Mission-Based Navigational Behaviour Modeling for Web Recommender Systems[*]

Osmar R. Zaïane[**], Jia Li, and Robert Hayward

University of Alberta, Edmonton AB, Canada
zaiane@cs.ualberta.ca

Abstract. Web recommender systems anticipate the information needs of on-line users and provide them with recommendations to facilitate and personalize their navigation. There are many approaches to building such systems. Among them, using web access logs to generate users' navigational models capable of building a web recommender system is a popular approach, given its non-intrusiveness. However, using only one information channel, namely the web access history, is often insufficient for accurate recommendation prediction. We therefore advocate the use of additional available information channels, such as the content of visited pages and the connectivity between web resources, to better model user navigational behavior. This helps in better modeling users' concurrent information needs. In this chapter, we investigate a novel hybrid web recommender system, which combines access history and the content of visited pages, as well as the connectivity between web resources in a web site, to model users' concurrent information needs and generate navigational patterns. Our experiments show that the combination of the three channels used in our system significantly improves the quality of web site recommendation and, further, that each additional channel used contributes to this improvement. In addition, we discuss cases on how to reach a compromise when not all channels are available.

1 Introduction

A web recommender system is a web-based interactive software agent. It attempts to predict user preferences from user data and/or user access data for the purpose of facilitating users' information needs by providing them with recommendation lists of suggested items. The recommended items could be products, such as books, movies, and music CDs, or on-line resources such as web pages or on-line activities. In general, a web recommender system is composed of two modules: an off-line module and an on-line module. The off-line module pre-processes data to generate user models, while the on-line module uses and updates the models on-the-fly to recognize user goals and predict recommendation lists.

In this chapter, we investigate the design of a hybrid recommender system to recommend on-line resources, with the emphasis of the presence of concurrent

[*] Research funded in part by the Alberta Ingenuity Funds and NSERC Canada.
[**] Corresponding author.

© Springer-Verlag Berlin Heidelberg 2006

information needs. Pursuing more than one goal simultaneously (i.e. concurrent information needs) is fairly common for on-line users, but this fact has so far been ignored by web usage-based recommender systems and the research community. We call these simultaneous goals *"missions"* and we use different information channels to identify them, namely the web access usage, the web content, and the web connectivity. Unfortunately, these channels are not all always available and we need to find compromises depending upon the application at hand. Our preliminary goals are to first accurately identify users' multiple information needs, and then assist them to fulfill their needs by predicting their goals and recommend shortcuts to them. Our system has been designed for and tested on both a generic web server log (University of Alberta Department of Computing Science web server log) and an idiosyncratic log created by VIVIDESK$^{(TM)}$, a commercial desktop application that integrates user accesses to multiple on-line applications and resources for health care providers.

One of the earliest and widely used technologies for building recommender systems is *Collaborative Filtering* (CF) [21] [9]. CF-based recommender systems aggregate explicit user ratings or product preferences in order to generate user profiles, which recognize users' interests. A product is recommended to the current user if it is highly rated by other users who have similar interests to the current user. The CF-based techniques suffer from several problems [20]. First of all, they rely heavily on explicit user input (e.g., previous customers' rating/ranking of products), which is either unavailable or considered intrusive. With the sparsity of such user input, the recommendation precision and quality drop significantly. The second challenge is related to the system scalability and efficiency. For a CF-based recommender system, user profile matching has to be performed as an on-line process. For very large datasets, this may lead to unacceptable latency for providing recommendations.

In recent years there has been an increasing interest in applying web usage mining techniques to build web recommender systems [22] [8] [14] [24]. Web usage recommender systems take web server access logs as input, and make use of data mining techniques such as *association rule* and *clustering* to extract implicit, and potentially useful navigational patterns, which are then used to provide recommendations. Web server access logs record user browsing history, which contains plenty of hidden information regarding users and their navigation. They could, therefore, be a good alternative to the explicit user rating or feedback in deriving user models. In web usage recommender systems, navigational patterns are generally derived as an off-line process.

However, a web usage recommender system which focuses solely on access history has its own problems:

– *Incomplete Information* Problem: One restriction with web server logs is that the information in them is very limited. Thus, a number of heuristic assumptions have to be made to identify individual users, visit sessions, and transactions in order to apply any data mining algorithm. One such assumption is that user information needs are fulfilled sequentially while in practice they are often in parallel.

