender which combines collaborative filtering with content-based filtering is around 55%. These results support the superiority of hybrid recommenders over the pure algorithms.

Other interesting result we obtained is that, contextual information that is the type of day has importance in the recommendation process and gives better performance when it is considered in

the model. Our results showed that when the recommendations are made with a recommender which is trained according to the type of the day, results are improved by 5-10% compared to recommender which uses all data without separating data according to the day type.

Lastly we observed that our recommendation algorithm based on search terms makes around 3 search step reduction. In this evaluation we tried to see when our recommender engine was recommending the purchased item to user while they are searching. And it is observed that in average our algorithm is reducing the search step by 3 searches where in average 10 searches happen in a purchase. This shows that our engine reduces the average search path by 30%.

1.6 Guide for the Readers

In this section we will explain the structure of our paper and try to guide our readers. In section **Error! Reference source not found.** we will explain the literature study we made before and during working on this project. We gave an overview of Web Personalization in section 2.1. Then we explained the current state of art for recommendation engines in section 2.2 and also explained the type of recommendation engines in sub chapters of that section. Related work about Web Usage mining which is the essence of our recommender is explained in section 2.3 and in section 0 we gave examples of the projects which used recommendation engines.

Our framework is given in Section 3. We explained the goals of our project in section 3.1 and section 3.2 summarizes our methodology for achieving these goals. In section 3.3 technical details of our working environment is given.

In Section 4 we explained the development process of our thesis. Section 4.1 gives information about how we collected the data which is the first phase of web usage mining. Then in section 4.1.4 data analysis which is the second phase of the web usage mining is explained. Section 4.2 and its sub-sections give our design choices in our development such as assumptions we made in the development, how we determined the user interest, how we modeled users and items, choice of recommendation algorithms and usage of contextual information are all explained in this section. Details of the recommendation algorithms that we used in our system are explained in section 4.4. In 4.5 we tried to map our developed system to the architecture that we proposed for the ultimate recommendation engine.

Section 5 explains our evaluation techniques and we gave our results in the subchapters of this section. Finally in section 6 we gave our conclusions about this master project and show possible directions for the future work in this project.

2 Background

In this section we will describe what has been done so far in the area of recommendation engines and web personalization which covers recommendation engines and more. We will try to gain experience from what has been done so far and try not to fall into the pit-falls and improve current state of recommendation engines by applying our approach.

2.1 Overview of Web Personalization

Web personalization combines information about user data such as name, age, ratings users give to items with analysis of the users' navigational behavior. We believe that, successful web personalization should combine methods of web mining, and traditional recommendation techniques with incorporating the contextual information, for the following reasons:

- Neither pure explicit feedback based systems nor implicit feedback based systems are enough for modeling the decision making of users, which is required for effective recommendations.

- Explicit-based approaches are not sufficient because of the fact of uses' reluctance in giving explicit feedback. Contrary to pure explicit-based approaches, hybrid approaches could use little amount of explicit feedback such as ratings much more efficiently by combining it with implicit feedbacks.

- Implicit-based approaches are insufficient as most of the implicit feedbacks are not significant enough to use them in the recommendation approach. Also there aren't suitable techniques for measuring the value of implicit feedbacks objectively. It is seen that many misinterpretations are made in the evaluation of the implicit feedback; therefore these implicit feedbacks should be supported by using explicit feedbacks.

- In many applications such as recommendation of movies, contextual information is needed for coming up with the "right" recommendation. For example according to companion you will watch the movie, the recommendations would be totally different [6].

Next section will give an overview of the recommendation systems. It will describe the widely used recommendation techniques. We will give each technique's overview and explain the advantages and disadvantages of them.

2.2 Recommendation Systems

Actually what recommendation systems do, can be summarized as information filtering. Essentially they are filtering items (books, music, movies, news, images, web pages, travel packages, etc.) according to the needs and interests of each individual user. This is achieved by comparing users' profiles with recommendation model that is built up by applying different filtration techniques such as content-based or collaborative.

User model is composed of personal information such as age, sex, nationality etc. and preferences, interests of users. These preferences and interests can be gathered by asking

explicitly from users by means of internet forms, rating-based evaluations. But the more interesting part is gathering this information by implicit feedback which could be gathered by analyzing the usage behavior of users. As you can see, recommendation systems can be categorized into many different types according to the methods they use. This categorization can be made according to many aspects such as recommendation approach, data collection method, user profile generation etc. [10].

In the following sections each of these aspects and their corresponding techniques will be elaborated. We will start by discussing recommendation approaches and we will first explain the content based approach.

2.2.1 Content Based:

It is based on measuring similarities between new items and items that users preferred in past. This preference can be measured in many ways such as whether user purchased the item or not, binary ratings such as good or bad, decimal rating based. Actually content-based recommenders try to predict ratings for un-rated items, and recommend the items to users that got the highest rating.

So it can be seen that, this approach solely depends on the user profile which does not change frequently. One advantage of this property is that users most of the time don't get recommendations that are not interesting for them. Because whatever they get as a recommendation supposed to be related with their interests as those recommendations based only on their preferences.

This approach requires contents of items contain enough information which would distinguish them from each other. This requires items of these systems to have a structured data definition, so that they can be compared easily with each other. For example restaurants (food type, restaurant size, price, music type, etc.) can be stored in a structured data easily. However if the items are not suitable for storing in a structured data such as web pages with plain text, it is not convenient to use content-based filtering.

Nevertheless there are still methods for converting these unstructured texts into structured format by applying conversion techniques such as stemming. Stemming is the method of extracting root of each word. For example it can extract the main meaning which is recommend behind the words of recommendation, recommending, recommender [11]. Then methods such as terms frequency (tf), terms frequency times inverse document frequency (tf*idf) can be used for comparing unstructured data. However this conversion still loses the context of the word used in, it also loses the relationship between words. For improving the accuracy with these kinds of items, natural language processing is needed.

As content-based filtering only depend on user and item profiles, this prevents them recommending novel items, because profiles' static nature. This leads to over-specialization which is one of the biggest problems of this approach. Users always receive similar recommendations to their current interests, so they don't encounter with different items which could reveal their hidden interests.

Another problem is the lack of desire for users to give feedback. In many applications users are unwilling to rate items, or give personal information. Of course this prevents content-based approach from making recommendations; because for building user profiles they need to interact with the user all the time, personal information of users, history of user transactions are needed.

A common problem in many recommender systems is also seen in this approach. It can be named as new item problem, new items added to system can't be recommended until users rate

these items; because they don't have any rating in the beginning, there isn't any basis for recommending these items. Possible partial solutions to this problem is predicting ratings according to the features of the item, or giving manual or random ratings while adding new items to system.

There are many ways for learning user modeling. Creation of the user modeling is a type of classification learning. The main idea of the classification is to distinguish which items user "likes" and which items user "doesn't like". These distinctions mostly have done through using explicit feedbacks from users. However implicit feedback such as considering "purchase of a product" as a positive feedback can be also used. There are several machine learning methods which can be used for deciding on whether user will like the item or not. These are the following Decision Trees, Rule Induction, Nearest Neighbor Models, Relevance Feedback, Linear Classifiers, Probabilistic Methods and Naïve Bayes.

- **Decision Trees:** It recursively partitions training data which is text documents into subgroups until, these groups contain only instance of a single class. These partitions are formed based on some distinctive features which are found out by using expected information gain function. They are particularly useful in classifying structured data; however same cannot be said for unstructured data. Its simplicity and understandability are the biggest advantages.

- **Rule Induction:** It also recursively partitions the data into subgroups similar to decision trees. However its computation performance is better than decision trees as a consequence of post-pruning. It is also suitable for using in semi-structured or even unstructured text documents.

- **Nearest Neighbor Models:** It stores all its training data in memory, and classifies the new items by comparing it with all stored items using a similarity function. Then it computes the "nearest k-neighbors", and the new item is put into this neighborhood. For structured data, Euclidean distance metric is used whereas cosine similarity is used for vector space model. As it is a memory-based technique, it has computation deficiencies while working with large data sets. However, contrary to its simplicity, especially for the text classification it still competes particularly well with complex algorithms.

- **Relevance Feedback:** It allows users to rate documents retrieved upon their queries. So the retrieval system, according to the feedback of the user, refines the initial query and come up with adapted results. Rocchio's algorithm is widely used in relevance feedback algorithms. It modifies the initial query based on weighting relevant and non-relevant documents. Its idea is to incrementally get query closer to the cluster of relevant documents and get query away from the non-relevant document cluster. Empirical experiments have shown that this approach improves retrieval performance [12].

- **Probabilistic Methods:** While vector space models don't have support from theoretical justifications, probabilistic methods seen much attention from that perspective. Naïve Bayesian classifier is one of the most popular examples of this approach. Researchers

have shown that Naïve Bayes performs very well in text classification [11]. There are two formulations used in Naive Bayes that are multivariate Bernoulli and the multinomial model. The first one works on top of structured data, and it assumes that every document is represented as a binary vector over the space of all words. So each element of the vector represents whether a word has occurred in the document or not, therefore it doesn't consider word frequency. In contrary multinomial approach uses word frequency for classification of text documents. Empirical results shown that multinomial approach outperforms Bernoulli approach. Independent from the formulation used in Naïve Bayes algorithms, they are used widely in content-based applications and empirical results show that they perform very well.

After giving the techniques used in content-based recommenders, the advantages and disadvantages of the approach will be given as a summary.

Advantages:

- Simple to understand, works very well for static preferences that doesn't change frequently.
- Based on well-established methods of information retrieval.
- Users are not encountered with items unrelated to their interests.

Disadvantages:

- Users' reluctance in giving explicit feedback. User modeling is the core of this approach and without user feedbacks; reliable user models cannot be constructed. Using implicit feedback can be seen as a remedy to this issue; however implicit feedback may easily mislead recommenders because of the high uncertainty of it compared to explicit feedback.
- New item problem. New items cannot be recommended until they are rated by users. This reduces the coverage aspect of content-based systems.
- Over-specialization. Users get recommendation only similar to items that they rated highly and they don't receive any novel recommendation [14]. Therefore pure usage of this technique is not suitable in dynamic environments where interests of users, items change rapidly.
- Content based approach needs a representation of items in terms of features of items. Items that cannot be represented in terms of features are not applicable to be used for content-based recommenders; because they need manual intervention to be used in recommenders. For example news can be automatically represented in terms of features by parsing, however representation of items such as movies, songs need manual intervention to represent them in feature sets which makes content-based applications not scalable for large item sets.

Content-based approach has several deficiencies and doesn't perform well in many situations because of the fact that it only considers individual user item set. It focuses on what current user has done in the past, how he or she interacted with items, and according to this information, it tries to recommend similar items to user. However this approach misses a very important aspect that is the social interaction between people. It doesn't exploit the fact that people who are interested in particular common items might have similar tastes and this similarity between users and other people's opinions could be used in recommendation process. In the next section, contrary approaches to Content-Based filtering such as Collaborative Filtering that is based on opinions of other people will be elaborated.

2.2.2 Collaborative Filtering

Collaborative Filtering (CF) uses the opinions of other people in the recommendation process. Instead of focusing on only current user's transactions and behaviors, it considers the opinions of other people who seem to have a similar taste with the current user. Most of the time similarity of taste is evaluated by considering the similar items the users have. Actually it bases on the observation that, people share opinions with each other, and they use these opinions while choosing their actions [16]. Therefore collaborative filtering can be called as a much more social-based recommendation approach, as it tries to mimic how people makes recommendations in real life.

In real life, people use suggestions of other people for going out to cinema, or planning a holiday. Through the years, based on their experiences they started to trust more to the suggestions of their particular friends, and stopped getting recommendations from specific friends. In real life, this process contains tens of people, whereas with internet same procedure can be applied between thousands of people. Collaborative Filtering tries to exploit the information processed in the huge social network of the internet.

According to [16] CF can be used in the following situations:

a) Help users in finding new items that might be interesting to them

b) Advise users on a particular item by providing other people's opinions.

c) Help users in finding similar users to them.

d) Help group of users in finding something that is interesting for all of them.

e) Help users find interesting activities according to the context.

Many applications are designed to work under the situations of a, b and c such as movie recommender systems (a, b), e-commerce applications (a, b), match-making sites (c); however there are still not many applications which works under the situations d and e. Recommendations to group of user is an area which haven't got much attention from researchers yet. However incorporating contextual information to recommendation process which is the case e, started to get attention from the academic environment [6, 18, 19]. It is seen that recommendation of the same item type such as a movie might be rather different according to the situation of the user.

Above we presented where collaborative filtering could be useful, and now we will present what are needed for collaborative filtering to be useful.

a) There need to be many items so that users need computer assistance to learn about them.

b) There are many ratings per items and there is an overlap between the items that users rated, because of the requirements of the techniques for computation of the similarity of users and items. For comparing similarities of users and items, base set is needed where the recommendations can be built on, for CF this base-set is the overlap item set.

c) For each user, there are several similar users who have common interests in the user community. Actually this is the assumption based on CF; because it believes that people have common tastes which could be shared for recommending new items to each other. If every user is unique and each user has its own unique tastes CF can't work.

d) Item evaluations need to be subjective; such as evaluation of a song. Items shouldn't be categorized based upon on objective features that can be automatically calculated such as calculating the theme of an article by computing term frequency. An example could be evaluation of music where it is not easy to evaluate it by usage of objective features. Even though features such as genre, artist could be helpful in categorization of songs, it is still not possible to make a recommendation only based on these objective features.

e) Items are homogeneous. This can be seen as an extension of above requirement. Items need to be homogenous in terms of objective criteria such as price, size, etc. Recommending a 7-star hotel from Dubai to someone who purchased a hostel room is not very useful.

As can be seen from the above requirements, CF requires a setting where comparison of items, users can be made which based on subjective criteria. More importantly CF needs to match users who share common interests by evaluating their past transactions.

However matching similar users is a difficult task, this can be considered as determining the "friends" of users. Because when the proper "friends" are chosen for users, better recommendations can be made considering the opinions of these "virtual friends". Therefore one of the important issues in CF is determining user similarity. There is an alternative method for making recommendations which based on item-similarity instead of user-similarity. This approach should not be confused with the approach of Content-Based Filtering. Even though process is called measuring item similarity, it considers all users of the community while computing item similarity, opposed to Content-Based Filtering which considers only individual user's ratings. Importance of item-based techniques is that, researches have shown that item-based recommendations perform better than user-based techniques in terms of accuracy and computation-time [17, 10].

For making the recommendations, according to the algorithm, user-similarity or item-similarity needs to be calculated. Then these similarity matrixes are used in the algorithms for making the real recommendations. In most application, similarities are calculated upon the feedbacks users give to items. Most of the time user feedbacks on items are kept in the n-dimensional vector of ratings of each item in the item set [7]. Then user similarity is computed by comparing these vectors between each other.

Cosine angle is one of the most popular methods for computing user similarity. It is the normalized dot product of user vectors and it computes the angle between the vectors. Pearson is another popular method used in computation of user similarity [23]. Different versions of

Pearson, Entropy, and Mean Squared Difference are other methods for computing similarities. All of these methods consider the similarities upon the rated items of the users. In computation of the similarity of two users, items only rated by both users are used. So for these techniques to be applied efficiently, users need to have many common items that they rated. This is the biggest deficiency of collaborative filtering, because this requirement is not fulfilled in most of the time. Usually, these techniques have to work with sparse-data set; which leads to unreliable user-similarity computations [7].

There are several collaborative recommendation algorithms. In theory algorithms were separated into two categories; model-based and memory-based. However in practice memory-based computations can't scale well with real-world requirements [16]. To reduce time complexity, almost all algorithms have pre-computation phase. Nowadays, this separation is distinguished as both approaches started to mix into each other. Most-of the model based approaches started to keep ratings in memory while doing computations and memory-based approaches started to be built on top-of a probabilistic model. Therefore we will also not make this separation, and explain the important algorithms in this area.

- **User-based Nearest Neighbor Algorithm:** This is the earliest approach used in collaborative filtering based recommenders. It tries to compute similar users, and if a user u is similar to a user n, it is said that n is a neighbor of u. This algorithm tries to predict the rating for an item i for user u depending on the ratings of the u´s neighbors´ ratings on item i. As mentioned above for computing user similarity, techniques such as cosine angle, Pearson coefficient etc. can be used. The details of the formulas of neighborhood formations, user similarity computations can be found in [16]. It has computational deficiencies in practice. Calculating the user neighborhood in an environment where there are millions of users is computationally very expensive task. Therefore to reduce time complexity techniques such as clustering needs to integrated into these approaches. Clustering fastens the process of finding the neighbors of a user by comparing users to groups of users instead of individual user. Because of these deficiencies and difficulties of the user-based nearest neighbor algorithms, item-based nearest neighbor algorithms are tried and it is seen that they perform better than user-based algorithms in many cases [17].
- **Item-based Nearest Neighbor Algorithm:** Similar to the user-based approach, item-based approach generates predictions for ratings of items by users. However this time, predictions are generated by computing the similarities between items. The rating prediction of an item is based on user's ratings for similar items. Similarities of items are computed by using the ratings of users given to the items. For example there are 4 different users; u1 to u4, and 4 different items; i1 to i4. Assume that u2 to u4 has given ratings for every item, and u1 have given ratings to every item except i1. Then a rating for i1 from u1 can be predicted by evaluating the other users' ratings for items. This evaluation would reveal the neighbors of i1 and then according to the ratings given to the neighbors of i1 from u1, a prediction for the rating of i1 would be generated. In theory, the size of the model of this algorithm could be as large as the square of the number of items. However this number can be reduced by only considering items who have more than k co-ratings. This reduction makes this algorithm usable in terms of memory-usage and CPU performance. Similar to user-based algorithm, usage of items with few co-ratings could result into skewed correlations.

- **Association Rule Mining:** This approach builds models on frequently occurring patterns in ratings matrix. It can be seen that particular items have similar ratings from many users, such as users who rated item1 highly, also rated item2 highly. This could be turned into a rule with an input condition ratingOfItem (item1) is high and a result condition ratingOfItem (item2) is high [16]. Each rule is represented by its support and confidence values. Support value represents the fraction of users who have rated both the input and result conditions whereas confidence of the rule is fraction of users that show the result condition by having the input condition. Association rule mining is also highly used for extracting frequent patterns from web logs.
- **Probabilistic Algorithms:** Above algorithms were not based upon probability distributions where as algorithms that will be described in this part built on probability distributions for computing predictions of ratings. Most of these algorithms calculate the probability that given a user u and an item i, the u would assign i the rating of r i.e. p(r | u, i). Then the rating with highest probability would be assigned as the predicted rating of the i. Bayesian Network is the most popular technique used in this area. There are implementations of Bayesian networks with decision trees which performed particularly well [16].Regression is another approach for predicting the ratings of items. One of the advantages of these algorithms is that, it computes the likelihood of each possible rating value. This can be used for checking whether the given rating is correct or not, therefore confidence of the algorithm can be easily evaluated.

Even though each algorithm has its advantages, there aren't one perfect algorithm and according to the scope of the project the proper algorithm need to be chosen. However current algorithms all have common problems that need to be addressed while developing recommenders. Here are the problems and possible solutions that are encountered in CF recommenders:

Common Problems of All Algorithms:

- **Sparse Data Set:** CF needs to have a dense rating data set for making un-biased recommendations. However many applications have sparse rating set; therefore they need to be adjusted while applying CF. For coming with proper recommendations, items and users should not have few ratings. Using these kinds of resources would lead to skew in recommendations; the naïve way for overcoming this is discarding resources with few ratings. This approach is also used implicitly in many algorithms such as nearest neighbor algorithm. However this approach can dramatically reduce the coverage of the system which is another important metric of recommender systems. Adjusting the calculation for rarely-rated resources could increase the coverage while not skewing the recommendations. Sparseness is a fact in most of the applications, famous applications such as Movielens and Eachmovie are 95.8% and 97.8% sparse respectively [23]. Therefore it would be a big mistake not to consider it in development of recommenders. One way to overcome this is; making recommenders that may work with sparse data by exploiting the few ratings that are available. This could be achieved by Trust-based systems where different weighting values could be applied into ratings of users, so trusted users may have high weights in the recommendation process [23].

- **Confidence of the Recommendations:** Each recommendation has a different confidence value according to the algorithms used. These confidence values need to be used for filtering out the recommendations with low confidence value. Also confidence value can be used to choose the item to be recommended, instead of recommending the item with the highest rating but low confidence.