- *Incorrect Information* Problem: When web site visitors are lost, the clicks made by them are recorded in the log, and may mislead future recommendations. This becomes more problematic when a web site is badly designed and more people end up visiting unsolicited pages, making them seem popular.
- *Persistence* Problem: When new pages are added to a web site, because they have not been visited yet, the recommender system may not recommend them, even though they could be relevant. Moreover, the more a page is recommended, the more it may be visited, thus making it look popular and boost its candidacy for future recommendation.

To address these problems, we proposed a hybrid web recommender system [11], which attempts to use three information channels to model user navigational behavior: web access logs, the structure of a visited web site, and the content of visited web pages. In particular, the approach uses the terms within visited web pages to partition visit sessions into overlapping sub-sessions, called *missions.* Our preliminary experiments [11] [12] demonstrate that combining the different information channels has great potential to improve the quality of recommendation. In this chapter, we build upon our previous work to further test and compare the effectiveness of using information from different channels, and from different channels in combination. The experiment is performed on a dataset provided by a generic web site. Our initial approach makes the assumption that all channels are available, which is true only when the recommendation is done on the web server itself, and when web pages are static. In some scenarios, however, the page content is not readily accessible by the recommender agent. When pages are generated dynamically and their content changes, if the content channel needs to be used, the model has to change to attach the content to the use at access time rather than the page itself. In this chapter, we present our initial model and discuss ways to reach a compromise when not all channels – content in particular – are available. We test our approach on a different datasets with different channels available.

A few combined or hybrid web recommender systems have been proposed in the literature [15] [16]. The work in [15] adopts a clustering technique to obtain both site usage and site content profiles in the off-line phase. In the on-line phase, a recommendation set is generated by matching the current active session and all usage profiles. Similarly, another recommendation set is generated by matching the current active session and all content profiles. Finally, a set of pages with the maximum recommendation value across the two recommendation sets is presented as recommendation. This is called a *weighted* hybridization method [3]. In [16], Nakagawa and Mobasher use association rule mining, sequential pattern mining, and contiguous sequential mining to generate three types of navigational patterns in the off-line phase. In the on-line phase, recommendation sets are selected from the different navigational models, based on a localized degree of hyperlink connectivity with respect to a user's current location within the site. This is called a *switching* hybridization method [3]. Whether using the weighted method or the switching method, the combination in these systems happens only in the on-line phase. Our approach, however, combines different information channels in the off-line phase, and therefore, possesses the advantage of high efficiency. There is other work which discusses the combination of different channels, albeit they were not proposed for use with recommender systems. In [6], Chi et al. develop a system that combines multiple data features of each web page to construct user profiles. The user profile in [6] is built mainly upon the content of web pages, represented by keyword vectors; while the web access log is used to provide weights to individual keywords, giving keywords appearing in more frequently visited pages higher weights. Our approach, on the other hand, makes use of content information to identify missions from usage sessions, to better model users' concurrent information needs and navigational patterns. In [17], Nasraoui et al. define a web session similarity that takes web site structure into account, and hence implicitly fuses structure information to the usage clustering process.

Our contributions are as follows: First, we propose a novel web recommender system, which investigates combining and making full use of distinctive information channels available, such as usage data, content data, and structure data, to improve recommendation quality. Second, we propose a novel notion, *mission*, to capture users' concurrent information needs during on-line navigation, and discuss different ways to identify *missions*. Third, a new on-line navigational model – a mission-based model – is proposed and generated, based on the notion of *mission*. The mission-based model has been proved to better capture users' on-line behavior for the purpose of fulfilling information needs.

This chapter is organized as follows: Section 2 presents the off-line module of our system, which pre-processes available usage and web site data to generate users' navigational models, as well as our on-line module, which generates the recommendation list. In particular, in the off-line module, we present the preprocessing step by step and explain our new notion "mission", how missions model concurrent information needs during visits and how missions are detected using the available information channels. Section 3 presents experimental results assessing the performance of our system on two real datasets. Finally, Section 4 concludes.

2 Architecture of a Hybrid Recommender System

As most web usage recommender systems, our system is composed of two modules: an off-line component, which pre-processes data to generate users' navigational models, and an on-line component which is a real-time recommendation engine. Figure 1 depicts the general architecture of our system.