- **Gathering of Rating:** Collaborative filtering needs user feedback in order to make recommendations. So far, in most systems only ratings are used in terms of user feedback. However as it is mentioned in the earlier sections, gathering explicit feedback from users is a difficult task. Therefore other techniques of gathering user feedback are needed such as implicit feedback. There is a tradeoff between explicit and implicit feedback. Explicit feedback is the most reliable source for building user models however giving them is an overhead for users. This limits the size of explicit feedback collection. On the other hand implicit feedback gathering costs nothing and huge amount of implicit feedback can be collected. But the accuracy of the user models based on only implicit feedback is very low. Therefore aggregation of these techniques needs to be used as user feedback. [20, 21] showed that by using voting and averaging schemas different data sources could be aggregated into a single user feedback estimation which reduces the uncertainty in user models. Another solution for directing people in giving explicit feedback could be by showing the benefits of it to them. In [22], it is proposed that, users are getting more and more open to giving user feedback because users feel as a part of a community when they contribute to it, they feel satisfaction when they see that their opinions have an effect on the community, and they started to use these systems as a memory-based system to track what they like and dislike.

- **New User and New Item Problems:** Similar to Content-Based Filtering, collaborative filtering also cannot handle cold-start problem such as recommendation to new users and recommending new items. Actually here the problem is even more severe; because of the user-feedback based nature of the CF. Some systems come up with some temporary solutions to overcome these problems. For example when a new user enters system; some applications don't provide them recommendation feature until they rate some predefined number of items. Or even more, in the registration phase, they force users to rate some items. Of course these approaches can lead some users to walk off from the system. The biggest challenge is boot-strapping of a new application, because with new systems, everybody is new users and all the items are new items without any ratings. The easiest solution to this, before releasing the application, open the beta version of it to a small community and gather feedback from these users until the system is ready to be used in larger communities. Another solution could be using different recommendation techniques such as demographic filtering until sufficient number of ratings is collected to use CF in the system.

- **Trust Issues, Malicious Users:** As CF is based on user opinions, many applications have public services for gathering user opinions. This enables anyone to join to a community and comment on the items as they want. However this makes system vulnerable against attacks. Especially in e-commerce applications, it is seen that many companies started to use their own agents for increasing the ratings of their items [23]. Because everyone is treated same in these systems and there are no authorities, malicious users can easily harm the stability of the rating-set. One possible solution to this is not considering each user same, but determining user authorities by analyzing transactions and giving more weights to feedbacks of these users. However trust issues are started to be realized recently as recommender engines were not so popular in the past, therefore this issue still needs further investigation in research community.

- **Black Box Problems:** As it is the case for most of the recommenders, collaborative filtering recommenders are also implemented as black boxes where users cannot understand the reasons behind the recommendations and cannot control the recommendation process according to their preferences. [24, 25] reported that users just stop using recommendations after keep getting bad recommendations and not even seeing the reasoning behind it. Amazon started to give explanations why users receive particular recommendations, and this is a good start for creating a communication between users and recommenders. Other possible solution is to adapt the recommenders according to the users' relevance feedback or provide users capabilities to modify the recommendation process according to their preferences. One another example is Facebook where users could specify what kind of items (pictures, videos, news, etc) that they want to see and from which users they want to receive news about.

After giving the problems of CF techniques and possible solutions to them, the advantages and disadvantages of the CF will be given in the following parts.

Advantages of Collaborative Filtering Algorithms:

- CF makes its suggestions based on user opinions, therefore it is more similar to real suggestion process occurs between people. This leads to novel recommendations that attract more users.

- Scalable to work with huge item sets, because it doesn't need manual intervention to make suggestions.

- It doesn't need a representation of items in terms of features because it makes recommendations based on user opinions. This makes it applicable to any kind of item.

- Over-specialization does not occur, because recommendations are not based only on individual users but they utilize other people's information too.
- Applicable to reveal social relations inside communities such as forming networks of users.
- Applicable to reveal hidden interests of users by recommending items from different users.
- It is easier to adapt the recommendation strategy according to user preferences by letting users to choose other users who they trust more. Stumble is one example of this approach where users can specify their friends or other users who post interesting items, and their recommendations can be based heavily upon these specified users.

Disadvantages of Collaborative Filtering Algorithms:

- It relies on user feedback and is useless in environments where users are not prone to give feedback.
- Cold-start issues also exists in CF and even more difficult to overcome it in a pure CF recommender.
- Open to attacks by malicious users who can give fake ratings to increase the popularity of specific items.
- Fact of sparse data sets makes pure usage of CF inefficient; because popular user similarity computation techniques such as Pearson, cosine metrics require dense data sets.
- Most of the recommenders are black boxes where users cannot control the recommendation process. This leads to user dissatisfaction when users keep getting bad recommendation, even there is a chance to get better recommendations by

Even though Collaborative filtering has several disadvantages in theory, they are still performing quite well in practice (Amazon, Last.fm, etc.). Therefore most of the current applications use collaborative filtering approaches in their recommendation process. However there are other proposed techniques for overcoming deficiencies such as cold-start problem. One recent technique on the horizon is case-based (knowledge-based) reasoning. In the next section it will be described briefly.

2.2.3 Case-Based (Knowledge-Based) Reasoning

Case-based recommenders have a similar working philosophy to content-based recommenders. They also need a structured representation of items in terms of features for making recommendations. Then recommendations are made upon the similarities between the items and queries of users [58].

Case-based reasoning techniques are the building blocks of these recommenders. Early systems were used for problem solving and classification tasks. They were different than traditional techniques in the sense that, they were based on problem-solving experiences rather

than theoretical models [59]. So essentially they were using a database which keeps the solutions of past problems. Each problem was stored in two parts; specification and solution part. Specification part describes the problem and solution part describes how to solve the problem. Then new problems' specification parts are compared to specification parts of the cases in the database. The most similar problem is chosen and its solution is adapted to the current problem.

Therefore items are represented as cases and they are compared to the user's profile or query. Items or cases which have similar descriptions to the user's query are recommended to users. Even though it seems very similar to content-based reasoning; it has two main differences than content-based filtering that are the way items are represented, and the similarity computation of items.

Content-based recommenders represent items in an unstructured or semi-structured manner. Representation of news articles are made by keyword matrixes. However case-based recommenders rely on structured representations of items in terms of well-defined feature sets. This makes them applicable to be used in many e-commerce applications because these applications have already had representation of many products in terms of well-defined features. One example could be travel packages which can be represented in location, price, accommodation, etc. Each of these features can contain several values according to the used value-space (integers, real numbers, pre-defined string values, etc.) In our application for the knowledge component part we defined promotions in terms of destination, price range and the dates that the promotion is valid.

Another advantage of case-based recommenders is that they are suitable for making reasoning in recommendations which is one of the important problems of recommender engines. As case-based recommenders determine recommendations based on the structured feature sets; they can inform users why they receive these recommendations.

So it can be seen that case-based reasoning recommenders could be quiet useful especially in e-commerce applications. Downside of this approach is that; it requires manual intervention especially for determining similarities of feature values (However studies are performed to apply machine learning techniques to make this process automatic). Other problem is; pure usage of it turns the recommendation process into retrieval process. Therefore in the development of recommendation systems; it is mostly used along with another recommendation technique such as collaborative filtering. This combination of different techniques results in hybrid recommenders, and they will be described in the following sections.

2.2.4 Hybrid Recommenders:

Hybrid recommenders combine more than one type of recommendation strategy in the recommendation process. The main idea behind this approach is combining several techniques in a complementary way such that limitations of each technique would be handled by the other recommender strategies. So it tries to create a synergy between the used recommender strategies to improve the performance of recommendation process.

There are common problems of many of the recommendation strategies such as cold start problem which can be seen in a form of new item or new user problem. Another problem is adaptation to changes in the user preferences. It is difficult to reflect the changes in the user preferences to the recommendations, especially in learning based recommender strategies such as collaborative filtering. It can be said that traditional recommenders are not dynamic and flexible enough to cope with the changes in the preferences of users. One solution to this deficiency is combining knowledge-base component into recommendation system, which responds to user's

immediate needs instead of relying on the historical data of users [32]. Therefore the main goal of the hybrid recommender is to eliminate this limitations and deficiencies of pure recommender strategies by combining their powers.

In the development of hybrid recommenders, there are two crucial decisions to make, first one is to choose the recommendation strategies that will be part of the system, and second is determining the combination schema of these approaches. Most of the hybrid recommender systems are composed of content-based and collaborative filtering [14, 29, 33, 34, 38 and 40]. However other techniques such as demographic filtering [36], case-based reasoning [35] are also started to be used in hybrid recommenders. Web-based applications also integrate web mining into the recommendation process. One of the most popular approaches is using web mining results in collaborative and content-based filtering techniques to recommend items to users [7, 27, 37 and 39]. Most recently some studies are also made to incorporate contextual information, ontology into hybrid recommenders [6, 10 and 18].

So it can be seen that, hybrid recommenders can use of content-based, collaborative, demographic filtering, case-based techniques, web mining, and contextual information while making recommendations. However, even more important design decision is determining how to combine these approaches. There are several ways to perform hybridization:

- **Weighted:** In this approach, each recommender technique gives a score for the items and these scores combined in a linear formula. This is the simplest design for a hybrid system.

- **Switching:** In switching, each time a recommendation needs to be made, one of the recommender is chosen according to some criterion. This criterion could be the confidence value of each recommender, and the recommender with the highest confidence value can make the recommendation.

- **Cascading:** This approach is a hierarchical model where there is one primary recommender and rest of the recommenders are considered secondary. Primary recommenders make the recommendations, but in case of a tie score between the items, resolution is made by applying secondary recommenders. Because most of the scoring functions are real-value, in this approach secondary recommenders don't have too much role.

- **Feature Augmentation:** Here there are again two types of recommenders; primary and secondary. However here secondary recommenders generate a new feature for each item, and augment that to the data which will be used by the primary recommender. Therefore here secondary recommenders are much more active and their role is improving input data so that primary recommender can make better recommendations.

- **Mixed:** Here there is no hybridization of methods, but each recommender makes their own recommendations, and all of these recommendations are presented to the user.

In [35], it is shown that, cascading and feature augmentation hybridization techniques give the highest accuracy results in recommendations; while weighted technique gives poor results. The reason of poor performance of weighted approach could be explained by considering the fact that; each recommender chooses different types of items to recommend according to their working style. Therefore there is a very small subset of overlap items between recommenders which is not a reliable basis for making weighted approach work.

Determining the recommendation strategy and hybridization techniques depends on the context of the application. However results of [35] can be used as a good starting point for making these decisions. Also many papers empirically show that hybrid methods provide more accurate results than pure recommenders [14, 32 and 41] which are a good motivation for developing hybrid recommendation systems.

Before giving the examples project about recommendation engines one another important component of recommenders needs to be analyzed. It is web mining which is especially useful in gathering implicit feedback from web logs. Most of the time web mining is used in combination with recommendation techniques such as the ones mentioned above. Especially after seeing that retrieving explicit feedback is not so easy in practice, importance of web mining for revealing hidden trails that users left behind while using web sites increased. Therefore many researches are going on the field of web mining for improving the techniques to gather more information from web logs. In the next section web mining will be explained, but we will focus on web-usage mining which is especially used for collecting implicit feedback from web logs, therefore other types of web mining such as content mining, structure mining will not be part of next section.

2.3 Web Mining: Web-Usage Mining

As the resources in internet grow with each day, opportunities for applying data mining on these resources are also increasing daily. Applications of data mining started to be called as Web mining which is a cross field where researchers from different areas are studying together. Web mining can be divided into three categories, content mining, structure mining and usage mining. The main focus of this section will be usage mining; but for giving a complete coverage, brief information about content and structure mining will also be given.

So far web mining is only used for gathering statistical information about usage of web sites. This information is used to leverage the server traffic or re-structure the web site for attracting more users. Web Analytic tools are mostly used for these purposes, but with the increased usage of recommendation engines, it is seen that more elaborate usage of web mining can be useful in recommendation process. Therefore it started to attract interest from social topics such as marketing, consumer behavior analysis beside technical fields such as database, machine learning communities. Diversity of areas interested in web mining shows the importance of it. Proper usage of web mining can be used as a solution to many different problems of both parties of internet that are information providers and end users.

From the information provider's view, web mining could be used for learning information about consumers. They want to know what customers do in their web site, and what they want from them. Mass customization of information to the individual user is required. This can be achieved by applying personalization using web mining techniques. Information providers could also use web mining in marketing, design and management of their web sites. Web analytics is one of the areas that attract interests of many information providers.

Web analytics tools can help them identify how their site is used, when are the peak times, how can they better structure their web sites so that user can easily access information they are looking for. Recently these tools started to be used for evaluating whether the advertisements companies are using is really useful or not. This can be done as web logs keep information of from where the users come to the sites. For example they can check whether users came to their sites from an advertisement link that is shown in Google and increase or cancel the advertisements they are using according to the usage of these links.

Users encounter with different problems while surfing in web and web mining can also provide a solution to these problems. Here are these problems [1]:

- **Low Precision:** Getting low ratio of relevant items while making search. Users encounter with many irrelevant items while making search.

- **Low Remark:** Users are not able to get most of the relevant resources while searching information in internet, because they are not indexed by search engines. So users can not find un-indexed information.

- **Extraction of Knowledge out of Available Data Set:** Users wants to gain new information from the data that they already have.

- **Personalization:** Users wants web sites to be adapted to their preferences, and they want to retrieve information that is relevant to their needs.

These problems could be solved by applying different web mining techniques in combination with other techniques such as information retrieval, natural language processing etc. For example web content mining, could be used in the extraction of knowledge problem, whereas web structure mining can be a remedy for retrieving relevant information, and personalization issues can be resolved by web usage mining.

Web content mining is the process of discovering useful information from the content of the web pages which could be plain text, images, videos etc. [1]. Because most of the data in web is unstructured data, content mining can be seen as an example of data mining techniques such as knowledge discovery in texts. Goals of content mining are to improve information finding, develop better models for the data on the web to improve search engines.

Discovery of the models of the link structures that forms the topology of the web is named web structure mining [26]. Categorization of the web pages according to these models can be made which helps finding similarities and extracting relations between web pages. Authority and hub pages could be identified using this technique, or the web site topology could be re-structured for improving the convenient usage.

Both web content and web structure mining utilizes the main data on the Web which can be considered as the primary data. Web usage mining is different from the both methods in this perspective, as it doesn't work with primary data but tries to extract information from the secondary data. By secondary data we mean the data that is generated by users while navigating in the web sites. This data mostly includes information such as IP address of the user, page visits, mouse clicks, time of entry, etc. Web logs are the main data source for web usage mining, but it combines different data sources such as browser logs, user profiles to generate useful information.

Most applications use all three methods in combination, as they are not very clear cut between the categories [1]. However from the recommender engines perspective the most useful category is web usage mining, because by using that very important information such as frequent navigation patterns, association between pages could be extracted.

In the rest of this section we will get into the details of web usage mining, techniques used in usage mining will be given and we will describe the three phases of (data collection and pre-processing of data, pattern discovery and pattern analysis) usage mining.

The main goal of the usage mining is to extract usage patterns from web logs for improving the usage of web-based applications. This improvement could be made in terms of re-structuring

of the site to personalization of information according to individual users. This could be achieved by analyzing web logs that keeps information as detailed as mouse clicks, scrolling events of users which reveals information about behavior of users.

Web usage mining applies data mining techniques to these web logs to extract valuable usage information. Usage mining requires three steps for extracting information that are data collection and preprocessing of the data, pattern discovery and pattern analysis. Here these three steps will be elaborated.

2.3.1 Data Collection and Preprocessing:

As collaborative filtering needs user feedback to make recommendations, web mining requires web data for extracting usage patterns. This web data's source can range from web servers to client servers or from proxy servers to databases. Each source keeps different type of information such as some of them keeps information about content of web pages, some structure and some usage information. Web usage mining mostly interests in the data resides in server and client logs.

- **Data Sources:**

Data collection from server sides has potential problems. Much information could be missed according to the level of caching. Pages that are loaded from caches are not stored in web servers. Also information can be passed through POST or GET method. GET method don't create any real problem but information passed through POST can't be stored in server logs. By using cookies users can be identified which is very important in recommendation process. However because of privacy concerns many users don't allow cookies to be used in transactions, therefore user identification become main issue in usage mining [1]. This problem is also related with session identification. Session is described in [28] as a sequence of requests made by a single user during a certain time period to the same web site. Apart from cookies query data can be also retrieved from server logs which might be used in user modeling.

Client side data collection can be seen as an alternative to the server side data collection. It is mostly implemented by using remote agents such as Java Script; however this again requires permission from users. Nevertheless, client side collection has particular advantages over server side collection. Client data can cure the problems of caching and session identification. Therefore client side data collection can be very useful for preprocessing of data which is responsible for cleaning of noise data, session identification etc.

Web logs could also be strengthened by integrating data collected from CRM and ERP systems for gathering business intelligence [2].

- **Preprocessing**:

Preprocessing of web data is the most important and time consuming phase of the web mining [28]. Preprocessing can be defined as converting the unstructured data resides in data sources to structured data for making it available to use in pattern discovery.

This structured data can be interpreted as firstly conversion of raw data into page visits, then clustering these page visits to sessions or episodes according to the required data abstraction level. Since it is not possible to track page requests from different servers, for session identification a thirty minute timeout assumption is used [30]. This means that if the duration between two consecutive page visits from the same IP address exceeds thirty minutes, those two page visits belongs to different sessions, otherwise they belong to same

session. Also as can be understood from the above statement, page requests from same IP address assumed to be coming from same user which is not always true; because of proxy servers.

If possible user identification is also made in this phase. This is a straight-forward task if cookie usage is enabled; otherwise it is an almost impossible task to do because of the stateless nature of HTTP [3]. Similarly client-side data collection also converts user and session identification into trivial tasks.

For easing the job of pattern discovery, content preprocessing can also be applied in this phase. In this phase classification and clustering are performed to make the content data such as images texts more useful to next phase. The same procedures could be applied to structure data.

Another important activity in preprocessing is data cleaning such as feature selection and record removal. Web logs contain much information which might not be very useful in some situations. Reducing the number of features, decreases the memory usage in the following stages [28]. Generally following fields are essential for usage mining; page ID, IP address, and page request time. Other fields are needed to be chosen according to the type of the application, for example recommender engines could use the referrer field in user modeling, and therefore it is smart to add that field into feature set.

Web logs also stores many irrelevant pages because of the structures of web pages. Examples could be the image files that are loaded when a user visits a page. In [31], it is stated that irrelevant files in a web log are found in a ratio 10:1. This may not be very important according to the type of the project, but crawler activities stored in the web logs needed to be cleaned out in most of the applications [2].

After cleaning of the data, and identification of sessions from the web logs, processed data is ready for using in the pattern discovery phase. In the next section, how this data is used to extract navigational patterns will be described.

2.3.2 Pattern Discovery:

Pattern discovery is the process of applying data mining, machine learning techniques on the web logs to extract information about web sites. These techniques could be used for simple tasks such as creating statistical reports to generating association rules. In this section we will explain the popular techniques that could be used in this phase of the web usage mining.

- **Log Analysis**

Log analysis can be seen as the simplest method of web usage mining. After preparing raw data to be analyzed efficiently, statistics about web site could be generated. This statistics could be about site activity, diagnostic statistics, server statistics, user demographic etc. Some instances of these activities are total number of visits, average view time, server errors, top referring sites, most active countries in the site, visitor's web browsers, etc) [2]. These statistics mostly presented to the administrators in a report format, so that they can modify the web site for improving the performance of it.

- **Association Rules**

This technique can be used for finding frequent patterns, relationship between items. Click-stream analysis is used for identifying the paths used in the web site. Then association analysis on these paths could reveal associations between pages. It could find out which

pages are visited together and it can reveal association between pages which don't have connection by hyperlinks. Then by considering these rules, web designers can re-structure the web site to increase the usability of web sites.

- **Sequential Pattern Discovery**

It can be regarded as extension of association analysis. It is used for revealing patterns such that set of pages is visited after visiting some other pages in a time-ordered style [3]. One application of this approach could be placing advertisements at particular pages according to the navigation path of the current user. Trend analysis, predications about visit patterns could be made using this method.

- **Clustering**

Clustering is a very famous technique used in machine learning for grouping a set of items that have high similarities. In web usage domain, clustering can be applied for clustering of users and pages. User clustering can consider user's navigation behavior, demographic information while putting users into corresponding clusters [2]. Demographic information can be used for market segmentation whereas according to the navigation behavior, users can be recommended with pages that are relevant to them. On the other hand clustering of pages can be useful for search engines, as it will group pages with related contents into same clusters [3]. Clustering is very important for web personalization as it groups users according to their preferences, it can also be used for reducing the time-complexity of recommendations by applying techniques on corresponding clusters instead of the global model.

- **Classification**

Classification is almost same as clustering but here there are predefined classes where as in clusters there aren't any predefined clusters, but clusters formed on the fly. In the web, classes usually represent user profile belonging to a particular category. Therefore extraction of features that best describes the properties of a class is very important. There are many techniques for classification such as Decision Trees, Bayesian Classifiers, Nearest Neighbor Algorithms, Neural Networks and etc.

After discovering patterns from the web logs by applying chosen techniques the last step of the web mining process which is named pattern analysis should be executed. Main purpose of this phase to remove the patterns those are not very interesting or not useful for the application. Most popular analysis methods are using query mechanisms such as SQL or OLAP tools.

2.3.3 Remarks and Conclusions for Web Mining

Web mining is very important in the process of web personalization. Because it reveals the user's hidden preferences and interests besides the trivial ones as it uses the navigational behavior of users. Also it is very useful for improving the performance of web sites, even only by making statistical analysis.

User profiles can be generated implicitly and explicitly by using the web logs, and databases of web sites. Explicit information from registration forms can reveal personal information from users such age, nationality, sex whereas web logs can store hidden information about user's navigational behavior. These sources should be combined while generating user profiles. User profiles should be matched with the rules, patterns that are found out by applying machine learning techniques such as association analysis, clustering, etc. OLAP tools and visualization

techniques should be used for refining and filtering out the patterns found in the pattern discovery phase.

Then user profiles and refined rules could be used for personalization of web. Dynamic recommendations can be provided to users according to their profile, current navigation behavior and context. This would increase the user satisfaction, as users would use web sites that are adapted to their needs. On the other hand, web usage mining can be very valuable for market segmentation and can provide competitive advantage to the applications. Because it helps companies to acquire business intelligence by providing knowledge in the form of rules and patterns that describes the navigational and purchasing behavior. Therefore from the perspective of both parties' of internet (information provider and users) web mining is a very powerful method to improve the satisfaction of parties besides improving the quality, usage and performance of web applications.

So after giving general overview of methods and techniques used in recommendation systems, the related work that have been done so far in these fields will be given in the next section.

2.4 Related Work

In this part we will summarize the work that has been done so far in the area of recommender systems which will include sub topics such as content-based, collaborative filtering, hybrid approaches, web mining applications, surveys about these subjects etc.

In [1 and 28] detailed survey of web mining research have been given. Authors in [1] deal more with content mining and structure mining where as [28] focuses on web usage mining. Again [1] tries to resolve the term confusion occurred in web mining because of the fact that there are many fields such as database, information retrieval; machine learning, etc. interested in web mining and each area puts different meanings to the same terms according to their field. This creates confusion in some readers' minds and [1] clarifies these term confusion. It also focuses on the issues of representation of data, process of web mining, applications, and describes the learning algorithms that are used in web mining.

Whereas [28] gives a comprehensive overview of web usage mining and don't interest with other types of web mining. It focuses on the difficult tasks of web usage mining such as data cleaning, user identification and finishes with giving useful web log mining software packages. In the paper, algorithms such as association analysis, apriori algorithms, clustering techniques are also described, and it explains the importance of OLAP and visualization tools for improving the performance of web usage mining; as they make the analysis of web logs much easier. Last but not least it emphasizes on the importance of incorporation of web mining into recommendation process for improving the performance of recommenders.

In [41] general survey about the content-based, collaborative and hybrid recommendation approaches are given. It explains the limitations of each method and provides solutions to these problems. Possible solutions they proposed includes incorporation of the contextual information into recommendation process such as considering weather conditions while making recommendations about traveling. Using multi-criteria ratings instead of unary rating systems and relying more on implicit feedbacks instead of forcing users into giving explicit feedback is another method they propose to improve recommendation process. These proposed solutions are also explained in more details in their other paper [6].

In [32], the comparison of hybrid recommenders has been made. The hybrid recommenders were composed of different combination of four different recommendation

techniques that are content-based, collaborative, demographic and knowledge-base. It also uses seven different hybridization strategies which results into 41 different hybrid recommenders. According to their results, they concluded that the best hybridization strategies are cascade and feature augmentation. However in their evaluations, their main focus was how to include knowledge-based component into hybrid recommenders in the best way, therefore their measurements are biased towards seeing the effect of knowledge-based hybrid recommenders.

Authors of [54] give a general overview of web personalization from the perspective of businesses and developers. They give definitions of what is personalization and shows the three high-level personalization architecture which are provider-centric, consumer-centric and market-centric. Process for the personalization which consists of understanding consumers (by means of data collection and user modeling), delivering personalized offerings (by finding relevant items corresponding to the user models) and measuring impact of personalization (by evaluating the personalization in particular metrics and making corresponding modifications to personalization strategy) are given. The main idea is personalization process is an iterative process where all these three steps affect each other in a feedback loop. One of the most crucial criteria for the personalization process is given as personalization strategy needs to adapt itself to the constant changes in customer preferences. Also good metrics are needed for evaluating the personalization strategy, behind the philosophy of you can't manage what you can't measure. Therefore a goal-oriented approach needs to be used in the personalization process which should start by determining metrics to be used in evaluation. Then according to these metrics personalization offering that will be applied in the system should be determined. User modeling and recommendation strategies that will be used in the delivery of these personalization offerings needs to be chosen, and finally relevant data needed for doing these tasks should be collected. However most of the applications are data-driven which applies these steps in the opposite order.

These surveys are very useful for getting a general overview about recommender systems. In the following parts we will get into details of the related work that is done for content-based, collaborative, hybrid recommenders and web mining approaches. Our goal is to show what has been done so far in these fields, point out the deficiencies of current systems and we will present our solution to these deficiencies by explaining the structure of our recommender system and techniques that will be used in our work.

2.4.1 Related Work on Content-Based Approaches

Pure content-based recommender systems have been widely used in the past especially in document retrieval systems, but as recent developments revealed that these recommenders lack social relation aspect in recommendations and web becomes more and more a social network, most of the systems combine content-based filtering with some other technique such as collaborative filtering to improve their recommendations.

In [42] authors built the system PRES which uses a content-based filtering technique for recommending articles about home improvements. In their model, they had used positive feedback to dynamically update their user profiles and used the Rocchio algorithm as a learning algorithm. Then according to the content of the articles and user models, recommendations have been made. In the recommendation process, they used stemming techniques to improve the accuracy as they used tf-idf methodology for computing word frequency. Cosine similarity is used for computing the similarity of users and documents. However system could only make %50 relevant recommendations, because they didn't take into account the synonym analysis as many concepts can be explained by different terms. They also accepted that recommendations

could be improved by using collaborative filtering as content-based filtering cannot predict future interests of users. And in their future works they proposed to combine collaborative filtering and web mining into process.

Pandora is an automated music recommendation and internet radio services created by the Music Genome Project [43]. They made music recommendations according to the content of the music songs and user feedback. More than 400 musical attributes such as whether piano is played or not, rhythm of the songs is used for making recommendations. However this system lacks from the over-specialization as users keep listening to similar music they have rated high. For example if the user only rated a song of Beethoven high, then the users would only get recommendations of classical music, even though user could have different tastes in music. Therefore recently Pandora also started to combine collaborative filtering in their recommendations too to solve overspecialization problem. Another problem of Pandora is separation of songs according to these 400 attributes required manual intervention from domain experts. Therefore it is not feasible to use this system with huge amount of data; which is another fact that attracts developer to use collaborative filtering in their systems to improve the scalability.

2.4.2 Related Work on Collaborative Filtering Approaches

On contrary to content-based filtering, there are many applications and studies based solely on collaborative filtering, as it is much easier to implement collaborative-based systems because it has many advantages in scalability and it has higher accuracy in recommendations.

Incremental collaborative filtering approach is proposed in [44] for making highly-scalable recommendations. Its main goal is to increase the scalability of traditional CF methods for improving the computation performance of recommenders. They suggest improving the efficiency by using incremental updates of user-to-user similarities which is not based on approximation approaches. They think that the scalability issues raises because traditional CF methods based on approximations mechanisms which are costly in computation. Traditional methods have $O(m^2n)$ computation complexity where m is the number of users and n is the number of items. For reducing complexity many systems have an offline and online phase in recommendations where in offline phase the similarities of users are computed, then the recommendations are made in online phase based on to these similarities. Recommender engine in [44] has a computation complexity of O(mn) by using incremental collaborative filtering. Their experiments showed the computation performance of the system is times higher than traditional methods and accuracy of their system is also %100. However their system also has problems dealing with sparse data sets and they propose to use trust analysis for handling sparse data sets.

In [19] authors propose to implement a collaborative-based recommender which uses contextual information also. In this paper, how the context could be used in recommendation process is investigated. Here they also provided the general architecture for a context-aware recommenders system. In their architecture they also added a module for making explanations to users about the recommendations. They left the implementation and evaluation of the architecture as a future work.

After giving a critical review of personalization in e-commerce, [10] gives a general architecture that could be used to improve personalization in e-commerce applications. This architecture tries to capture the following features; automatic identification of users, composition and presentation of content to users according to the model that is based on their preferences, and

making recommendations based on the interaction contexts. In their paper they used an item-based collaborative filtering approach and user models are represented as semantic networks. The paper gives a very good overview of the challenges in the personalization of web content, and a generic architecture for e-commerce applications.

Authors of [45] developed an adaptive recommender system for travel planning which based on reinforcement learning techniques and conversational approaches. They proposed architecture for their systems and left the evaluation of the performance of their implementation as a future work. For user modeling they combined explicit and implicit methods. Instead of following non-interactive approaches which is used widely in travel recommendation sites, they used a conversational recommender systems where at each stage of the navigation of the users, according to the actions of users, they either asks additional information from users or recommend some product. This choice is made based on recommendation strategy, which tries to optimize the recommendations. This optimization is done by maximizing a reward function that models how much the user gets benefit from each step.

2.4.3 Related work on Recommender Systems Using Web Mining

[40] applies web mining for web personalization. They use static information of user by applying clustering and dynamic behavior of users by pattern recognition to apply re-classification algorithm. The idea of re-classification algorithm is classifying web pages according to the type of users accessed them and then re-classifying the users according to class they have assigned previously and the resources they have accessed since last classification. So it could be seen as dynamic integration of user navigation behavior to history of user transactions. Dynamic behavior could be captured by means of submitted queries, traversed pages etc. They have developed an architecture which combines web mining activities and content of web page with recommendation strategy. According to their evaluations with a commercial web site, users are eventually classified into proper clusters which improve recommendations.

In [46], recommendation algorithm for meeting the personalization needs of e-supermarket applications has been proposed. This algorithm combines collaborative filtering with web usage mining and it uses decision trees. Their algorithm consists of four steps which are determining active customers (decision tree induction), discovering product connection (association rules are generated by making basket analysis), discovering customer preferences (analysis of previous shopping behavior by using web mining) and making recommendations (applying collaborative filtering). They also implemented PRSSES which is a recommendation system based on E-supermarket model and comparison of performance is made between the models that are using SVM classification algorithm, content-based algorithm based on VSM, Rocchio, k-nearest neighbor and algorithm of the paper. The comparison is made using recall and precision metrics. Authors claimed that algorithm presented in the paper gives the best performance without explaining the evaluation procedure which lowers the reliability of the results.

Web-Usage mining based recommendation system is developed in [47]. They used longest common subsequence (LCS) algorithm for classifying user navigational patterns. Their model consists of traditional two-phase model, which consist of offline and online phase. They focused on improving the online module by using LCS, and in offline model they follow the traditional methods for data preparation which includes, data pre-processing, pattern finding and analyzing. In their experiments they used K-fold cross validation method to test performance in accuracy, coverage and F1, F2 metrics. According to these experiments their accuracy increases up to 70%,

but they didn't compare their results with other algorithms which is required for understanding whether LCS improves recommendations or not.

Dynamic personalization is performed by system named SUGGEST in [57]. SUGGEST is proposed as a novel recommendation module in sense that it uses one online single module instead of standard two modules with offline and online components to make recommendations. It relies on web usage mining for making page link recommendation. It is based on an incremental personalization procedure which is tightly coupled with Web Server. Users are clustered according to the navigational behavior and according to the found frequent patterns, users are suggested page links. Usage information is represented as a graph. It has one limitation that it doesn't consider the content of the web pages but relies only on to the order of page visited in a session for determining user interests. Incorporating content of resources to determination of user interests can increase the accuracy of system.

[29] tries to improve the accuracy and computational performance of recommenders by discovering association rules from usage data. It tries to provide a solution to problems of sparse data, and scalability issues of collaborative filtering by analyzing click-stream data. In their paper they designed a data structure for storing frequent item sets which would improve the performance of recommender. Discovered item sets are stored in a Frequent Item set Graph which is an extension of lexicographic tree. The recommendation engine takes this graph as an input and matches it with the user's activities to generate recommendations. In the implementation, fixed-size sliding window is used to capture the actions of users. The performance is improved as the recommendation algorithm creates recommendations from items sets directly before generating the association rules. Experiments showed that this approach outperforms k-nearest neighbor algorithm in precision and coverage while improving the computation time.

Similar to the [40] [39] makes recommendations based on the navigational patterns of similar users. But instead of relying only on user navigational behavior, it also incorporates conceptual relationships between web documents into process. They used this approach in their web personalization system SEWeP. In their paper they tried to solve the problems of recommendations not considering semantic information of items and new-item problem. They used domain ontology in combination with users' navigational behavior for producing recommendations. They also explained the architecture of SEWeP which covers semantics characterization of web content and combination of this content with web usage data. They supported their methodology by applying blind tests to users.

Authors of [48] improve Markov Models in modeling of user's navigational behavior. Markov models traditionally only relies on usage data in prediction of user navigation. In this paper, Markov models are integrated with link analysis which uses page-rank style algorithm. This algorithm is used for assigning prior probabilities to web pages based on the importance of them. The deficiencies of Markov Models are mostly due to assignment of initial probabilities; for example they assign initial probabilities uniformly or proportional to the times each page visited. But these assignments lead unfair weighting between pages, because each page should have an initial probability proportional to its importance in the web graph. To overcome these problems, this paper proposes to use a page-rank style algorithm which assigns the prior probabilities according to the structure and navigational patterns of the web site. Their evaluations showed that their idea of assigning prior probabilities according to these criteria really improves the accuracy of predictions.

Web usage mining

[2], [3] and [55] gives general overview of application of web mining in web personalization. They describe the steps of web mining and explain how web mining can be exploited for adapting information of web according to the needs of users. [2] gives the modules of a web personalization system which covers web usage mining (user profiling and log analysis), content management, information acquisition and searching and web publishing. Whereas [3] explains the application areas of web usage mining which could be personalization, system improvement, business intelligence etc. But it also mainly focuses on web usage mining and gives the overview of WebSIFT which is designed to perform web usage mining techniques that they describe in their paper. [55] solely focuses on how to apply web usage mining in automatic personalization of web pages. It gives a detailed explanation of steps of usage mining that are data preparation, discovery of usage profiles, patterns and making recommendations out of these profiles and patterns. All of these papers are very useful in getting a general overview and learning the details of web mining.

2.4.4 Related Work on Hybrid Approaches

Instead of making recommendations solely on user ratings, [7] provides another sources for improving collaborative filtering. They integrate user ratings with item ontology in recommendation process. They also evaluated their system using a movie web site which showed that their methods outperform in accuracy the traditional collaborative filtering method. It uses item ontology for inferring the user preferences, and then uses collaborative filtering to generate recommendations. New metrics such as measuring similarity between users within item ontology and items within taxonomy is tried as a solution for sparse item-rating space. Drawback of their solution is that, computational complexity of their algorithm is $O(n^2)$ where n is number of items rated in a single visit. This in addition to fact of expensive item similarity computation prevents their algorithm to be applied in practice.

In [6] authors come with different methods to use in recommendation systems. Instead of following traditional one-dimensional rating systems, they propose to use multi-dimensional approach. In addition to this, usage of contextual information besides the typical information about users and items is also offered in this article. They also present a rating estimation model for multi-dimensional approach and an algorithm to combine multi-dimensional and two-dimensional approaches. The motivation behind using contextual information rises from the different requirements that need to be addressed in different contexts. For example while making a recommendation to a single user, consideration of only user preferences are enough, but recommending a movie to a couple on the valentine day has different requirements. Authors try to achieve better recommendations by incorporating different contextual information such as date, place, and companion into recommendation process.

Similarly multi-dimensional rating system is needed for handling importance of different perspectives of the items. For example a user can be interested in the success of the actors more than the general success of a movie. This paper addresses these issues in great detail, and proposes very novel techniques to use in recommendation models. It also gives a general survey about the traditional recommendation methods such as collaborative, content based, hybrid recommenders, user, item similarity functions, etc. OLAP tools are used for representing multi-dimensional rating systems and hierarchy of context information (time dimension can be separated into three categories such as weekday, weekend, holiday). It is highlighted in the paper that, the dimension selection is very important and needs to be chosen according to type of applications. Trivial dimensions which don't affect the recommendation process because they

have similar ratings from every user need to be discarded to fasten the computations. Because there are well-established methods for calculating two-dimensional ratings, they proposed the reduction-based method for reducing n-dimensional ratings into two-dimensions.

Then they proposed a combined reduction-based collaborative filtering which uses either pure two-dimensional approach or apply reduction for converting n-dimension to two-dimension. In pure two-dimensional approach the two-dimension out of n-dimension is chosen for making recommendations. Whereas in the reduction-based approach instead of choosing the best two-dimensions, the ratings for n-dimensions would be aggregated to two-dimensions. Then their approach would choose either pure method or reduction method according to the expected performance. Theoretically this hybrid approach should perform equally or better than pure approach as reduction based approach is only used if it performs better than pure approach. However in practice this may not be true as performance depends on the data.

Decision support metrics are suggested to be used in recommender systems because they measure how well a recommendation system can predict which items will be highly rated. This is better suited for recommendation systems; because they focus on recommending the best available items. They evaluated their system using F-measures, and results showed that their combined reduction-based collaborative filtering approach outperforms the standard two-dimension collaborative filtering approach. Nevertheless they noted that this performance highly dependent on the context of the application that is tested. They concluded their paper by remarking on the importance of scalability of recommendation algorithms and needs for high volume of data.

AWESOME is developed as an adaptive web site recommender in [37]. It uses group of recommenders for automatically generating web site recommendations. For each recommendation, according to the rules in the system, one of the recommender is chosen. These rules are not static and the criteria for choosing the recommenders are updated dynamically according to the feedback of the users. So AWESOME has two phases where in the first one the recommender is chosen, and then the chosen recommender makes the recommendations. Architecture of the system considers the context while making the recommendations. This context is defined by set of parameters such as timestamp of HTTP request, user-related data etc. It also uses web mining for using implicit knowledge in the process.

The main separation for the recommenders is made according to these three criteria, that are whether they use information on current content (page, product information), current user (referrer information, search keywords etc.), usage history of users (purchase data, web usage etc.). It also considers additional context information such as current date or time. In their implementation, they used 8 different recommenders according to usage of this information. They tested their system in live environment according to the accepted (it checks whether user clicked on one of the recommendations), viewed (the user eventually access one of the recommended page in their visit) and purchased (user purchase an item in the visit) measures. It is seen that each recommender's performance quality changes according to the context. For example the recommender that only consider current user's search keywords performs very well for study pages whereas the recommender which considers personal interests gives the best recommendations for navigation pages. Selection rules are straight forward if then rules, but weights of these rules are updated dynamically according to the feedback of users. As a summary the paper gives a very detailed analysis of their system. AWESOME is one of the most promising systems in the field which uses several important features such as web mining, context

information, referrer information in addition to traditional recommender approaches of content-based and collaborative filtering.

In [38], hybrid recommender system Yoda's complexity has been improved. Yoda combines collaborative and content-based filtering in its recommendations. Recommendations are made considering different data sources such as human experts, web navigation patterns, clusters of user evaluation. In the paper they also update confidence of the recommendations automatically by utilizing implicit feedback. Their experiments showed that, system's complexity is low (O (n where n is the number of items in the recommendation lists) and remains constant even though the user/item sets grow. System's accuracy is twice more than the accuracy of nearest neighbor algorithms. They owe this accuracy improvement to usage of genetic-algorithm based learning mechanisms. Most important feature of Yoda is; it doesn't depend on explicit feedback of users; but it exploits client-side data for getting implicit feedback. Also they use human experts in recommendation process too for improving the reliability of these implicit feedbacks.

Authors of [49] proposed a hybrid recommender system which also combines collaborative filtering with content-based methods. Their main idea is to group users according to the content-based user profiles instead of grouping them according to the co-rated items. They try to solve the problem of sparse co-rated item set problem, because collaborative filtering techniques uses these co-rated item sets for making recommendations. However their approach considers dissimilar item similarity too while forming neighborhoods which is not considered as a good practice because similarity of negative feedbacks don't work well in recommendations. They applied their approach to WordNet which is a lexical database for English. They used tf-idf and Rocchio algorithms for measuring item similarity and learning mechanisms. Instead of measuring document similarities solely based on word frequency, they used concept semantics for overcoming synonym problems. So their system can determine high similarity for documents which use different terms to explain the same concepts. They compared the accuracy of their algorithm with collaborative filtering algorithm which uses Pearson's coefficient for computing user similarity, and results showed that their algorithm slightly outperforms the traditional approach.