Entries in a web server log are used to identify users and visit sessions, while web pages or resources in the site are clustered based on their content. These clusters of web documents are used to scrutinize the discovered web sessions in order to identify what we call *missions*. A *mission* is a sub-session with a consistent goal. These *missions* are in turn clustered to generate navigational patterns, and augmented with their linked neighbourhood and ranked based on resource connectivity, using the *hub* and *authority* idea [10]. These new clusters

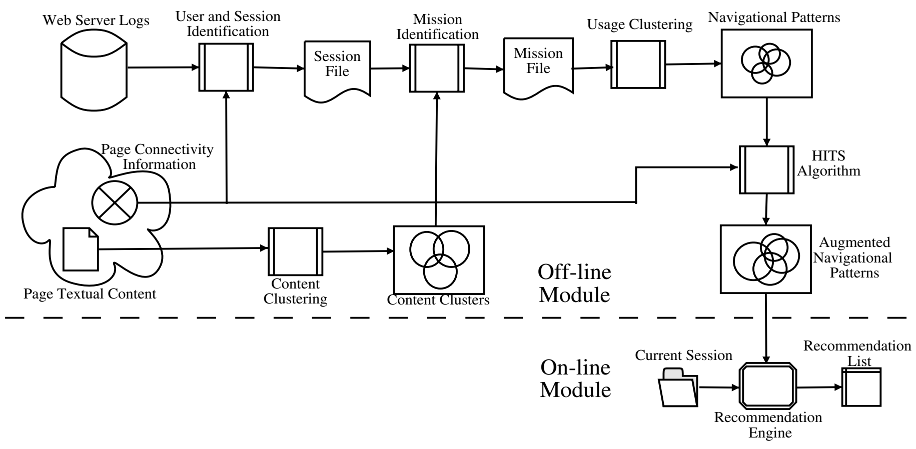

Fig. 1. System Architecture with all three channels available

(i.e., augmented navigational patterns) are provided to the recommendation engine. When a visitor starts a new session, the session is matched with these clusters to generate a recommendation list. The details of the whole process are given below.

2.1 User and Visit Session Identification

A web log is a text file which records information regarding users' requests to a web server. A typical web log entry contains a client address, the requested date address, a time-stamp, and other related information.

For any web access log data, several pre-processing tasks have to be performed before applying data mining techniques for pattern discovery. The pre-processing tasks usually include user identification, visit session identification, and transaction identification. We distinguish two types of web access logs as depicted by our experiments in Section 3: generic web access logs, and session-based access logs. For generic logs, we use similar pre-processing techniques as in [7] to identify individual users and sessions. To sessionize log entries, we chose an idle time of 30 minutes. Session-based access logs, however, have entries identified by users since the users have to log-in, and sessions are already identified since users have also to log-out.

2.2 Visit Mission Identification

The last data pre-processing step proposed in [7] is transaction identification, which divides individual visit sessions into transactions. Two transaction identification approaches are proposed: *Reference Length* approach and *Maximal Forward Reference* approach, both of which have been widely applied in web mining. Rather than dividing sessions into arbitrary transactions, we identify sub-sessions with coherent information needs. We call these sub-sessions *missions*. We assume that a visitor may have different information needs to fulfill during a visit, but we make no assumption on the sequence in which these needs are fulfilled. In the case of transactions in [7], it is assumed that one information need is fulfilled after the other. A *mission* would model a sub-session related to one of these information needs, and would allow overlap between missions, which would represent a concurrent search in the site.

Now how do we identify *missions*? The first approach we proposed to identify *missions* is based on web content [11]. While in the transaction-based model, pages are labeled as *content* pages and *auxiliary* pages, and a transaction is simply a sequence of auxiliary pages that ends with a content page, in the mission-based model we propose, the identified sequence is based on the real content of pages. Indeed, a content page in the transaction-based model is identified simply based on the time spent on that page, or on backtracking in the visitor's navigation. We argue that missions could better model users' navigational behavior than transactions. In our model, users visit a web site with concurrent goals, i.e., different information needs. For example, a user could fulfill two goals in a visit session: a, b, c, d, in which pages a and c contribute to one goal, while pages b and d contribute to the other. Since pages related to a given goal in a visit session are generally supposed to be content coherent, whether they are neighbouring each other or not, we use page content to identify missions within a visit session.

All web site pages are clustered based on their content, and these clusters are used to identify content coherent clicks in a session. Let us give an example to illustrate this point. Suppose the text clustering algorithm groups web pages a, b, c, and e, web pages a, b, c, and f, and web pages a, c and d into three different content clusters (please note that our text clustering algorithm is a soft clustering one, which allows a web page to be clustered into several clusters). Then for a visit session: a, b, c, d, e, f, our system identifies three missions as follows: mission 1: (a, b, c, e); mission 2: (a, b, c, f); and mission 3: (a, c, d). As seen in this example, mission identification in our system is different from transaction identification in that we can group web pages into one mission even if they are not sequential in a visit session. We can see that our mission-based model subsumes the transaction-based model, since missions could become transactions if visitors fulfill their information needs sequentially.

To cluster web pages based on their content, we use a modified version of the DC-tree algorithm [23]. Originally, the DC-tree algorithm was a hard clustering approach, prohibiting overlap of clusters. We modified the algorithm to allow web pages to belong to different clusters. Indeed, some web pages could cover different topics at the same time. In the algorithm, each web page is represented as a keyword vector, and organized in a tree structure called the DC-tree. The algorithm does not require the number of clusters to discover as a constraint, but allows the definition of cluster sizes. This was the appealing property which made us select the algorithm. Indeed, we do not want either too large or too small content cluster sizes. Very large clusters cannot help capture missions from sessions, while very small clusters may break potentially useful relations between pages in sessions.

The mission identification approach above relies on the availability of textual content of web pages, which could not always be satisfied. The purpose of identifying *missions*, however, is to identify users' concurrent information needs in the same visit. With some other specific application access logs, this goal can be achieved by other means. For instance, the URLs recorded in the VIVIDESK$^{(TM)}$ access logs come from different web sites, and a large number of them are dynamically generated. This makes the access to page content for mission identification close to impossible. The alternative, however, is that since VIVIDESK integrates the simultaneous accesses to multiple on-line applications, it records in its logs the application attached to each given access. Therefore, we use the application identifier as an indicator of a mission. Our experiments (see Section 3) show that this is a good approach to identify missions for VIVIDESK$^{(TM)}$ data. Moreover, this generalizes our notion of *mission.* In addition, this highlights the importance to have application related logs rather than just relying on information poor web server logs.

VIVIDESK$^{(TM)}$ (www.vividesk.com) is a commercial system developed by the Centre of Health Evidence at the University of Alberta as a gate to a multitude of applications and on-line resources, and is used by hospital personnel and other health practitioners. It has its specific session-based activity log which records details about user accesses to on-line pages via different applications. The log entries encompass more specific details than typical web server logs and pertain to different web sites rather than just one. Moreover, since users need to authenticate and then safely quit the application, users and exact sessions are automatically identified.

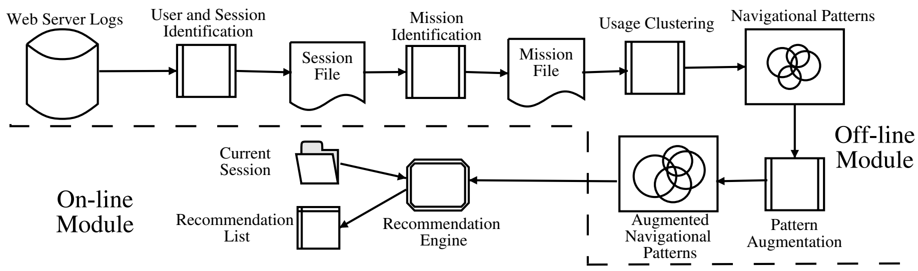

Fig. 2. System Architecture when web content is not available

Figure 2 shows the general architecture of our system when web page content is not available and clustering of web pages based on content coherence for mission identification is not possible. This is the case for the VIVIDESK$^{(TM)}$ data.