[50] proposes another hybrid recommendation model for web users. In their model, multiple recommender systems work in parallel and in the end their results combined and presented to users. They applied web usage mining for handling the frequent changes happening in dynamic web sites and to improve the accuracy of recommendations. Their system consists of two modules which are "Web Page Prediction Model Based on Click-Stream Tree Representation of User Behavior (CST Model)" and "Model-Based Clustering and Visualization of Navigational Patterns on a Web site (Markov-Model)". One of the recommendation modules is chosen according to the weights and these weights are updated with respect to the feedback from users and prediction accuracy. They tested their system with using different recommendation algorithms too and results showed that the system should be consist of recommender modules which complement each other, otherwise the system's accuracy is lower than pure usage of constituent modules. Therefore in the end they used CST model and Markov Model in their final system which gives the best results.

A Hybrid music recommender system is presented in [51]. This system ranks songs using collaborative and content-based approaches. In their evaluation of songs they used ratings of users and acoustic features of audio signals. Their systems' goal is to give recommendations with high accuracy and high coverage of artists. They used a Three-way aspect Bayesian Network model for accurately representing user preferences. Also instead of having an offline phase, they

used incremental online training module which updates this Bayesian model. Their results showed that accuracy of recommendations in average are increased by 5% than pure collaborative or content-based methods. However in artist coverage, their system is outperformed by model-based content filtering algorithm.

[52] proposes a novel hybrid recommendation approach. They make use of conceptual relationships among web resources and make web page recommendations according to the usage behavior which is enriched by semantic knowledge. They enhance reinforcement learning by incorporating different data sources into process. This approach is novel in sense that, instead of using user ratings and explicit feedback, it uses web mining to discover user preferences. It also uses conceptual relationships between web resources to improve the deficiencies encountered while applying web usage mining such as recommending new web pages, coverage of pages to recommend etc. They try to give semantic meanings to the users' navigation behavior for overcoming these problems. For this they mapped pages to higher level concepts such as catalogue pages and they also map user visited pages to higher levels and they suggest new pages according to these concepts. They applied DCC (Document Conceptual Clustering) clustering algorithm to cluster web site content which is an incremental hierarchical clustering algorithm. Rewarding function which tries to shorten the browsing time for users so that they can reach to desired page as quick as possible, takes content similarity into account while making recommendations. So the main idea behind this intuition is to recommend pages to visitors which are semantically relevant to visited pages. In their evaluations they used, hit ratio, percentage of hits, predictive ability and many other metrics. Results showed that hit ratio of this approach is not as high as pure usage mining techniques but it outperforms other algorithms in terms of page coverage. This shows the trade-off between generalized (higher coverage, lower accuracy) and detailed knowledge (higher accuracy, lower coverage).

Different from most of the papers, [18] proposes a hybrid approach which combines item-based collaborative filtering with case-based reasoning for personalized recommendations. System is composed of two phase where in the first phase, item-based collaborative filtering which also considers context comes up with information items relevant according to the past ratings and personal information. Then in the second phase, case-based reasoning component makes relevant recommendations according to the constituent of these information items. One example could be in the first phase according to the users past history it can be seen that user search CD's according to the lyrics and mood of the songs. Then first phase can collect CDs according to these perspectives and in the second phase songs from these CDs' are recommended to users. Novelty of this approach is instead of considering single dimension rating systems, it considers multi-dimension ratings, and makes preferences based on multi-dimension ratings. So in the item-based collaborative filtering, instead of separating items according to liked/disliked groups, they are separated differently according to the usage of different perspectives. Therefore comparison of user, item similarity is made between the similarity vectors with n-dimension where n shows the rating dimensions. They used F1 metrics for testing performance of recommendations in usage of different number of perspectives. Results showed that as the number of perspectives increased the performance also increased. This is explained as if the similarity based on fewer perspectives the similar item space gets larger which contains items that are actually irrelevant to target users.

[36] also applies a novel hybrid recommendation model. Their model is composed of demographic, content-based and collaborative filtering. Ontology is used in user and user-interface modeling for determining objects, concepts and relations between them. Interestingly

they also apply ant colony metaphor in their recommendation model. It tries to choose the most optimal path in the user interface graph. In their architecture they defined user, item and interface models where the first two are essential for any recommender systems. For enabling communication between different web-sites they proposed to use ontology. User model is composed of demographic information such as name, address, usage data which determines user preferences, interests. Ontology is used for determining similar user groups. Interface model is represented as a graph, and interface models are described by several interface attributes such as type of layout, number of columns, topics of interests, etc. Ants in ant colonies that are almost blind, can find their way from nest to a food and back, by communication through a particular chemical substance. This is called any colony metaphor, and they applied this approach to their system. So interface recommendation is made by finding the path which has the highest usability by a user (ant in the colony). However the paper didn't show the results of the applied technique. It only comments that there were slight improvements (0.3 out of 10 point scale) in user satisfaction.

Fab is one of the oldest recommendation systems which combine content-based and collaborative filtering recommendations. It is explained in [14] and it has started working on December 1994. In this paper, they highlighted how Fab overcomes the known limitations of pure content-based and collaborative filtering approaches, in addition to how Fab enables group awareness and communication between users. The latter is done by automatic identification of users with similar taste. Users are grouped together according to the user-profiles that are based on content analysis. These groups are used in collaborative filtering to make recommendations. This paper didn't consider usage of web mining for retrieving implicit feedback, which could remedy their deficiencies in eliciting explicit feedback from users. However this paper was one of the earliest works in this area and leads many researches. Problems mentioned in the paper such as scalability, cold-start problems are still common problems encountered in current recommendation systems need to be addressed.

Famous applications such as Stumble Upon, Facebook, Yahoo also uses recommendation engines to attract users for using their systems, and providing interesting resources to their users. Stumble Upon makes recommendation of web pages, videos, pictures from web to its users. These recommendations are made upon the chosen categories users specified (content-based) and also according to the ratings users are giving to items collaborative filtering is used to recommend items from similar users. Stumble Upon uses friend networks to make more accurate recommendations. Same approach is followed by Facebook, as people receive news from their friends' activities. Recently it started to ask for ratings from users to come up with more related items. Yahoo is one of the earliest systems that applied personalization for its websites [56]. My Yahoo! is the personalized Yahoo pages for individual users. Users specify which contents they are interested in and information about these subjects are displayed to them automatically. Recommender systems are the essence of the popularity of these systems (175 million of active users in Facebook [53]) and this shows the importance of recommendation engines in the success of web-based applications as it can be seen that market leaders such as Facebook, Yahoo, Google owns their success by applying personalization in their web sites using recommender engines.

As it can be seen, in the last two decades lots of researches have been done and applications have been developed using recommender engines. The main goal of all these studies is to personalize web resources according to the needs and interests of users. Even though recommender engines are started to be used in many e-commerce applications, most of these systems have deficiencies because of the simplicity of recommendations strategies used in these

systems. In the next section we will describe how we will overcome these deficiencies, and explain in detail what we will do in this master thesis.

3 Framework

In this part we will give the outline for our master project. More particularly; we will give our goals in the project; give the sketch of our development process which includes how we will proceed in the project; what techniques we will try, what we will evaluate and so on. We will first start by giving our goals in the project.

3.1 Goals of the Project

In this section we will briefly explain what we want to achieve in this master project. Here are our goals:

- Develop and implement a recommendation engine for Bizztravel to help them achieve their business goals.
    - Improving the chance of capturing the interest of customers by showing the most interesting items corresponding to their user model.
    - Making shortcuts in the search path that leads to purchases
- Implement a recommendation engine purely based on anonymous users and implicit feedback
    - Overcome the unreliable nature of implicit feedback by successfully determining the users' interest on items. Make correct assumptions on determination of user interest by data analysis.
- Determine whether hybrid algorithm that combines content, collaborative filtering beside contextual information performs better than pure collaborative filtering or not.
- Evaluate the effect of contextual information (usage of day type) in the recommendation process.
- Develop a general architecture for recommendation engines that tries to cover all perspective of recommendation engines.

3.2 Methodology

In this section we will briefly explain how we will achieve our goals, we will introduce the techniques that we will use in the project and give the details of the working environment that we will work on.

- **Web Mining:** Web mining will be used to model the users and items through implicit feedback that will be gained through the web logs. Web logs are created and stored by Adversitement and they are kept in a session level. Then in our project we had done some data cleaning to remove sessions that skew the data analysis such as sessions end after one click. After data cleaning we did some data analysis to gather some information about the customers of the system and usage statistics of the web site. According to this information; we made our assumptions for determining the users' interest on items. In the latest phase we modeled the items and users according to the clicked items and search parameters used in the web site.
- **Recommendation Algorithms:** We first implemented pure recommendation algorithms that are collaborative filtering and our own algorithm based on search parameters. We integrated the contextual information to both of these algorithms. Then we added content filtering to the pure collaborative filtering to come up with a hybrid algorithm. In the

latest version of the engine; we combined knowledge based component to the system which contains promotions and campaigns of Bizztravel, and make recommendations by combining the results of this knowledge based component, hybrid algorithm and algorithm based on search parameter.

- **Evaluation Techniques:** In the evaluation we tried to see whether hybrid algorithm of content and collaborative filtering performs better than pure collaborative filtering according to the hit accuracy metric. Also we tried our algorithms with making different assumptions on the determination of user interest. Then we also checked whether our algorithm based on search parameter makes any shortcuts in the purchase scenarios such as reducing the search path. Last but not least in all our evaluations we evaluate the effect of using day type in the recommendation process and see whether contextual information really improves recommendation process or not.
- **Discussion on the Results:** In the last part we gave our comments, insights that we gained and conclusion about the results we achieved in our evaluations.

3.3 Working Environment

We developed this project using Java under Eclipse Framework and for database transactions we used MySQL database which is also employed by Bizztravel. For the connection to the MySQL from Java we used JDBC. Our main motivation behind choosing these environments was all of these tools and frameworks are free and platform independent, so we want to provide a chance for future researchers to work in this project without making any constraints on working environment.

4 Development

In this section we will explain development process of our application. Here we will give the details of the algorithms we applied. Also our motivation behind choosing these algorithms will be explained. We will start by explaining how we did the web mining for gathering the required data to make recommendations.

4.1 Web Mining

In this section we will describe how we achieved web mining in our system. In the following parts we will describe how we collected the data and explain the pre-processing we applied to these data and finally we will explain our data analysis techniques that we used to gather important information out of the web logs.

4.1.1 Data Collection

Adversitement uses a tool called Site Catalyst which is a web analytics tool. It keeps all information for each clicks happening in the web site and it identifies users in the session level. Therefore each session can be considered as a different user in our system. A session is defined as a sequence of requests made by a single user during a certain time period to the same web site [28]. For Site Catalyst session ends when the user doesn't make any request from the web server for 30 minutes, so in a session time interval between consecutive clicks cannot be more than 30 minutes.

Site Catalyst keeps information such as unique session id, IP address of the computer making the request, name of the page user is viewing, timestamp of it, values in get and post request such as search keywords, the id of the items clicked, etc. However it keeps all these information in one long query string. Therefore it needs to be processed to be usable in recommendation process. In the next section we will describe this process.

4.1.2 Data Restructuring

First we will show the entity relation diagram of our system in Figure 5 and then give the details of some of the important tables that we used frequently in our system.

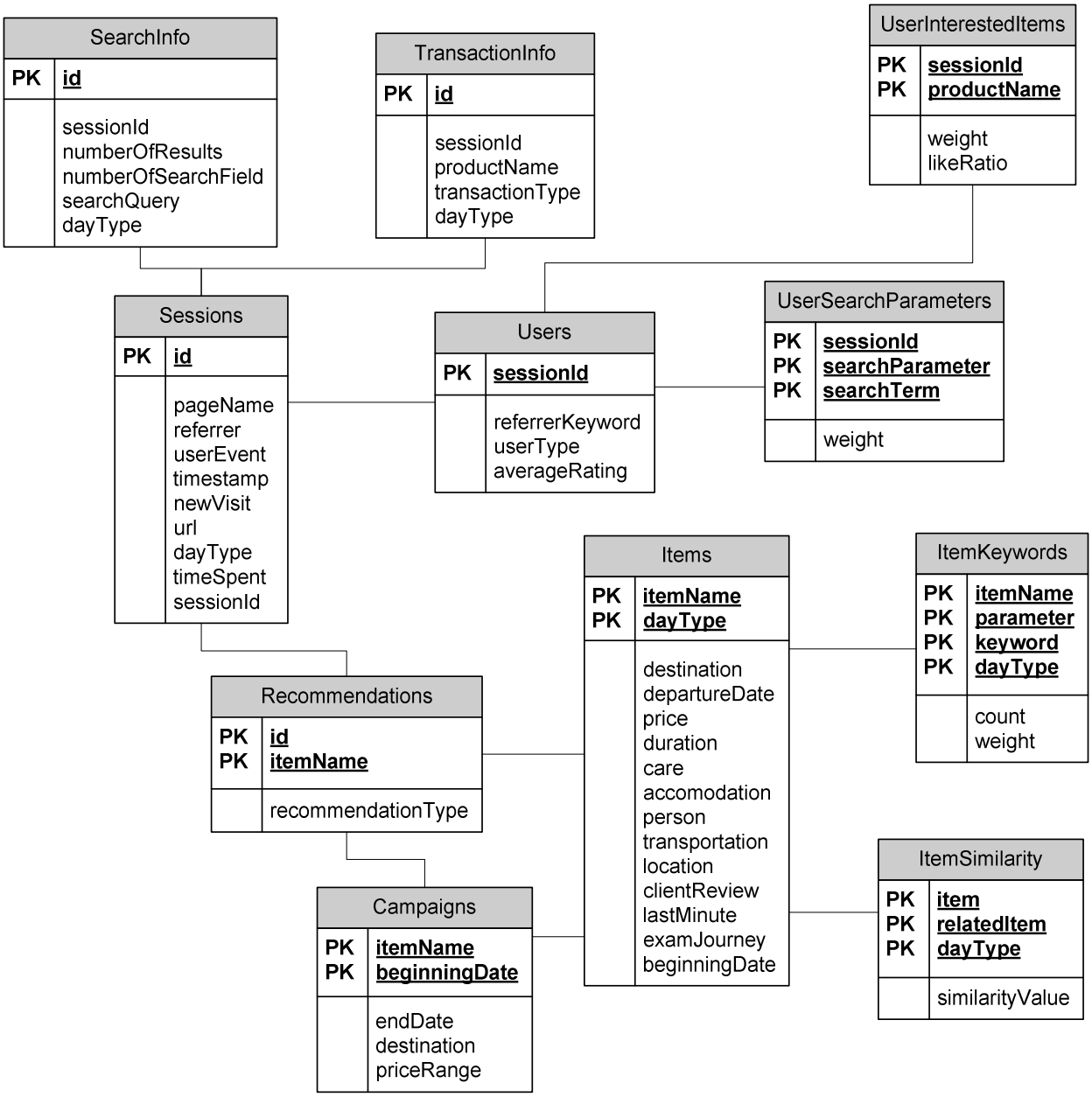

Figure 5: ER Diagram of our Database

Information about each click in Bizztravel is stored by Site Catalyst as a string, and we re-structured these data according to the following format and stored them in a database table called "Sessions".

- **Data Structure for each Row from Site Catalyst: Table for Sessions**
  - **id:** id which is the unique identifier of the row
  - **Session Id:** Session id that is assigned by Site Catalyst to identify users.
  - **Page Name:** The name of the page in the application that the user viewed in this request.
  - **Referrer:** It stores the name of the previous page that the users come from.

- **User Event:** It stores the type of the event user made in the application. User Events could be one of the followings:
  - **Enter Site:** The user entered the site for the first time; this identifies the beginning of a session.
  - **Search:** User made a search in the application, and for search event, information about the details of the search stored in another table in our database.
  - **View Product:** User clicked on an item and checks the details of it. Each View Product event is stored in transaction table to use them in recommendation process.
  - **Booking 1:** This is the first step of the booking a holiday which consists of five steps and this action means that user checked the prices of the holiday package. Again this event is stored in transaction table for later using it.
  - **Add Product:** This is the second step of the booking and here user added the item to their basket. Again this event is stored in transaction table for later using it.
  - **Booking 2:** This is the third step of booking. It is again stored in transaction table for later using it.
  - **Checkout Product:** This is the fourth step of the booking, and it is also stored in transaction table for later using it.
  - **Purchase Product:** This is the latest step of booking where the user made the actual purchase. It is also stored in transaction table for later using it.
- **Timestamp:** It stores the time and day of the request.
- **New Visit:** It determines whether the user visited the current page before or this is the first time user accessed this page.
- **URL:** URL of the requested page.
- **Query:** The query that is retrieved from site catalyst which keeps all the details of the current request. Even though storing it can be seen as redundancy, we store it in case that we might need some information in the future that isn't needed now.
- **Day Type:** It stores the type of the day that the request happened which could be either weekend or week. This is again a redundant field as it could be understood from timestamp but to fasten the computations we stored it as another field.
- **Time Spent:** It stores the number of seconds the user spent in the current page. We thought of using this value as measuring how much the user liked the item; but later we saw that values are not reliable because the users are browsing in tabbed fashion. In this case the time stamp of each new opened tag is almost same, but the latest tab the user looked will have a higher value for time spent which doesn't necessarily mean that the user spent the most time in this page.

As we described above; for each search of the user we keep details of the search in another table called "SearchInfo" in our database. This table is used for retrieving information about search which will be used later in our recommendation algorithms. Here is the structure of this table:

Table SearchInfo:

- **Id:** Id is the foreign key from the "Sessions" table which identifies request where the search happened.
- **Session Id:** It stores the session id of the session where the search has happened. It is stored for increasing performance as we want to avoid join operators between tables as much as possible.
- **Day Type:** The type of the day the search happened; this is also stored for improving the efficiency and making recommendations based on day type.
- **Number of Results:** It stores the number of results displayed to the user.
- **Number of Search Fields:** It stores how many search parameters the user used in their search.
- **Search Query:** It stores the search parameter and keywords pair that is used in the search.

Another table which is called "TransactionInfo" stores information about the events such as "View Product", "Booking" etc. This table is used for measuring how much the user liked the items in these transactions, and it will be used in the recommendation algorithms that we will describe later. Here are the fields of this table:

Table TransactionInfo:

- **Id:** Id is the foreign key from the "Sessions" table which identifies request where the search happened.
- **Session Id:** It stores the session id of the session where the search has happened. It is stored for increasing performance as we want to avoid join operators between tables as much as possible.
- **Day Type:** The type of the day the search happened; this is also stored for improving the efficiency and making recommendations based on day type.
- **Transaction Type:** This is the type of event the user did which is considered as a transaction in our system. This field can be one of the following values "View Product", "Booking", "Add Product", "Checkout Product", "Purchase Product".
- **Product Name:** It is the product id of the item that belongs to this transaction.

We keep information about users in the table called "Users" and each user is identified by session Id. Here is the table structure:

Table Users:

- **Session Id:** Unique Identifier of the users.
- **Referrer Keyword:** It describes how the user accessed the application. It could be either by bookmark where the user entered the website directly without coming from any other page or from Google by making search. In this case the search keywords that the user used are also stored and they will be used to make initial recommendations to users.
- **User Type:** Type of the user which will be defined by the search parameters the user used. For example if the user makes lots of search where the price keyword was constant but destination had been changed frequently, we tag this users' type as "Price" because this behavior shows that the most important criteria for this user is "Price".
- **Average Rating:** This is the average rating the user is giving for products. Even though users are not giving any explicit ratings for items; we assign ratings to the items by user according to the number of clicks the user made on items. In this field we store the

average number of clicks the user made on items and we use this value while measuring the adjusted cosine similarity.

For each user we also keep the search parameters the user used in their searches. Here is the structure of the table that we used for storing this information:

Table UserSearchParameters:

- **Session Id:** It defines the user that made the search.
- **Search Parameter:** It stores the name of the search parameter such as Price, Destination, etc.
- **Search Keywords:** It stores the value that the user used in this search parameter. For example Greece is one of the keywords can be used for Destination. Also in this application all the keywords are predefined and they have to be chosen from combo boxes.
- **Weight:** It keeps the number of this particular search parameter and keyword pair entered by the current user.

In another table we also keep the items that the user showed interest. We determine this interest by analyzing the number of clicks the user made. For each clicked item, if the user clicked on an item more than its average clicks it means the user liked the item, otherwise it means the user didn't like the item. So we tried to predict ratings for each item by considering number of clicks on an item as an indicator of interest. Here is the structure for this table.

Table UserInterestedItems:
Session Id: It defines the user that is related with the product.
Product Name: It is the id of the product the user related with.
Weight: It is the value of the item for the user according to our prediction, so it can be seen as the rating the user gave for the item. It is computed by reducing the average number of clicks of a user from the number of clicks the user made on the item. Negative value shows that the user is not interested in item and positive means the user interested in the items. We substitute average from the number of clicks for improving the efficiency as we will use that value in computing adjusted cosine similarity.
Like Ratio: It shows how much the user liked the item and it can be 0, 1 or 2. 0 means the user didn't like the item and it is determined by a negative value in the Weight field, 1 corresponds to a positive value in Weight and it means the user liked the item. 2 corresponds to the items that the user purchased which can be interpreted as the user loved the item.