2.3 Navigational Pattern Discovery

According to how missions are identified, we propose two ways to discover navigational patterns from discovered missions. If missions are identified based on content coherence, we could therefore discover content coherent navigational patterns which are sets of web pages that are frequently visited together and that have related content. These patterns are used by the recommender system to recommend web pages, if they were not already visited. To discover these navigational patterns, we simply group the missions we uncovered from the web server logs into clusters of sub-sessions having commonly visited pages. Each of the resulting clusters could be viewed as a user's navigation pattern. In this scenario, the patterns discovered from missions possess two characteristics: usage cohesive and content coherent. Usage cohesiveness means the pages in a cluster tend to be visited together, while content coherence means pages in a cluster tend to be related to a topic or concept. This is because missions are grouped according to content information. Since each cluster is related to a topic, and each page has been represented in a keyword vector, we are able to easily compute the topic vector of each cluster, in which the value of a keyword is the average of the corresponding values of all pages in the cluster. The cluster topic is widely used in our system, in both the off-line and on-line phases (see below for details). In the case where we discover missions in the absence of textual content, the navigational patterns discovered hold only usage cohesion characteristic and do not guarantee content coherence. Thus, no cluster topic vector is computed.

The clustering algorithm we adopt for grouping missions is *PageGather* [19]. This algorithm is a soft clustering approach allowing overlap of clusters. Instead of trying to partition the entire space of items, it attempts to identify a small number of high quality clusters based on the *clique* clustering technique [19].

The algorithm could be briefly summarized as follows:

- For each pair of web pages *P1* and *P2* in the visit missions (collectively), we compute $P(P1|P2)$, the probability of a visitor visiting *P1* after already visiting *P2* and $P(P2|P1)$, the probability of a visitor visiting *P2* after already visiting *P1*. The minimum of these two values is recorded as the co-occurrence frequency between *P1* and *P2*.
- We create a similarity matrix between web pages where the distance (similarity) between pages is either zero if the two pages are directly linked in the web site structure (i.e. there is a hyperlink from one to the other) or set to the co-occurrence frequency between the two pages otherwise.
- We create a graph G in which each page is a node and each nonzero cell in the matrix is an arc.
- In order to reduce noise, we apply a threshold to remove edges corresponding to low co-occurrence frequency.
- In this graph G, a cluster corresponds to a set of nodes whose numbers are directly connected with arcs. A clique – a subgraph in which every pair of nodes has an edge between them – is a cluster in which every pair of pages co-occurs often.

2.4 Navigational Pattern Improved with Connectivity

The missions we extracted and clustered to generate navigational patterns are primarily based on the sessions from the web server logs. These sessions exclusively represent web pages or resources that were visited. It is conceivable that there are other resources not yet visited, even though they are relevant and could be interesting to have in the recommendation list. Such resources could be, for instance, newly added web pages or pages that have links to them not evidently presented due to bad design. Thus, these pages or resources are never presented in the missions previously discovered. Since the navigational patterns, represented by the clusters of pages in the missions, are used by the recommendation engine, we need to provide an opportunity for these rarely visited or newly added pages to be included in the clusters. Otherwise, they would never be recommended. To alleviate this problem, our general system model expands the clusters to include the connected neighbourhood of every page in a mission cluster. The local neighborhood of a page, obtained by tracing a small number of links from the originating page, is a good approximation to the "semantic neighborhood" of the page [13]. In our case, the connected neighbourhood of a page p is the set of all the pages directly linked from p and having similar content of p, and all the pages that directly link to p also with similar content. Figure 3(A) illustrates the concept of neighbourhood expansion, and Figure 3 (B) shows the process of the augmentation. The cluster expansion is only possible when content and structure channels are available. In detail, this approach of expanding the

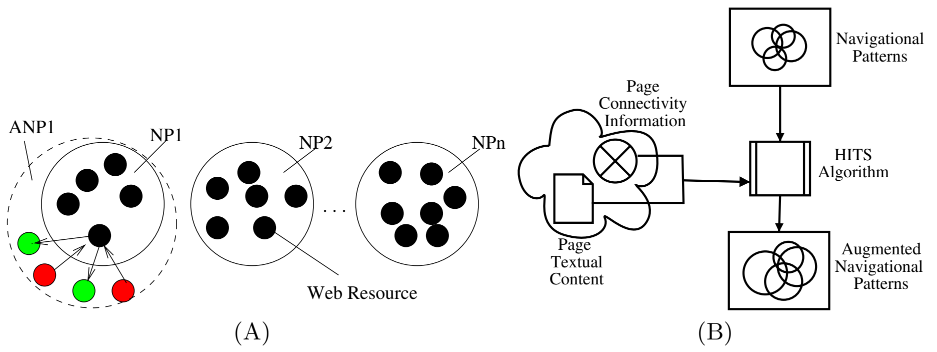

Fig. 3. (A) Navigational Patterns(NPs) and Augmented Navigational Patterns(ANPs) (B) The Augmentation Process