Items that are clicked after particular searches and their connection to these search parameters are also stored in a table in our database. This table is used in one of the recommendation algorithms that we will use later on. Here is the structure of this table:

Table ItemKeywords:

- **Item Name:** It is the id of the item which is unique.
- **Parameter:** It stores the name of the search parameter such as Price, Destination, etc.
- **Keyword:** It stores the value that the user entered in to the search parameter.

- **Count:** It stores the number of times this item is clicked after a search with this parameter and keyword pair.

One of recommendation algorithms needs item similarities and for improving computational performances we store item similarities in a table. Here is the structure of this table:

Table ItemSimilarities:

- **Item:** It is the id of the given product
- **Related Item:** It is the id the given related product
- **Similarity Value:** It is the value that can range between -1 to 1, where -1 show that the given items are totally opposite and 1 means the given items can be considered almost same. 0 shows that items are independent from each other and other values shows weight of similarities. This value is computed by adjusted cosine similarity which will be described in the later sections.
- **Day Type:** It stores which kind of transactions have been used while computing this similarity value. So for each related items three different kinds of similarity values are calculated which are for week days, weekend days and for any kind of day.

We stored the recommendations for each request in the table named recommendations. And recommendation type of each recommendation is kept in this table too, which identifies how this recommendation is made such as recommendation is made from collaborative filtering or from knowledge-based component:

Table Recommendations:

- Id: Id is the foreign key from the "Sessions" table which identifies request that the recommendations are displayed for.
- ItemName: Represents the item that is recommended.
- RecommendationType: Identifies from which component, algorithm this recommendation made, such as knowledge-based component, recommendations based on search query, etc.

4.1.3 Data Cleaning

After structuring the data we remove some sessions that could create skews in our recommendations such as sessions that ended very quickly. In our case we saw that the average number of page views happening in a session is almost 4; therefore we removed sessions that ended less than 4 clicks from our database for improving the performance of our system and also improving the qualities of recommendations. Apart from this we didn't have to deal with regular data cleaning happens in web mining thanks to the Site Catalyst as it already removes unwanted sessions happens through crawling activities, bots, etc.

4.1.4 Data Analysis

After restructuring the data we had analyzed our data and come up with some interesting insights for Bizztravel; here we will display them. Before them here are the statistics about the data we used while coming with these results.

Features of Web Logs	**Training Data**
Beginning Date:	2009-05-10 00:27:02.0

End Date:	2009-05-18 01:44:05.0
Number of Rows:	91242
Number of Sessions:	7283
Number of Purchases:	95
Number of Searches:	19713
Number of Accommodations:	374

Table 1: Details of Web Logs

4.1.5 Interesting Facts and Insights Gained From Web Logs

- Traffic of data doesn't change very dramatically according to the day type. It means that number of visitors to the website, and number of purchases happening in web site is similar in week or weekend days. This balanced ratio might be a result of the fact that the customer segment of this application consist of students and young people who are not bound to working hours and might easily access the web site during any time and day in week. There are approximately 90 purchases per week and 70 of them happens in week days and 20 of them happens weekend.
- Traffic is high between 11:00 to 15:00 and 17:00 to 21:00 and most of the purchases happen during this period. We were surprised to see this as experience of Adversitement made us expect that the purchases after working hours would be particularly higher than rest of the day because we were expecting to see that users would make searches in the morning and would buy products in the evening. However our analyses show that users are likely to make purchases in afternoon as much as in evening. We believe this is again as a result of the users of the system being students rather than employees.
- We saw that between 00:00 and 06:00 the data traffic in the website almost stop and this provides us a chance to run our offline phase of our recommender during this period. We plan to train our engine by using the last week of data every evening so that our engine will be up to date all the time.
- In average each session ended in 4 clicks and almost one third of the sessions end with less than 4 clicks (2634 out of 7283). As discussed above these sessions are removed from our database not to skew our recommendations. Also this shows that many users just come to this application and leave the site after making one search and not finding what they are looking for. This shows that Bizztravel have room for improvement in sense of not losing the potential customers by presenting better results to them.
- In average the sessions ended with purchases end in 20 clicks. In most of the cases users make couple of searches and choose a product, then purchase that product. Therefore we can conclude that in many cases the users first come to site and search for some items then later on they come back and make purchases.
- It is seen that almost %30-40 percent of the users access the application through bookmark. This shows that there is a high amount of customers who are familiar with this application.
- %20 of users come from Google, and most of these users access the web site by using specific keywords such as 'Beach Master' which is the name of the application. Therefore usage of keywords in the initial recommendations is not suitable for in most of the cases.

- Rest of the users is mostly coming from web site portals which include similar web sites to Bizztravel which also sell holiday packages. This shows that users are not making search from only this application but they look for options from more than one web site. Therefore providing the most interesting items to users as soon as possible is very crucial for not losing a potential customer to a rival who is quicker than Bizztravel in providing interesting items.

From the above statistics and insights we concluded that, users are not focusing on particular time periods to make searches and other times to make purchases but in contrary at any time of the time and week they can make search or purchase. But there is still a difference in the navigation behavior of users according to the day type that is users spend more time for searching in weekends compared to week days which could be a result of the fact that people have always more free time in weekends whether they are employees or students. Therefore we used day type in our recommendation engine as an explicit factor to see whether using it really changes the performance of our system.

Also it is seen that users are making searches from many applications for finding the holiday package with best options. Therefore even though we recommend the best item to users; it could be that the user might buy the recommended product from another web site that has a better price than Bizztravel's offer. Therefore we concluded that instead of evaluating the performance of our recommendation engine on basis of number of sales, we will use number of users that are led to the first step of booking where they check the price of the holiday package. This will be a more fair technique for evaluating the performance because in the end it is obvious that if the user might find a cheaper price in another application; we cannot make them buy that holiday from Bizztravel. This goal is illustrated in Figure 1.

Last but not least; high amount of sessions end before making even 4 clicks show that many users give up on searching in the web site because they don't get good results in their initial searches and it means that there is a high potential for capturing the attention of this users by recommending more interesting holiday packages to these users instead of displaying holiday packages that corresponds to the exact search criteria.

In the following sections; we will explain our design choices and state our motivation behind our decisions. We will describe how we determine whether the user like an item or not and explain why did we choose such a method. Also we will explain the recommendation algorithms that we will use and which kind of contextual information that will be used in recommendations.

4.2 Determination of User Interest

As our system based on anonymous users; it is not possible for us to know exactly whether the user liked a holiday package or not. In most of the applications users' interest in items are calculated through the ratings; but this is not the case in our application.

Therefore we needed to determine a metric for measuring how much user interested in items. To achieve this we made some assumptions about user's actions. Here in this section we will briefly explain them:

4.2.1 Initial Assumptions

Initially we assumed that; clicking on an item is a sign of interest for users. So when the user clicked on an item; we assumed that the user liked the item. Also give more importance to items where the user bought a product or checked the prices of it; we used a weighting scheme in

measuring the users' interest too. So here is the initial weighting metric for measuring user interest:

- If the user viewed an item it shows user is interested in the item; and its weight is 1.
- If the user began the booking step which consists of 5 steps; it shows user is interested in the item, and if the user continued until the n^{th} step of the booking process; its weight is n + 1. For example the user purchased the item it means the user passed through all the booking steps for the item and user is interested in that item with weight of 6.

With these initial assumptions; we calculated item similarities according to the ratings we assigned for items by users. Then with these calculated item similarities; we evaluated our system by evaluation techniques of hit accuracies and average number of step reduction. Our results for hit accuracy was around %24 and average step reduction was 1 which means in average with our recommendations users reached to the products they bought by making one less search. Also we saw that our hit accuracy became much higher which was around %44 while making evaluations only with sessions that ended in a purchase.

While analyzing these results we come up with some important insights about recommendation engines. Here we will share them with our readers:

4.2.2 Lessons Learned from Initial Assumptions

Since the main goal of Bizztravel is to improve their sales; implicitly this goal was also one of the goals of our project. The results of our evaluation shows that we are on the right track by using a weighted schema for computing the interest level of users for items. In our method; items that are purchased more has higher weight than other items that are just viewed and this led to better results for predicting what the users will buy.

Our specialty in this project is making recommendations without knowing too much about the users we are dealing with. For doing so we assumed that viewing an item shows a sign of interest. However in our scenario; viewing an item doesn't necessarily have to mean that the user is interested in the item; but they just viewed it to get some information about it; and then it could be that the user didn't like the holiday and keep searching for other holidays or even though they liked the item; they keep searching to see their options.

This leaves us with the fact that; there is only one event in the system which is an exact sign that the user is interested in the holiday and it is purchase event. When a user made a purchase; it can be counted as the user liked the item and our system can recommend them other similar holiday packages which are similar to this purchased item and everything would be fine.

However things get interesting here; because the following fact spoils everything about this good observation and make it useless in a way. Problem with Bizztravel is that; when a user made a purchase; their business with the website ends there and they quit the website. Because holiday packages are not items that people buy frequently; most of the time they are sold once a season per user. However recommendation engines are more suitable for businesses where lots of transactions happen by same customer; such as in Amazon where the user could buy several books in even one week. And most of the recommendation and evaluation techniques are designed for those kinds of applications. So these insights bring us to following conclusions:

- Recommendation engines need applications where; it is clear whether the user liked the given item or not. If this is not clear then assumptions about determining user interest

should be done very precisely by making data analysis and communicating with domain experts.

- The users of the system should at least show that they are interested in more than one item; so that there can be a solid foundation that the recommendations can be based on. If this is not the case; then at least they should make transactions with website frequently so that; relations between items themselves or between items and user behavior could be extracted out. However this is not the case for Bizztravel too; because users don't make frequent transactions but make only one transaction and stop using web site.

4.2.3 Differences of our Recommendation Engine than Traditional Ones

These facts made us understand that; our recommendation engine should be working with a different mindset than traditional recommender engines. This is because of the fact that in our system; our users don't get affiliated with more than one item i.e. in our case users don't buy more than one item. This is an important characteristic because of this feature; our recommender engine should be recommending items that matches the current user's profile as much as possible. However for traditional recommender engines this is quite different because one of the main goals of traditional recommender engines is to show people what might be interesting for them; rather than showing the best item.

So in a sense traditional engine should achieve some kind of discovery process rather than coming up with an item that is closest to the user's profile. This is because; in traditional application such as Amazon; Last.fm users enter the website several times and they are looking for more than one item; and it is more or less known that what they are interested in and what they like; but the main purpose of recommendation engine in those applications is not to recommend similar items to users but instead come up with new different items which might not be matched exactly with the user profile but might be interesting for user. Therefore traditional engines focus more on the novelty feature than similarity. However in our case we cannot focus on the novelty feature as our system is a one-time use system; therefore our main goal should be recommending the item that matches the current user profile as high as possible instead of trying to help users to discover their hidden interests.

These insights led us to re-structure our assumptions and see how they will affect the evaluations. They also made us think about new evaluation approaches that are more suitable for using in an anonymous user environment. Here we will give our refined assumptions.

4.2.4 Refined Assumptions about Defining User Interest

In our previous assumptions; we used viewing a product as a sign of user's interest in item. However we decided that it is not a correct conclusion as viewing the details of a product isn't supposed to mean that user interested in the item; but it only means that user is looking for some holiday.

Our problem here is that we don't have any explicit feedback from users and to train our recommender engine we have to determine what an indicator of interest for the users is. As the only feedback we get from users is their click; we come up with a modified version of our previous assumption. This time instead of considering viewing the details of a product as a sign of interest; we used the number of clicks on an item as a sign of interest. On top of it; we also assumed that if the user only views the details of a holiday but don't continue with checking the price of the holiday and instead of it view the details of another hotel; it means that the user

didn't like the item he clicked on. Again purchase of an item is the strongest sign of indicator. So we can list the rules as follows:

- If the user only viewed the details of an item it means the user didn't like the item.
- If the user checked the prices or added the item to its basket after viewing the details it means user interested in the item.
- If the user purchased an item that item is the favorite item of the user.

Figure 6 is an illustration of the assumptions we made in the determination of the user interest.

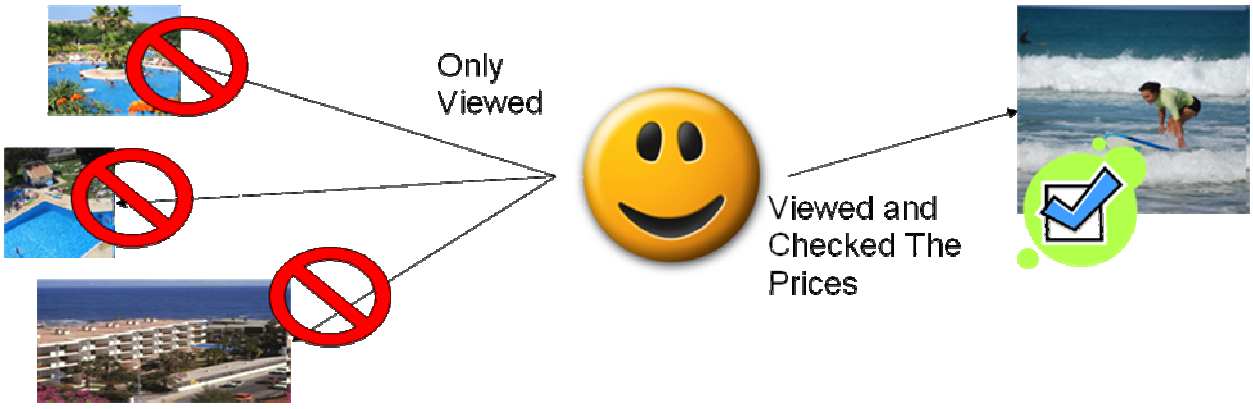

Figure 6: Determination of Item Similarities

In addition to these assumptions; we also improved our similarity measurement function. We were using pure cosine similarity function which doesn't consider the differences in users' ratings in our evaluations with previous assumptions. For example for a user who checks the prices of all holidays; checking a price of another holiday is not a sign of interest. On the other hand for a user who only views details of a holiday but only checks the prices of a hotel once in a while; checking the hotel prices is a sign of interest.

For overcoming the differences in rating scales; we decided to use adjusted cosine similarity. In this method; the ratings of users for each item is decreased by that user's rating average. It is obvious that the pure usage of above schema is not suitable for using this technique; because this technique is better suited for rating schemas with higher rating interval. So we updated our schema above; and included number of clicks and average number of click of users into the picture. Here is the latest version of our method:

- For each user the average number the user click on items is computed.
- For each item the user clicked on; if the user only viewed the details of it but didn't do any other action with the item such as booking it, adding it to a basket; it is a sign of dislike for the item.
- For each item the user viewed details and did other actions such as checking price, etc. if the user clicked on the item more than the average number of clicks per item, it is a sign of interest otherwise it shows that user didn't like the item very much.
- Purchase of an item still makes that item the favorite item of the user.

With this methodology; we computed and stored the item similarities in our database by using adjusted cosine similarity function. Then we made the same evaluations with our new assumptions and the hit accuracies increased to %29 from %25 and average step reduction increased to 1.82 from 1. Improvement in the step reduction means that; with our recommender engine; users of the application will find the holiday packages they are looking for in a shorter amount of time. And implicitly this means that part of the people who quit Bizztravel after not finding the items in their initial searches will able to find interesting items with our engine; and this will increase the potential of selling holiday packages in the application; which is the main business goal. So in other words; with our engine Bizztravel will have more chance of capturing the attention of customers and will not lose potential buyers.

It is obvious that our new assumptions led us to better results; but we still believe that we need to find a better evaluation technique than hit accuracy or at least update it a little bit so that it is more suitable to be used for anonymous user based applications. In the later sections of this thesis; we will describe modifications to hit accuracy and the details of our evaluations will be given too.

We used number of clicks for measuring the sign of interest of users; instead of that we could also use time spent on a page as a sign of interest. However we chose number of clicks rather than time spent as an indicator and we will explain our motivation for it in the next section.

4.2.5 Number of Clicks vs. Time Spent

Because of the anonymous nature of our application; we have to use implicit feedback for measuring the users' interest on items. There are two obvious indicators that we could use and they are number of clicks on an item and time spent on a page. Time spent on a page could be used for understanding whether the user interested in the item; so to say if the user spent more time than average on a particular page; it can be counted as user interested in an item.

However this approach has its drawbacks because of the unreliability nature of the implicit feedback. First of all if it is the case that users totally concentrate on the web site for searching a holiday and spent actually all the calculated time spent on page for examining the page; then this approach would bring us very important results. Unfortunately this is not the case; and there is no way for us to understand whether the user really viewed the page during the whole time the page is open or whether user went to a kitchen for getting a coffee and viewed the page for only very short amount of time.

If this was the only exception; this approach might be still useful; but there is even a bigger problem than not knowing what the user really did while the page was open. It is a problem created thanks to the famous browser Firefox who invented tab browsing. Because of it; in the same sessions users might open lots of page from the same web site and view those pages sequentially.

Even though tab-browsing brings a greater usability for internet users; it makes the job of web analytics almost impossible to understand when exactly the users start viewing a page. It is because the request time of all the pages opened in different tabs have almost the same time; even though the user don't start viewing the pages opened in other tabs immediately. This leads to skews in the calculation of time spent on viewing a page, and it makes it useless for us to consider it as an indicator of interest in recommendation process.

Therefore we chose the number of clicks for sign of interest rather than time spent. Because number of clicks doesn't get skewed as much as time spent get. Here we only make the assumption that each click on an item has same importance which might not be the case because

of the usage of back button. However as we don't have any chance to understand exactly how much the importance of a click we had to consider them equally with respect to understanding the users' interest in items.

4.3 Choice of Recommendation Algorithms and Components of Recommendation Process

In this part we will describe why we chose the algorithms for our application. As we described above we decided to use item-to-item collaborative filtering algorithm and an algorithm based on search terms.

4.3.1 Item-to-Item Collaborative Filtering

We used item-to-item collaborative filtering algorithm that is used in Amazon. First of all we chose item-to-item collaborative filtering because of the fact that they perform better than user-to-user algorithm and they are scalable to be run with a huge item space [60]. Also in our case working with a user-to-user algorithm would lead to computational problems because we treat each session as new user and this would lead to a huge user space compared to our small item-space.

We used Amazon's technique rather than traditional algorithm for improving the computational efficiency. Because traditional algorithm tries to compute item similarities between each item even though between some items there aren't any common customers who showed interest to both items. In Amazon's approach items that are interested by common customers only taken into account while computing the similarities between them.

Apart from these facts success of Amazon also pull our attention to work with this algorithm and see how it will work in an anonymous user application which is not the case in Amazon.

Also in our application the main feature set that we model users based on the items they click; therefore it is logical for us to compute similarities between items so that we can recommend items to users according to the items they showed some kind of interest. Therefore by using a collaborative filtering algorithm; we believed that we can overcome the deficiencies of not able to define users with personal information such as age, holiday interests etc.

4.3.2 Recommendations Based on Search Terms

The main purpose of this algorithm is to make relations between items and search terms. Search terms are one of the most important feedbacks that we were getting from web logs and also one of our goals is to reduce the amount of time users are spending on search. So we believe that if we can make some relations between search terms and items then according to the entered search terms we might bring the most interesting items to users.

Therefore we chose such an algorithm so that we can exploit resources in our hands which were search terms. Also one other important feature of search terms is that they implicitly give us content and contextual information too. Search Parameters such as "Destination", "Hotel Type" gives us information for coming up with recommendations using content information. Search parameters such as "Price", "Number of Travelers" give us contextual information and by making connections between these parameters and items; we are able to put contextual information into recommendation process which is a deficiency in many of the traditional recommendation engines.

4.3.3 Choice of Contextual Information

After analyzing the data of Bizztravel we concluded that using week day and weekend day data separately in recommendations might bring us better results. By doing so we wanted to see whether people have different kind of decision mechanism according to the week or weekend days. Our motivation was; weekend days are holidays and people would be more motivated in making purchases and would have more free time than week days to make better searches.

We are suggested to use time of day too separately and check how the users' navigation behavior changes with respect to the working hours (9:00-18:00). However after analyzing the data we saw that; traffic load of the web site wasn't difference in working hours and after work hours. Therefore we decided to give up on making a separation because of time constraints of our project; even though it would be interesting to see how this separation would change the performance of recommendations.

Also as our application based on a search mechanism where users might search for items according to the features such as "Price", "Number of Travelers" we implicitly used this kind of contextual information in our recommendations too; but again because of time limitations couldn't evaluate the effects of them.

4.3.4 Item Representation

We first represented all our items based on a customer vector. So each item is represented by the users who showed interest in these items. This was also helpful for us to compute item similarities by using this vector in adjusted cosine similarity metric.

Then later on we decided that content information needs to be part of the item representation. Therefore we concluded to represent our items in two feature set where one is based on customers and second is based on content information that we retrieved from Bizztravel. This information is about the destination of the accommodation, type of the hotel, location of hotel with respect to beach, etc.

4.3.5 User Modeling

Similar to item representation we used more than one feature set for modeling users. Here one feature set was the holiday packages that user interested in. Second feature set was the search parameters and keywords that the user used in their searches.

According to the type of users one of those features set would gain more importance than the other. For example for an user who makes lots of search but click on items from time to time, weight of interested items will be higher than search parameters, as a rare click on an item shows that that item is really important for that user. Then according to the weight of these feature sets more suitable recommendation algorithms will be used; which is item-to-item collaborative filtering in this example.

Apart from these feature sets; we will also use from where the user accessed our application, and the home page of our application will contain recommendations according to the referrer type. For example the user entered our site from Google by making a search about Greece; then we will make recommendations that are related to the keyword Greece by using our algorithm based on search parameters.

Also the order of the search parameters used in the search of users will affect the user modeling too. Each search parameter will have a weight according to the usage ratio. For example if user is making lots of searches where the "Destination" parameter changed 3 times but "Price" parameter stays constant; then the importance of "Price" parameter will have a

weight of 1/2 where each keyword for "Destination" parameter will have a weight of (1/2)/3 which is 1/6.

4.4 Recommendation Algorithms

After explaining our design choices; we want to give the details of the recommendation algorithms that we will use in our project. We first start with explaining usage of our algorithms in a pure fashion; then in the end we will describe our hybrid technique where we combine these algorithms.

4.4.1 Item-To-Item Collaborative Filtering: Amazon's Algorithm

In our recommendation algorithm; we decided to use a similar algorithm to the one used in Amazon's recommendation process. It is an item-to-item collaborative filtering algorithm and it is composed of two phases; where the first phase executed off-line and the second-phase is executed on-line and makes the real recommendations. We gave the reasoning behind choosing this algorithm in section 4.3.1; readers could check that section again to refresh their mind.

In our algorithm instead of matching similar customers; we match customer's each interested items to similar items [60]. Then sort these similar items according to the weight given by our algorithm and returns the top-n result as the recommendations. We defined user interest according to the criteria we defined in section 4.2.

In our algorithm; we build a similar items table and keep that table in the database as it is very huge to keep it in the memory. Similar item tables are computed by determining the items that attracts interests of the same customer. For increasing the computational efficiency; instead of iterating through all item pairs; we only choose items which has a common customer. This is because of the fact that; many product pairs don't have common customers at all [60]. Also we use the weights that we computed for defining the interest of user's into items. Here is the algorithm of our off-line phase:

```
For each item h1,
    For each customer C who are interested in h1,
        For each item h2 that is interesting for C
            Record that C is interested in i1 and i2
    For each item h2
        Compute the adjusted cosine similarity of h1 and h2 by using the average ratings of
        customers
```

For computing the similarity of related items; we used the well-established method of adjusted cosine similarity. Adjusted cosine similarity is an updated version of similarity method named cosine similarity. Cosine similarity method computes the cosine angle between the given two vectors. And here the given two vectors will be the related items. Each item is represented by m dimensions which hold the customers who are interested in this item. In adjusted cosine similarity average ratings of customers are also considered in computations for normalizing the differences in rating methods of users. Here is the formula for computing the similarity between items i and j [61] and Ru,i is the rating user u gives for item i and Ru is the average rating of the user u gives for items.

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

Equation 1: Formula for Computing Adjusted Cosine Similarity

The computation of the above algorithm will be done off-line. After analyzing the web logs of Bizztravel we have seen that; users are not using the web site during the night; as Bizztravel only works in Netherlands and Belgium; this gives us the chance to run our algorithm every-night; and every morning we can make our recommendations with updated data.

The worst case offline computation of the above algorithm is $O(N^2M)$ where N is the number of customers and M is items; but in practice it is much closer to $O(NM)$, because most customers have interests in very few products [60].

After making this computation; in the online phase; our algorithm checks the current user's interested items and comes up with recommendations that are most similar to those items. This computation is very quick as it is only a look-up operation from the database and only depends on the number of items the customer is interested in.

This is only one part of our main recommender. This algorithm will be combined with our other algorithms; but in some scenarios it will be used purely. For example when the user directly clicks on an item after entering the website; we only know this information and by using this algorithm the user will be recommended similar items. But if a user clicks on a product after making some search; the user's search terms beside the clicked item will be used for the recommendation.

How we use these search terms will be described in the following section. Also as the main objective of Bizztravel in using this recommendation engine will be achieving their market goals; we will give them chance to promote particular products than others by letting them enter their promotion products to our knowledge-based component. These products entered by Bizztravel will have more weights and will be ranked in higher positions according to the current user's preferences.

4.4.2 Collaborative Filtering based on Search Terms

Bizztravel uses an application for selling holiday packages that is based on a search mechanism which uses search facets. These search facets ranges from destination to price, location of the hotel to inclusive type, etc. Therefore these search mechanisms tells a lot about the user's interests and they will be part of the user modeling; beside the interested items of users.

Data analysis on the web logs of Bizztravel shows that; most of the users directly start searching for a holiday package before clicking on any of the holiday products. Therefore we thought of making recommendations based on the used search facets. This also solves the deficiencies of the item-to-item collaborative filtering described above as that algorithm always need users to show some interest in some items; however most of the users don't click on any item before making couple of searches.

Therefore while users making searches; we also try to recommend them holiday packages according to the keywords and search facets they used. In this approach we again used a collaborative filtering technique. This time we relate each item to search keywords. But as a result of the characteristics of search facets in these applications we also use contextual information such as "Price" and "Number of Travelers" implicitly.

For better illustration we will give an example how our algorithm works. After the user search for a holiday in Greece which doesn't cost more than 300 Euros; the user clicked on the holiday package h1 and purchased that item. It means that the users who are looking for a holiday in Greece or looking for a hotel which doesn't cost more than 300 Euros; could be interested in h1. Therefore we relate h1 to both of these keywords. We again used a weighting schema; so each parameter have different weight for their related items; i.e. if there are 10 users who clicked on h1 after making searches about Greece; keyword Greece for h1 will have the weight of 10.

This computation will be done on-the fly; so when a user makes a search and clicks on an item; it will immediately be stored in the database and this will increase the quality of the recommendations.

So when a new user enters Bizztravel and search for a holiday; according to the keywords they used; they will receive recommendations by getting the items which got most attention from other users who used same keywords. Here is the algorithm for this technique:

```
For each keyword k entered by the current user:
        Add the top n items to recommended list r that are related to this keyword
                [ If the added item is a new item, set its count to 1
                Else increment its count ]
Return the top n item from the list r according to the count value of the items in the list.
```

This algorithm will be used purely by itself; when there isn't any information about users. For example if the users didn't click on any item we can't apply the above item-to-item algorithm; but when the users make searches we can use this algorithm and provide recommendations to the users. After the user starts showing some interests on some items; then this algorithm will be combined with item-to-item algorithm and users will get recommendations from both algorithms.

We will combine these two approaches like this; we will come up with the similar items to the interested items of the users. Then these items will be ranked according to the relation to the search keywords users used and the top items will be recommended in the end. In the next section we will describe how we combine our algorithms in one hybrid model.

4.4.3 Hybrid Model Based on User Models and Recommendation Algorithms

For our recommendation engine; we will model the users on the fly according to their navigational behavior. We explained how we will do the user modeling in section 4.3.5, and as it is said over there the main component of our user models will be interested items and search keywords. Apart from them referrer URL that shows the link where the user came from will be part of user modeling too.

So in our hybrid model; as long as user don't show any interest on any item or make any search; we will try to make initial recommendations according to the referrer URL. Of course if the referrer URL doesn't give any hint at us about user such as user accessed our application from a link from web portal; then we will wait for users to show some interest in items or make some search until we start making some recommendations.

Then if the user only makes search but not click on any item; we will make recommendations based on search keywords and otherwise if the user only clicks on items but don't make any searches; then we will recommend items only according to the item-to-item filtering.

However when user starts making some search and at the same time start clicking on items; then we will start to combine the results of our two algorithms to improve our recommendations. As soon as user started to show some interest in some items after making some searches; search keywords will be used as content or context information will be used in combination with item-to-item filtering. In essence we will have two different similarity values for each item; one of them will be computed according to the customer vector and the other will be computed according to the content information. So in this example average of item similarities computed from customers and content will be taken; and top similar items will be recommended.

In a similar fashion; after user show some interest in some items and make a search; then the recommended items that based on search keywords will be compared to interested items and most similar items to the known interested items will be displayed to the user. Here is the algorithm that we used in our hybrid algorithm:

> Make one recommendation from the knowledge-based component;
> Make one recommendation according to the latest action of users:
>
> If it is a click on an item make recommendation according to the item to item
> Else if it is a search; make recommendation based on the search terms
> Else if the user entered the site from another website by making search; make recommendations based on this refferer keyword.
>
> Make the remaining recommendations from either item-to-item or search term alogirthms proportional to the searches and clicked items of the user. (i.e. if the user only clicked on items but didn't make any search, make all the recommendations from the item-to-item algorithm)

Hybrid Recommendations in the Application:

In our application, we will show 5 recommendations to each user in each page view. This is a requirement from Bizztravel and another requirement is that, they want to recommend their promotions more than ordinary items, so essentially in the recommendation process they want particular items to be pushed forward in the recommendation list.

For achieving this we created a GUI tool for Bizztravel to put those promotions into our system, and from that tool they can enter destination values, price ranges so that when the users make searches, if the keywords matches with the keywords of promotions, these promotions will be recommended even though real algorithm wouldn't recommend them. Bizztravel wants such a feature, because from some travel packages are only sellable in particular interval of dates, therefore they have to sell these packages before the end of these intervals.

Also to make our recommendation engine dynamic, we wanted to give more importance to the last action of the user. Therefore in each page view, one of the recommended items will only be according to the last action of the user. This enabled us to come up with changing and dynamic recommendations.

So in the end for each page view, one item will be recommended from the promotions of the company if there are any promotions, and one item will be recommended according to the last action of the user and the rest of the items will be recommended according to the user model. Figure 3 illustrates the working mechanism of our hybrid algorithm.

4.5 Mapping of the Implementation to the General Architecture

In this section we will try to show which components of the architecture we gave for recommendation engine that we used in this master project. We will do the mapping between our application and proposed architecture. Here is the architecture:

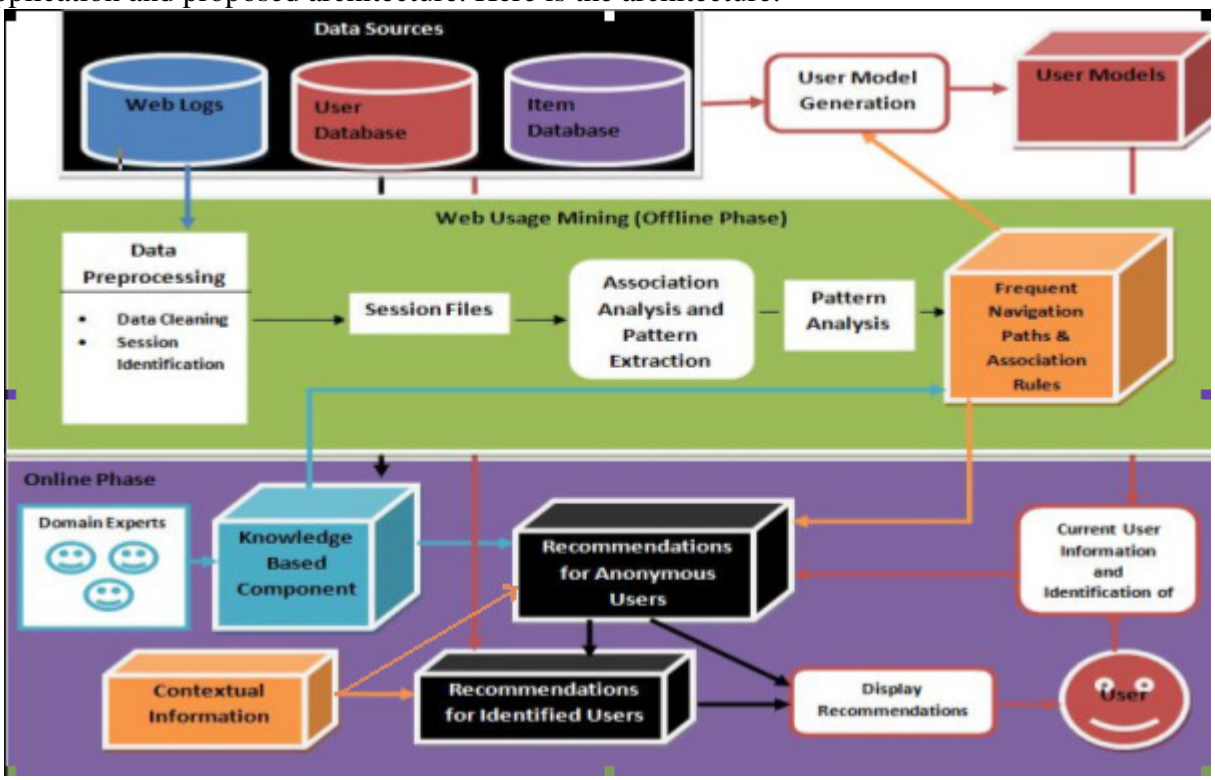

Figure 7: The General Architecture for Recommender Engines

4.5.1 Mapping of Current Version to the Modules in the Architecture

1. **Data Sources:**
   a. **Web Logs:** Web logs of Bizztravel.
   b. **User Database:** Users Table in our Database
   c. **Item Database:** Waiting for Bizztravel to send the details of accommodations.
2. **User Model Generation:** User models are generated on the fly as users navigate in our application by considering users' search parameters, interested items, and link the users used to access our web site.
3. **User Models:** Users are separated according to the search parameters they use most. Also from where the user came to our site is also stored in our user table to be used in the user models.
4. **Offline Phase:** Update of the training data will be done in offline phase. Every night; the previous day's logs will be added to our training data set (we plan to keep the last week of logs for training our recommender) and item similarities will be computed.
   a. **Data Pre-processing**
      i. **Data Cleaning:** Adversitement is responsible for main part of the data cleaning. Apart from them in our application we remove the sessions end with less than 5 clicks.

ii. **Session Identification:** Adversitement is responsible for it and our application gets the unique session ids which helps us to identify sessions.

b. **Session Files:** In our application; we re-structure the Bizztravel log files into sessions. We keep session, search (searching through facets), transaction (view product, booking, etc.), users (identified by session id), item similarity (stores the item similarities for fastening the computations) information in different tables; so that recommendations can be done easily.

c. **Association Analysis, Pattern Extraction, Frequent Navigation Paths and Association Rules:** These will not be implemented in this project; because the application we worked on is not suitable for it to do. Nevertheless making recommendations based on search terms uses techniques similar to association analysis where associate search keywords with items, and order of the search keywords will be used in the future which can be considered as a practice of frequent navigation paths.

5. **Online Phase:** Actual recommendations to the users on the fly while they navigate in the application will be done by the modules in this component.

   a. **Domain Experts:** Guido Budziak who suggested to change our goal metric from increasing sales to leading users to the first step of purchase which is checking prices. Employees of Bizztravel also helped us by stating their requirements in our meeting where we convinced them into work with us for developing this project.

   b. **Knowledge-Based Component:** Bizztravel focuses on this component as they want full control on the recommendation process and wants to increase the weight of campaign products in the recommendations. This component will help them achieve their marketing objectives; it is a GUI based tool where the users will be able to easily update the importance of particular products; add campaign details, etc.

   c. **Recommendation for Anonymous Users:** For this application all the recommendations are based on anonymous users. Users are bound to the time of their sessions; and for every user who starts a new session is treated as a new user and except for the history stored in the current session, we don't use any transaction history of users in our recommendations. In this module we will use hybrid model which consists of our recommendation algorithms and make recommendations according to the current user model.

   d. **Recommendation for Identified Users:** This module will not be implemented as this application only works with anonymous users.

   e. **Contextual Information:** In our algorithm where we make recommendations based on search terms; we implicitly use contextual information in our recommendations such as price, number of person, etc. We also evaluate how the recommendations are affected by separating recommendations into week and weekend days.

5 Evaluation Techniques

In this section we will describe the evaluation techniques that we used to evaluate the performance of our system and give the results of our evaluations and interpretations of our results will be given in the end of the section.

In the evaluation of our recommender engines; we come up with two different techniques. We used a technique similar to hit accuracy and another technique to compute average number of step reduction in holiday purchases.

Our main goal in the evaluations is to determine whether hybrid algorithm has a better performance than pure collaborative filtering algorithm. Also we wanted to see the effect of using day type directly in the recommendations and to check whether using day type really improves the performance or not.

Also because of the anonymous user and implicit feedback nature of our application; we had encountered with difficulties to evaluate our system. Because in our application we didn't have any historical data for our users which leads to even more sparse data set which reduces the quality of our evaluation. The best approach would be running the application integrated with recommender engine in live environment and see how the users are navigating; but because of time limitations we couldn't try it in this thesis.

In applications where the users are identified and gives explicit feedback such as ratings; according to the interested items of the users the evaluations might be done rather easily. For example if the user gives ratings to 4 items; by using 3 of these 4 items to train the recommendation engine; according to the accuracy of predicting the rating of the last item the evaluation of the recommender might be done.

However in our case as the determination of user interest was done through implicit feedback which is not reliable enough this evaluation technique resulted in low results and as the user models were only generated based on the clicks happened in one session; there weren't enough data to understand the interest of users.

To overcome this problem besides predicting the items that user might be interested in; we tried to evaluate whether we correctly determined if the items that the user liked are indeed similar to each other. So for example according to our determination of user interest technique; if the user liked 3 items; we check whether these 3 items are indeed similar to each other according to the adjusted cosine similarity method (similarity value bigger than 0 means that the items are similar where as negative values show dissimilarity). We will describe the details of each technique that we used in the evaluations in the following sections.

5.1 Hit Accuracy Metrics

There are traditional evaluation techniques for recommendation engines such as hit accuracy and rating prediction. However these techniques are based on explicit feedback. For example to compute the hit accuracies for rating prediction; the data set would be separated into training and test data. Then the recommender engine would be trained by using the training set, and it would try to predict ratings for the test data. Then the predicted ratings and real ratings would be compared to measure the accuracy of the recommendation engine.

However in our case; as we don't have any explicit feedback from users and we cannot keep the history of the transactions of users because of the anonymous nature of Bizztravel we have to use different techniques as we stated in the beginning of this section. But again as the traditional evaluation techniques; we separated our data set into training and test data. We trained our

recommender with the training set and perform the evaluation techniques on the test data. Below we will give the details of the data sets we used.

Features of Data Set	**Training Data**	**Test Data**
Beginning Date:	2009-05-10 02:00:00.0	2009-05-24 02:00:00.0
End Date:	2009-05-24 02:00:00.0	2009-05-31 02:00:00.0
Number of Rows:	108010	58635
Number of Sessions:	10135	5793
Number of Purchases:	183	73
Number of Searches:	38904	21892
Number of Accommodations:	293	291

Table 2: Details of Training and Test Data Sets

One of our goals in this project is to see the affect of contextual information for recommendations. Therefore we make evaluation under different context i.e. we separated the web logs into weekend and week days. We computed three different similarity values for items by making computations based on only week day, or weekend day or any day. Our goal is to see whether making recommendations based on week day or weekend day makes any improvement or not in the general performance compared to the performance achieved by using all data in recommendations. Here are the details for the training and test data with respect to weekend and week days. Figure 8 displays the distributions of number of sessions per number of searches.

	Training Data		**Test Data**	
Features of Data Set	Week	Weekend	Week	Weekend
Number of Rows:	82075	25935	44687	13948
Number of Sessions:	7734	2432	4504	1305
Number of Purchases:	142	41	62	11
Number of Searches:	29404	9500	16487	5405
Number of Accommodations:	291	283	287	266

Table 3: Details of Training and Test Data Set Separated into Week and Weekend Days

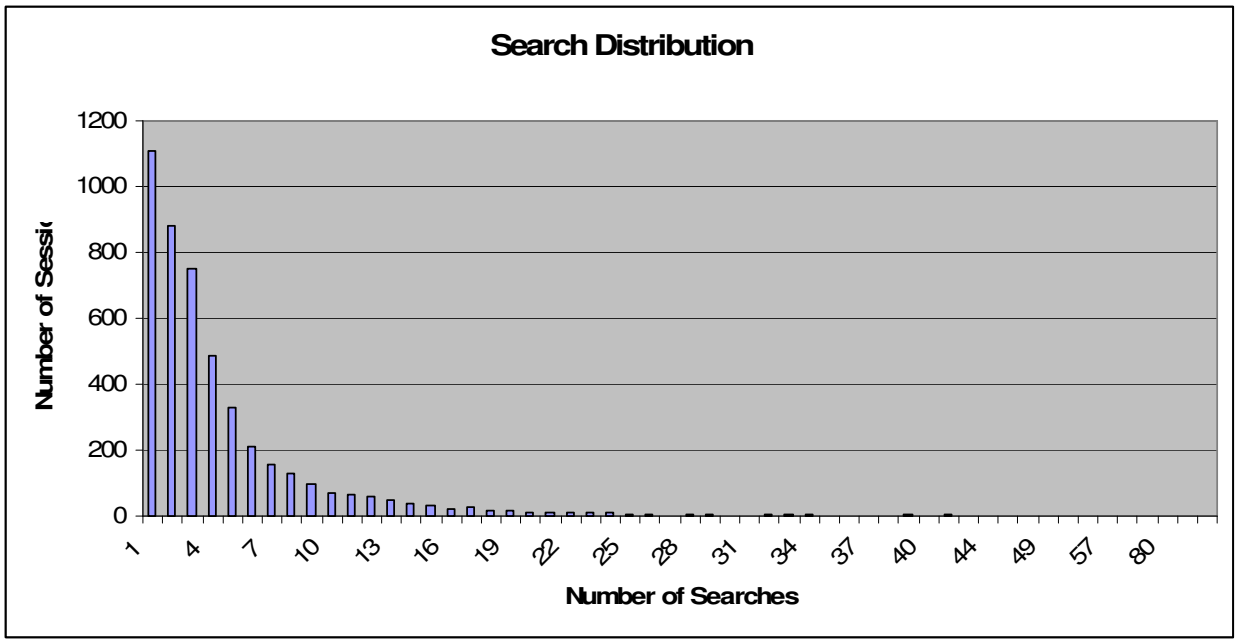

Figure 8: Search Distribution along the Sessions

5.1.1 Definition of Hit Accuracy

As traditional hit accuracy evaluation method based on rating-based applications; we needed modifications in it for making it applicable to our recommendation engine. Therefore we made some assumptions and determine user interest according to the assumptions we described in 4.2.4.

We try to evaluate how successfully we are in predicting the users' interested items. Our methodology works like this; for each interested item of users we recommend 5 products to users and check whether our recommended items include rest of the items that the user interested in. Requirements for measuring hit accuracy is that the user should at least view two items in a session so that we can predict the user's interested items by making computations based on the previous viewed product of the user. Figure 9 explains our approach in the evaluation of hit accuracy.

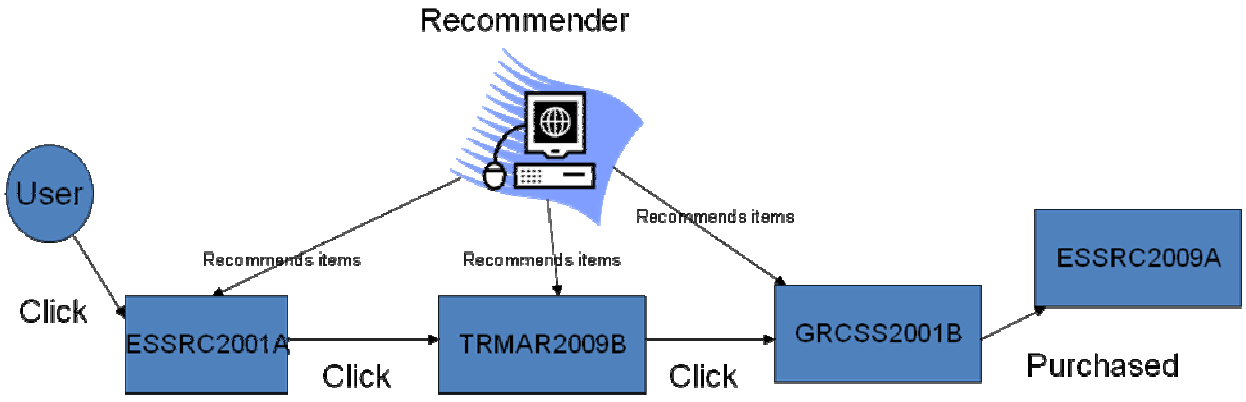

Figure 9: Evaluation of Hit Accuracy

In this evaluation technique we only evaluate the performance of our "Item to Item Collaborative Filtering" Algorithm as we make all the recommendations in this evaluation by using this algorithm.

5.1.2 Results of Hit Accuracy for Pure Collaborative Filtering

Here are the results for this evaluation technique.

	Training Day Type: All Days	**Training Day Type: Week Days**	**Training Day Type: Weekends**
Test Day Type : All Days	%29	%31	%32
Test Day Type : Week Days	%28	%31	%31
Test Day Type : Weekends	%30	%32	%35
Number of Sessions Used in Test Data Set	**389**	**286**	**108**

Table 4: Hit Accuracy Results

5.1.3 Interpretation of Results

As it can be seen our results didn't have very high accuracy and actually this is reasonable if we consider the purpose of our algorithm and the evaluation technique.

For making a proper interpretation of the results; we first need to understand what this evaluation technique evaluates. Traditional hit accuracy evaluation technique is mostly used for predicting what rating the user give to a new item considering the ratings the user have given for the other items.

In this approach we extended this evaluation one step further and tried to guess which item the user would be interested in by considering their previous interested items. For doing so we first retrieved the most similar items to the users' interested items and we checked whether the user really interested in those items later in their navigation.

However this technique can not measure whether the items that we recommended to our users, was indeed interesting or wasn't interesting at all. Because in the end our algorithm tries to recommend items that are most similar to the items the user interested in; but in the end it is highly possible that in their navigation users didn't check all the similar items; but only checked items that are first presented to them in their search results. Therefore it is quite normal that the items that we suggested to our users don't comply with what they really clicked on.

Another drawback here was we didn't use any content-based filtering but the search results shown to users were always content-filtered; and because of this characteristic of the web logs; it was pretty difficult for us to recommend same items to users clicked on their navigation.

For example if we determined in our algorithm that a holiday GR1 from Greece and holiday TR1 from Turkey is indeed very similar; and if we recommended TR1 to a user who showed interest in GR1; it is highly possible that it will result in a low precision because of the content-filtering nature of the application. As the user most probably clicked on GR1 because they searched for holidays in Greece; in the results they didn't have TR1 and the user didn't have any chance to show interest in that.

Therefore we believe that if we filtered our results in the end according to the content information; we would have much higher accuracies; but this would also disorient us from our aim. Because this mentality is already presented in the application itself; as the search mechanism filters out all the results according to the given content information; and the recommendation engine should be showing other items that would get interest of users even though the recommended items don't comply exactly with the given content information.

But also we are aware that; we should find a way to evaluate the performance of our engine and we decided to change our hit accuracy evaluation technique slightly and come up with the following evaluation technique which we called "Hit Accuracy for Evaluating the Success of Determination of Similar Items".

5.2 Hit Accuracy for Evaluating the Success of Determination of Similar Items

This technique evaluated whether the recommended items to the users indeed similar with the items that the users showed interest or not. Our motivation in this technique it was very difficult for us to predict the exact items the user showed interest without considering content information.

Therefore we needed a more realistic approach for evaluating our performance; and we determined that we should check whether we recommend items to users that are similar to the items that they show interest; and if it is the case it shows that our recommendation engine is successful in recommendation process. Because in a way it would complement the search mechanism of the application itself by recommending items that are similar to items the users showed interest but don't have similar content information. As search mechanism itself is enough for showing results that already comply with the content information; our goal in the recommendation engine should be making a difference than this search mechanism by recommending similar items that don't have to comply with content information.

5.2.1 Definition of Hit Accuracy for Evaluating the Success of Determination of Similar Items

As we used adjusted cosine similarity for calculating user similarity, positive similarity values shows that the given items are indeed similar. So in this evaluation technique we checked whether the recommended items and the items user showed interest have positive similarity values and we used the accuracy of these results as the performance indicator of our engine. In the next section we will give our results for this hit accuracy evaluation.

5.2.2 Results of Hit Accuracy using Pure Collaborative Filtering for Evaluating the Success of Determination of Similar Items

Here are the results for this evaluation technique.

	Training Day Type: All Days	**Training Day Type: Week Days**	**Training Day Type: Weekends**
Test Day Type : All Days	%38	%40	%46
Test Day Type : Week Days	%37	%40	%47
Test Day Type : Weekends	%39	%43	%45
Number of Sessions Used in Test Data Set	**1294**	**990**	**285**

Table 5: Hit Accuracy Results for Determination of Similar Items Using Pure Algorithm

5.2.3 Interpretation of Results

Compared to the previous hit accuracy technique; results show that our recommender engine has better performance in predicting the similarities of items. Here it is also seen that contextual information plays a very important role in recommendation process. As can be seen there is %7-8 of improvement in the accuracies when the data set is trained with sessions from weekend days. We think this is a result of the fact that; users spend more time on search on weekends as they might have more free time; and this leads us to better determine the users' interests on items.

Another interesting fact here is that; we make better predictions in sessions that end in purchases. Our average accuracy in this case is %48. This shows that our algorithm gives more importance to items that are purchased which is requested by Bizztravel too. Improvement in the results is logical because we used a weighted algorithm to determine user interest; we put more weights to items that are purchased so this led us to make better connections between the interested items and the purchased items.

In the next section we will show the results that we achieved using hybrid algorithm rather than pure algorithms.

5.2.4 Results of Hit Accuracy using Hybrid Algorithm for Evaluating the Success of Determination of Similar Items

Here are the results for this evaluation technique using hybrid algorithm.

	Training Day Type: All Days	**Training Day Type: Week Days**	**Training Day Type: Weekends**
Test Day Type : All Days	%57	%54	%50
Test Day Type : Week Days	%58	%55	%50
Test Day Type : Weekends	%55	%50	%50
Number of Sessions Used in Test Data Set	**1294**	**990**	**285**

Table 6: Hit Accuracy Results for Determination of Similar Items Using Hybrid Algorithm

5.2.5 Interpretation of Results for Hybrid and Pure Algorithms

As the results clearly show that, hybrid algorithm highly outperforms the pure algorithm; the difference between the hit accuracies is around %10-15. This result is not a surprise for us because of the following reasons:

- Hybrid algorithms are able to model the real recommendation process better than pure algorithms as they take several perspectives into account which is the case in real life too.
- Pure collaborative filtering was lacking the content information to make proper recommendations; because this application is based on content information search; therefore users are biased toward navigating in similar content space. Therefore not considering content information at all was producing recommendation from a larger content space than the user is navigating through which was decreasing the performance.

These results clearly show that in our application content information has a very important role, especially because of the fact that the application itself is driven through content information; our recommender must consider content information in the recommendations too.

One might think that the best approach will be just making pure content based recommendations instead of hybrid recommendations, but we also applied a pure content-based algorithm to see the results and actually the results were not as high as hybrid recommenders (around %45). Therefore we preferred to use hybrid algorithm in our final product. Actually even the results of the content based algorithm was higher; we would still prefer to use hybrid algorithm as our goal here is to recommend items with varying features to users so that they see more than one type of options in navigation. The search mechanism of the application itself is quite enough for helping users navigate through a content-based item space.

Therefore we concluded to use the hybrid model in our application so that we can recommend items to users which are interesting for them and also by collaborative filtering we hope to cover the item space more than a content-based algorithm.

5.2.6 Comments on Affects of Contextual Information in the Recommendations

Even though the results are not very high for any case in hit accuracy evaluations; there were still slight improvements in hit accuracies when the context information is considered. The hit accuracies improved to %31 from %29 when the recommendations are based on the training data captured from week days and results improved to %35 from %29 percent for weekend days. However for the hybrid case, it looks like the context information didn't have any benefits for the recommender.

Although the results are still not very high; it is obvious that putting contextual information into the algorithm improves the qualities of recommendations for pure collaborative algorithm. Especially if we consider that; this information is the only information used as context information in this algorithm as this algorithm purely based on collaborative filtering. These results point out the fact that this application is really affected a lot by content and context information. Also the results of the hybrid algorithm points out that the content information has more importance than context information actually; as the addition of content information to collaborative filtering definitely improved the results but the context information along with content information has no significant effect on the recommendations. This is quite normal as we described above several times that; the web logs we are using right now are affected a lot by content-space as users are directed in the web site purely by content information.

Another interesting point here is that there are higher accuracies for weekend data than for week data. This could be explained by considering the fact that users have more time to focus into the application on weekends than week days. As in this algorithm we measured the users' interest according to the number of clicks they made; it makes sense to have better accuracies in weekend than week days because; as people have more free time in weekend than week days; it is more likely for us to determine users' interest in items correctly in weekend days. Because users would spend more time in items they are really interested in by clicking more on them to get details of it where as in week days it is more likely for users to just spend same amount of time on items to just get a quick glance on the options they have.

We will continue with our comments about the importance of the contextual information in recommendation process in the section 5.3.4 after considering the results of step reduction evaluation. In the next section we will continue with this evaluation method.

5.3 Step Reduction Metric

In this technique; we want to see the performance of our algorithm based on search terms. One of our goals in this project is to reduce the number of clicks in a website the user made to reach their goal. More specifically we want to reduce the search time user spent and led them to purchases.

Our motivation behind this is that Bizztravel is not the only holiday website in Netherlands and they have many competitors and we want to reduce the chance of losing a potential customer as much as possible by leading them to items that they might be interested in.

Here we only analyzed sessions end with a purchase; because these are the only type of sessions that we can exactly determine whether we made a correct recommendation or not. Because if a user purchased an item; it shows that the user liked that item; and from their previous search terms if we can suggest the purchased item to the user; it shows that we have created a link between these search terms and items. Then these links could be used to direct users to items that they might consider buying.

5.3.1 Definition of Step Reduction

We followed the following steps in our evaluation. We first retrieve all the sessions resulted with a purchase which has happened after making some search in the website. Otherwise if the purchase has happened without making any search; they are not considered which is reasonable; as these users already decided what they are looking for and they indeed don't need any recommendation.

For each search term used before the purchase; we recommended 5 holiday packages to user and check whether the purchased product is in the recommended items set. If so, we calculate in the real session how many more searches the user made after this search and the calculated number is the number of reduced step. Figure 10 is the illustration of our approach.

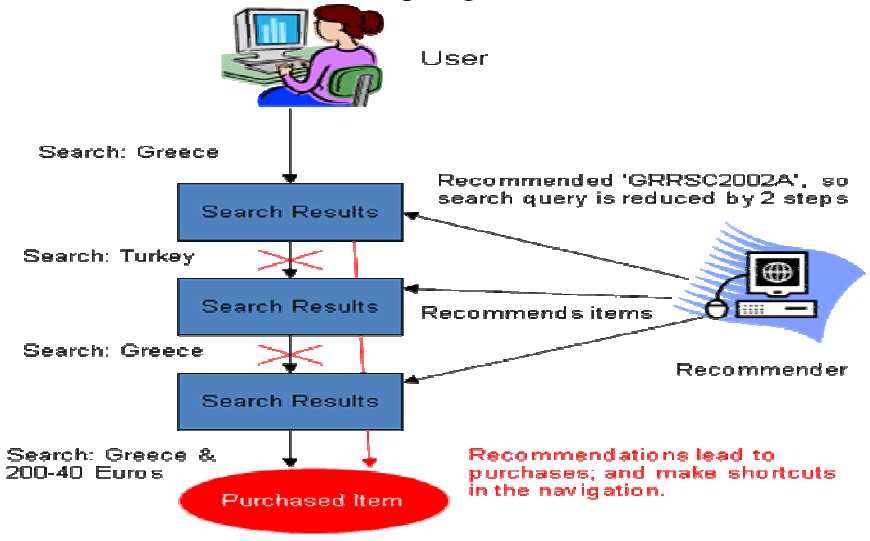

Figure 10: Visualization of Step Reduction Evaluation

5.3.2 Results for Step Reduction

Here are the results for this evaluation technique.

	Training	Training	Training

	Day Type: All Days	Day Type: Week Days	Day Type: Weekends
Test Day Type : All Days	1.82	2.10	0.35
Test Day Type : Week Days	2.86	3.34	0.14
Test Day Type : Weekends	0.00	0.00	0.66
Number of Purchases Used in Test Data Set	**73**	**62**	**11**
Average Number of Searches Per Session	**4.68**	**4.57**	**5.03**

Table 7: Average Number of Step Reduction in Purchases

5.3.3 Interpretation of Results

Our results show that our algorithm is quite successful in predicting what the user will buy by considering the used search keywords. If we consider average number of searches happening; we see that almost in average we make approximately %40 of step reduction which is a very high amount considering the fact that even %5 increase is considered as significant improvement in research community.

Interestingly our other algorithm; item-to-item collaborative didn't seem to have the same success in prediction of items the user would be interested in; but our results here also support us in our claim that the web logs and the users' navigation is affected too much by content information. As the search results given to users are exactly filtered according to the given search keywords; users are biased toward searching in same content space.

Because of this fact our algorithm that takes content information more into account looks like have a better performance in predicting items that the user will show interest in than our item-to-item collaborative filtering algorithm which doesn't consider content-information at all. We can easily improve the results of our other algorithm by adding content-filtering to the algorithm itself but we prefer not to do that because as we stated above; we believe that recommendation engines should have a different working principles than search engines which doesn't consider item space that don't comply with the given search query vector.

And our results actually made us believe that; we are on the right track as both of our algorithms in a way complement each other; i.e. algorithm based on search keywords is successful in making recommendations based on content information and item-to-item algorithm is successful in making recommendation of similar items to users' interested items that have different content information.

5.3.4 Comments on Affects of Contextual Information in the Recommendations

From our results it is seen that; putting the type of day into the recommendation process improves the prediction accuracies. For week days it is seen that; if we make recommendations based on the training data that consists of week day transactions rather than using all data set; the step reduction improves to 3.34 from 2.86 which is almost an improvement of %16. The similar effect is also seen for the weekend days; but because of the limited data used in weekend data the results are not very reliable. Results show that for weekend days; average step reduction jumps 0.6 from 0 which shows that using the whole data set for weekend data results in making wrong predictions.

These results support our claim about the importance of the usage of contextual information. It is obvious that considering contextual information while making recommendations improves the qualities of recommendations. Usage of contextual information such as “Price” and “Number of Travelers” implicitly in our algorithms resulted in high accuracies for predicting products that the users purchased.

Especially separation of day types and improvements in the recommendations process shows that users have different navigation behaviors under different context information which is the day type in this case. It shows that users’ decision in making purchases on week days are different than on weekend days.

These results should be reflected on the recommendations engine models. For example for users in week days, recommending holiday packages that complies with the search keywords would improve the satisfaction of customers and recommending holiday packages with varied characteristics would be more suitable for customers in weekend as they prefer to analyze all their options before making a purchase.

5.4 Comments on User Anonymity

Recommendations engines were mostly used in application areas where the users can be identified and there have been lots of researches in this area for identified users. However there are still not many researches going on for making recommendations based on anonymous users. Even more importantly most of the evaluation techniques for evaluating performance of recommendations engines are designed for identified users and new techniques are needed for anonymous user based recommendation engines.

For example as stated above one of the most famous evaluation technique is hit accuracy and with its current version; it is very difficult to apply it for our case because this method requires explicit ratings from users and then it has enough data for evaluating recommenders. However we don’t have any explicit feedback from users and we just try to come up with our schema to determine the users’ interest on items which is biased because of unreliability of implicit feedbacks; such as counting on number of clicks as a sign of interest can lead to poor results because it actually doesn’t tell whether the user really liked the item or not.

Also the problem of sparse data in general recommender engines is even a bigger problem with anonymous user space; because here the data set becomes even sparser. As even though the different sessions can belong to same users because of anonymity we can’t determine that and this reduces the chance of understanding similarities of items between each other. Therefore in many cases making recommendations based on item-to-item filtering is not possible as there isn’t any information about most of the item pairs.

Nevertheless usage of content and context information can reduce the deficiencies in these cases as our application is a very content-dependent one; usage of content information can help us make better recommendations easily and provides us a solution for dealing with sparse data set. Also usage of context information helps us making better connections between the similar users. Even a simple separation such as separation of day types helps us improve our recommendations by modeling our recommender with users who are more similar to current users.

6 Conclusions

In this section we want to present our conclusions that we learned from this project. In the end we explained how this project might be improved and we tried to guide the interested researchers who want to work in this field.

6.1 Pure Algorithms vs. Hybrid Algorithms

Our results showed that in our application hybrid algorithm which combines the content and collaborative filtering performs much better than pure algorithm (hybrid outperforms pure collaborative filtering by almost 15%). We expected to have a result like this; because pure collaborative filtering doesn't consider the content information at all which creates a lot of deficiency as content information is used a lot in the application itself. As our application was driven by the search mechanism; content information has high importance and the navigation behavior of users is highly influenced by the content information. Therefore a pure algorithm which doesn't use content information will lack from it. On the other hand our hybrid algorithm combined collaborative filtering with content information and this led to much better results.

In our opinion, hybrid recommenders in most of the cases would outperform pure algorithms because of the nature of recommendation process. The main goal of researchers is to model the recommendation process so that it can be automatically performed by computers. Therefore the modeled process should reflect all the properties of the real recommendation process that is performed between people.

This real recommendation process is a very complex procedure which cannot be modeled by pure algorithms which only consider one part of recommendation process such as considering only content information or considering only social effect by collaborative filtering. In a real recommendation process; people consider tons of issues ranging from content to context such as season, weather and companion information and from friendship type to history of our friends and even more. For example when people recommend a movie to their friends; they first consider which movies are watched by these friends, what type of movies they liked, with whom they will watch this movie, when will they watch it and etc.

As you can see we considered a lot of different issues while making recommendations and therefore in the recommendation model these issues should be considered too. Hybrid algorithms even though still not enough to reflect all dimensions due to the complexity of the real process; are the best approaches for covering the most of the dimensions in the recommendation process.

6.2 Importance of Content and Context Information

Recommendation engines can be seen as suggesting items to users that might be interesting for users. In this process the most important thing is the representation of users and items. About users most of the time we don't have too much information and because of the complexities users are tried to be mapped to the items so that items corresponding to the interested items of the users or suited to the user profile can be recommended. So in this approach the representation of items is very important and the most important feature set of the representation is content which describes the items in terms of properties that separates them from the rest of the item space such as describing a music album in terms of genre, artist etc.

One advantage of the items is that; they are most of the time static their features don't change; such as a music song will always have the same artist and same duration. However important issue here is selecting the correct feature space so that it really represents the item

correctly and it really separate it from the rest of the item space. Especially for the separation of items from each other, some items can be represented with a different feature set which can describe the context of the item. For example in our application; hotels from same region has more or less same content information but some of these hotels are more suitable for travelers looking for a cheaper holiday whereas some are more suitable who are looking for a place to stay with a crowded group.

In our application we achieved this feature by mapping items to search parameters such as recommending a holiday package to a user looking for eight person room after seeing that this recommended hotel is clicked on a lot by the visitors searching for an eight person room. So therefore in our application we represented our items in both content (destination, all inclusive, etc.) and context information (price range, number of person, etc.).

We believe that by doing so we created an opportunity to map the users to the items according to the context and content they are looking for. For example if we only know the user is looking for a holiday in Bulgaria; it is not easy for us to make good recommendations but as soon as the user start showing some sign that the user is actually interested in a cheap holiday; then we can recommend cheaper holidays from other countries to the user too. Therefore we believe that content and context information should always be part of the recommendation process.

Also our results showed that integration of context (5% improvement in hit accuracy) and content information (%15 improvement in hit accuracy) to the pure collaborative filtering improves the performance of our recommender. Integration of both content and context improves the results by 10%; so integration of content rather than both content and context has a better effect. This might be a result of the fact that our application and the usage data that we used to train our recommender is highly influenced by content information as the users are navigated through the website by content information.

Unfortunately in our application we couldn't test the effect of knowledge based component even though we implemented it. This is because we didn't have time to collaborate with the domain experts to put their campaigns into our knowledge based component. Also because of Bizztravel's requirements on the knowledge based component, it isn't implemented in a way to improve the results but to recommend promotions of the company regardless of what the user is searching for.

6.3 Remarks on Working with Implicit Feedback

One of the biggest problems in recommendation engines is the reluctance of users in giving explicit feedback. Even in better cases where the users give feedback such as reviews; they are not easy to be used as they are not understandable by computers and need advanced natural language processing. The problem here is that recommendation engines have to get some feedback from users so that according to them recommendations can be displayed to users. So one solution to getting feedback is not asking directly from the users but just guessing it according to the navigation behavior of the users and that is exactly what we tried to do in our application.

According to the items the users clicked on, according to the keywords the user entered; our system tries to model the users so that we can recommend items similar to these created models. However we also suffered from the known problem of implicit feedbacks which is unreliability of them. Nice thing about explicit feedback is that when the users click on the I like it button, it means that the user really liked the item; however for the implicit feedback case; if the user spent

ten minutes on viewing an item it doesn't necessarily mean that the user really liked the item but it can also mean that the user went to toilet.

Unreliability of the implicit feedback created a lot of problems for us in the determination of the users' interest on items and in the end we decided to use number of clicks on an item for determining user interest. However we are aware that even this criterion is not very reliable but at least we saw that our results got improved with this approach compared to our previous trials where we used click on an item regardless of how many times it is clicked as a sign of interest.

Nevertheless we concluded that for the recommendation engines the best approach is motivating users to give explicit feedbacks because these feedbacks are the most reliable facts that recommendations can be made upon. For the motivation of the users we believe that best approach would be making the users feel that they are part of a community and made them believe that they are incorporating into something and their community is improved by their feedback.

So the idea in this context should be improving the interaction between the users so that they don't see giving feedback as a task but as a way of improving the environment they are working in. And of course giving feedback should be very easy such as just clicking on a link and should not interfere with the navigation of the user. One example could be the "Liked It" link of Facebook; where the user can easily click on it and it doesn't even need page refreshment. It is seen that many users are using this link to show their interest on the objects they find interesting in their friends' pages. This might be a result of the fact that after showing this kind of interest, users get feedback from their friends too; so in a way giving feedback is not a task for the user but is an initiation for starting a social activity. This approach should be applied in recommendation engine based applications to improve the interaction between users and application which will lead to better recommendations.

6.4 General Remarks about Recommendation Engines

While working in this project we encountered with lots of difficulties and these difficulties made us understand the obstacles in front of recommendation engines and what are needed for recommendation engines. Here in this section we want to share these insights with you.

First of all in the development process we understand that our application has a rather different characteristic than the famous applications such as Amazon and Last.fm. From the beginning we were aware of the fact that the item space in Amazon was much larger than our system and for them the main difficulties were encountered with scalability. But later we see that the huge difference in item space is not the only difference with our application as in these applications the interaction with the users, the transaction frequency is much higher compared to our application.

For example in Amazon; a user can visit the web site any time in the year; and can come next day to check another product right after buying an item. However in our application; as it only sells summer holiday packages; most of the times users only use this website for one time i.e. after purchasing a holiday the users don't use the website until next year. Therefore this makes recommendation process difficult for us; because the real sign of interest about items is purchasing and as soon as we learn something important about our customer we were losing the chance to make recommendations to them as they stop using the application. So we concluded that recommendation engines are not suited very well for applications where the transaction frequency is low.

Because of this observation we believe that; recommendation engines are more suitable for applications where the transaction and interaction frequency are higher. And of course having a larger item space is also an advantage for recommendation engines; because with larger item space users have really no chance to see all the products; so a clever recommendation engine can come up with the most relevant items for the users. However in our case the item space was rather small around 400; and if the user already has a destination in mind the item space was reduced to only 20 items and after that there is no use of using recommendation engines; because most often the users would just look at the details of each item and choose the best one.

Another important concept is determination of user interest. We already discussed this issue in section 6.3 but here we want to briefly go over it again. This is a very important issue especially for recommender systems working with implicit feedback because in these applications there isn't a standard method for understanding the users' preferences and the developer itself needs to determine the criteria for indication of interest.

Some of the implicit feedback measures for measuring the interest are duration spent on a page, number of clicks etc. Actually the best criteria might be time spent on a page as it really reflects how much the user really interested in an item; but in our application and actually in most of the applications in internet, time spent couldn't be used because if the user opened several pages at the same time in different tabs, all of them will have the same starting time and even though the user only looks at one page at a time; it seems like user spent more time on other pages opened in different tabs. This is a difficult problem to solve for developers because most of the web mining techniques can't calculate exactly how much the user spent looking at a page; and web mining techniques should be improved to measure this correctly.

Regardless of the type of feedback i.e. implicit or explicit; for a recommendation engine to work properly users should be able to express their interest on items and these signs should be as reliable as possible. For example rating based systems compared to implicit feedback based systems are much more reliable. Recommendation engines are not suitable for applications where users cannot show their interest on items therefore the application developers need to find ways for providing environments to the users so that users can express their interests on items.

6.5 Future Work

We finished our application but research in this field is far away from ending and there are still lot of potential and opportunities for improving this project. First of all in our system all the evaluations are applied through theoretical techniques but the real performance of our system can only be seen when it is applied in real environment and see how the users are using our recommendations.

Apart from that user interest could be determined by learning how long the user viewed each page rather than number of clicks. However to achieve this web mining phase should be improved to understand exact time the user spent on each page.

Also we believe that affect of contextual information on the recommendations should be tried in different applications too and our application should be modified to be worked in an identified user environment; so that recommendations can be based on history of users rather than only making recommendations based on the current session. In this thesis; one of our main objective was to see how the recommenders perform with anonymous users; but we believe that working with identified users would improve the performance of recommendation engines; as the user models will be more reliable and complete than anonymous users. Also the evaluation of the recommender engine will be more reliable as the evaluation techniques such as hit accuracy are

more trustable when it is used with identified users' data. In this new application the developed recommender can contain all the modules that are given in our general architecture.

Frequent navigation pattern analysis can also be applied in our application for making recommendations according to the search path of the users. For example there might be commonly used search paths in this application, then for the new users who might be on the same track with these popular paths; recommendations or guidance in their navigation might be given to these users.

Another good approach will be running the current application of Bizztravel with our recommender engine in live environment and collect the data of the developed application. Then using this data the same evaluations should be applied to see whether the recommendation engine itself is performing better or not. This will reduce the dependency to the content information in the system as the current application heavily depends on it and with this approach the real importance of recommendation modules i.e. collaborative filtering, content, context information, etc. will be measured better.

In the end it is obvious that there are still lots to do for improving the performance of our recommender by determining the user interest with different techniques; or making recommendations using different context information such as price, number of travelers rather than week day type and hopefully in the future researchers will achieve better results by applying new techniques.

7 References

1. Kosala, R. and Blockeel, H. 2000. Web mining research: a survey. *SIGKDD Explor. Newsl.* 2, 1 (Jun. 2000), 1-15.

2. Eirinaki, M. and Vazirgiannis, M. 2003. Web mining for web personalization. *ACM Trans. Internet Technol.* 3, 1 (Feb. 2003), 1-27.

3. Srivastava, J., Cooley, R., Deshpande, M., and Tan, P. 2000. Web usage mining: discovery and applications of usage patterns from Web data. *SIGKDD Explor. Newsl.* 1, 2 (Jan. 2000), 12-23.

4. Facebook's New Privacy Policy, http://blogs.ft.com/techblog/2009/02/facebook-retreats-after-latest-privacy-row/

5. Truste: Building a web you can believe in. http://www.truste.org

6. Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.* 23, 1 (Jan. 2005), 103-145.

7. Anand, S. S., Kearney, P., and Shapcott, M. 2007. Generating semantically enriched user profiles for Web personalization. *ACM Trans. Internet Technol.* 7, 4 (Oct. 2007), 22.

8. Smyth, B. and McClave, P. 2001. Similarity vs. Diversity. In *Proceedings of the 4th international Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development* D. W. Aha and I. Watson, Eds. Lecture Notes In Computer Science, vol. 2080. Springer-Verlag, London, 347-361.

9. Ziegler, C., McNee, S. M., Konstan, J. A., and Lausen, G. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international Conference on World Wide Web* (Chiba, Japan, May 10 - 14, 2005). WWW '05. ACM, New York, NY, 22-32.

10. Koutsabasis, P. and Darzentas, J. 2008. Item-Based Filtering and Semantic Networks for Personalized Web Content Adaptation in E-Commerce. In *Proceedings of the 5th Hellenic Conference on Artificial intelligence: theories, Models and Applications* (Syros, Greece, October 02 - 04, 2008). J. Darzentas, G. A. Vouros, S. Vosinakis, and A. Arnellos, Eds. Lecture Notes In Artificial Intelligence, vol. 5138. Springer-Verlag, Berlin, Heidelberg, 148-159.

11. Billsus, D. and Pazzani, M. J. 2000. User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction* 10, 2-3 (Feb. 2000), 147-180.

12. Rocchio J (1971) Relevance feedback in information retrieval. In: Salton G, Ed., The Smart Retrieval System-- Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, NJ, pp. 313-323.

13. Pazzani, M. and Billsus, D. 1997. Learning and Revising User Profiles: The Identification ofInteresting Web Sites. *Mach. Learn.* 27, 3 (Jun. 1997), 313-331.

14. Balabanović, M. and Shoham, Y. 1997. Fab: content-based, collaborative recommendation. *Commun. ACM* 40, 3 (Mar. 1997), 66-72.

15. Strands: Plug and Play Recommendation Engine for eCommerce and Media, http://blog.strands.com/2008/06/10/strands-launches-plug-and-play-recommendation-engine-for-ecommerce-and-media/

16. Schafer, J.B., Frankowski D., Herlocker, J., Sen, S. 2007. Collaborative Filtering Recommender Systems. Lecture Notes in Computer Science, Vol. 4321, May 2007, pp. 291-324.

17. Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international Conference on World Wide Web* (Hong Kong, Hong Kong, May 01 - 05, 2001). WWW '01. ACM, New York, NY, 285-295.

18. Chedrawy, Z., Abidi, S.S.R. 2006. An Adaptive Personalized Recommendation Strategy Featuring Context Sensitive Content Adaptation. Lecture Notes in Computer Science, Vol. 4018, June 2006, pp.61-70.

19. Baltrunas, L. 2008. Exploiting contextual information in recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems* (Lausanne, Switzerland, October 23 - 25, 2008). RecSys '08. ACM, New York, NY, 295-298.

20. Delgado, J., Ishii, N. 1999. Memory-based weighted majority prediction for recommender systems. ACM SIGIR'99 Workshop on Recommender Systems.

21. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. CSCW '94, 175--186.

22. Harper, F. M., Li, X., Chen, Y., Konstan, J. A. 2005. An economic model of user rating in an online recommender system. In User Modeling, page 307, Edinburgh, Scotland, Springer Berlin.

23. Massa, P., Bhattacharjee, B. 2004. Using trust in recommender systems: an experimental analysis. Proceedings of 2nd International Conference on Trust Managment, Oxford, England.

24. Herlocker, J. L., Konstan, J. A., and Riedl, J. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work* (Philadelphia, Pennsylvania, United States). CSCW '00. ACM, New York, NY, 241-250.

25. Zaslow, J. 2002. If TiVo Thinks You Are Gay, Here's How To Set It Straight -- Amazon.com Knows You, Too, Based on What You Buy; Why All the Cartoons?. The Wall Street Journal, sect. A, p. 1.

26. Chakrabati, S., Dom, B., Gibson, D., Kleinberg, J., Kumar, S., Raghavan, P., Rajagopalan, S., Tomkins, A. 1999. Mining the Link Structure of the World Wide Web *Computer,* vol. 32, no. 8, pp. 60-67.

27. Eirinaki, M. and Vazirgiannis, M. 2007. Web site personalization based on link analysis and navigational patterns. *ACM Trans. Internet Technol. 7*, 4 (Oct. 2007), 21.

28. Pabarskaite, Z. and Raudys, A. 2007. A process of knowledge discovery from web log data: Systematization and critical review. *J. Intell. Inf. Syst.* 28, 1 (Feb. 2007), 79-104.

29. Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. 2001. Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd international Workshop on Web information and Data Management* (Atlanta, Georgia, USA). WIDM '01. ACM, New York, NY, 9-15.

30. Catledge, L.D., Pitkow, J.E. 1995. Characterizing browsing strategies in the World Wide Web. http://www.igd.fhg.de/www/ www95/papers/80/userpatterns/UserPatterns.Paper4.formatted.html.

31. Cooley, R., Mobasher, B., Srivastava, J. 1999. Data Preparation for Mining World Wide Web Browsing Patterns. Knowledge and Information Systems, vol. 1, no. 1, pp. 5-32.

32. Burke, R. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction* 12, 4 (Nov. 2002), 331-370.

33. Nguyen, N.T., Rakowski, M., Rusin, M., Sobecki, J., Jain, L.C. 2007. Hybrid Filtering Methods Applied in Web-Based Movie Recommendation System. KES 2007/WIRN 2007, Part I, LNAI 4692, pp. 206–213.

34. Herder, E., Karger, P. 2004. Hybrid Personalization for Recommendations.

35. Burke, R. 2004. Hybrid Recommender Systems with Case-Based Components. Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 pp.91-105.

36. Sobecki, J. 2007.Web-based system user interface hybrid recommendation using ant colony metaphor. *Knowledge-Based Intelligent Information and Engineering Systems,* pp. 1033-1040.

37. Thor, A., Rahm, E. 2004. AWESOME: a data warehouse-based system for adaptive website recommendations. In *Proceedings of the Thirtieth international Conference on Very Large Data Bases - Volume 30*, pp.384-395.

38. Shahabi, C. and Chen, Y. 2003. An Adaptive Recommendation System without Explicit Acquisition of User Relevance Feedback. *Distrib. Parallel Databases* 14, 2 (Sep. 2003), 173-192.

39. Eirinaki, M., Lampos, C., Paulakis, S., and Vazirgiannis, M. 2004. Web personalization integrating content semantics and navigational patterns. In *Proceedings of the 6th Annual ACM international Workshop on Web information and Data Management* (Washington DC, USA, November 12 - 13, 2004). WIDM '04. ACM, New York, NY, pp.72-79.

40. Albanese, M. Picariello, A., Sansone, C., Sansone, L. 2004.Web Personalization Based on Static Information and Dynamic User Behavior. Sixth ACM CIKM International Workshop on Web Information and Data Management, pp.80-87.

41. Adomavicius, G., Tuzhilin, A. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. on Knowl. and Data Eng.* 17, 6 (Jun. 2005), pp. 734-749.

42. Meteren, R.V., Someren, M.V. Using Content-Based Filtering for Recommendation. http://www.ics.forth.gr/~potamias/mlnia/paper_6.pdf

43. Pandora Music Recommender, Pandora.com.

44. Papagelis, M., Rousidis, I., Plexousakis, D., Theoharopoulos, E. 2005. Incremental Collaborative Filtering for Highly-Scalable Recommendation Algorithms.

45. Mahmood, T., Ricci, F., Venturini, A. 2009. Hopken, W. Adaptive Recommender Systems for Travel Planning.The 16th International Conference on Information Technology and Travel & Tourism.

46. Feng, X., Luo, Q. 2007. Research on Personalized Recommendation Algorithm in E-Supermarket System. Advances in Web and Network Technologies, and Information Management, Volume 4537/2007, pp 340-347.

47. Jalali, M., Mustapha, N., Sulaiman, N. B., and Mamat, A. 2008. A Web Usage Mining Approach Based on LCS Algorithm in Online Predicting Recommendation Systems. In *Proceedings of the 2008 12th international Conference information Visualisation*. IV. IEEE Computer Society, Washington, DC, pp. 302-307.

48. Eirinaki, M., Vazirgiannis, M., and Kapogiannis, D. 2005. Web path recommendations based on page ranking and Markov models. In *Proceedings of the 7th Annual ACM international Workshop on Web information and Data Management*. WIDM '05. ACM, New York, NY, pp. 2-9.

49. Semeraro, G., Lops, P., and Degemmis, M. 2005. WordNet-based User Profiles for Neighborhood Formation in Hybrid Recommender Systems. In *Proceedings of the Fifth international Conference on Hybrid intelligent Systems*. HIS. IEEE Computer Society, Washington, DC, pp. 291-296.

50. Göksedef, M., Öğüdücü, Ş. G. 2007. A Consensus Recommender for Web Users. In *Proceedings of the 3rd international Conference on Advanced Data Mining and Applications*. Eds. Lecture Notes In Artificial Intelligence, vol. 4632. Springer-Verlag, Berlin, Heidelberg, 287-299.

51. Yoshii, K., Goto, M., Komatani, K., Ogata, T., Okuno, H.G. 2008. An Efficient Hybrid Music Recommender System Using an Incrementally Trainable Probabilistic Generative Model. Audio, Speech, and Language Processing, IEEE Transactions. Volume 16, Issue 2, pp. 435-447.

52. Taghipour, N., Kardan, A. 2008. A hybrid web recommender system based on Q-learning. Proceedings of the 2008 ACM symposium on Applied computing, pp.1164-1168.

53. Statistics about Facebook, http://www.facebook.com/press/info.php?statistics

54. Adomavicius, G. and Tuzhilin, A. 2005. Personalization technologies: a process-oriented perspective. *Commun. ACM* 48, 10 (Oct. 2005), pp. 83-90.

55. Mobasher, B., Cooley, R., and Srivastava, J. 2000. Automatic personalization based on Web usage mining. *Commun. ACM* 43, 8, pp. 142-151.

56. Manber, U., Patel, A., and Robison, J. 2000. Experience with personalization of Yahoo!. *Commun. ACM* 43, 8, pp. 35-39.

57. Baraglia, R. and Silvestri, F. 2007. Dynamic personalization of web sites without user intervention. *Commun. ACM* 50, 2, pp. 63-67.

58. Smyth, B. 2007. Case-Based Recommendation, The Adaptive Web, LNCS 4321, pp. 342–376.

59. Aamodt, A. and Plaza, E. 1994. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun.* 7, 1, pp. 39-59.

60. Linden, G., Smith, B., and York, J. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1, pp. 76-80.

61. Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international Conference on World Wide Web,* WWW '01. ACM, New York, NY, pp. 285-295.