neighbourhood is performed as follows: we consider each previously discovered navigational pattern (i.e., a cluster of content coherent and visitation cohesive missions) as a set of seeds. Each seed is supplemented with pages it links to and pages that link to it as well as having similar content. The result is what is called a connectivity graph which now represents our augmented navigational pattern. This process of obtaining the connectivity graph is similar to the process used by the HITS algorithm [10] to find the *authority* and *hub* pages for a given topic. The difference is that we do not consider a given topic, but start from a mission cluster as our set of seeds. Moreover, it was shown in [1] that HITS, using pure connectivity analysis, introduces a problem known as "topic drift". We eliminate this problem in our case by computing relevance weights of all supplemented pages. The relevance weight of a page equals the similarity of the page content to the corresponding mission cluster, which is represented by the cosine normalization of web pages and mission clusters keyword vectors. We then prune nodes whose relevance weights are below a threshold from the connectivity graph. For simplicity, we use *Median Weight* (i.e. the median of all relevance weights) as the pruning threshold [1]. The pruning process avoids augmenting the navigational patterns with pages that focus on other topics and guarantees that the augmented patterns are still coherent and focused. After expanding and pruning the clusters representing the navigational patterns, we also augment the keyword vectors that label the clusters. The new keyword vectors that represent the augmented navigational patterns have also the terms extracted from the content of the augmented pages.

We take advantage of the built connectivity graph by cluster to apply the HITS algorithm in order to identify the *authority* and *hub* pages within a given cluster. These measures of *authority* and *hub* allow us to rank the pages within the cluster. This is important because at real time during the recommendation, it is crucial to rank recommendations, especially if they are numerous. Long recommendation lists are not advisable.

Authority and *hub* are mutually reinforcing [10] concepts. Indeed, a good *authority* is a page pointed to by many good *hub* pages, and a good *hub* is a page that points to many good *authority* pages. Since we would like to be able to recommend pages newly added to the site, in our framework, we consider only the *hub* measure. This is because a newly added page would be unlikely to be a good authoritative page, since not many pages are linked to it. However, a good new page would probably link to many *authority* pages; it would, therefore, have the chance to be a good *hub* page. Consequently, we use the *hub* value to rank the candidate recommendation pages in the on-line module.

Some may argue to use the content similarity (if applicable) to rank the candidate recommendations. However, the success of Google (www.google.com) encourages us to embed web structure analysis into this task. In Google, *PageRank* [2] – a pure web linkage analysis algorithm – is combined with the textual content information of web pages to provide search results. In general, when a user submits a query, Google searches all pages containing the keyword(s) in the query. The resulting pages are ranked according to their *PageRank* scores, which have been pre-computed. The higher its *PageRank* value, the earlier a page is presented to the user. Traditionally, a search engine can be viewed as an application of information retrieval with the focus on "matching": a search engine is supposed to return all those pages that match users' query, ranked by degree of match. On the other hand, the semantics of a recommender system is "interesting and useful" [3]. However, Google blurs this distinction by incorporating *PageRank* into its ranking, which uses web structure information to measure the authoritativeness or importance of web pages. From this point, Google can be viewed as a form of hybrid recommender system combining content and structure analysis with a one-input interface (By contrast, regular recommender systems have a zero-input interface). Indeed, this linkage analysis could compensate the possible limitation of our content coherent mission identification to web pages that are related by the virtue of their functionality rather than content. Ranking recommendation candidates based on this connectivity analysis could also allow rarely visited or newly added pages to be include in recommendations. This is important because rarely visited pages and newly added pages do not have corresponding entries in the web server access log and are typically excluded from any potential recommendation. Our approach gives them the chance to be picked up as recommended pages.

2.5 The Recommendation Engine

The previously described process consists of pre-processing done exclusively off-line. When a visitor starts a new session in the web site, we identify the navigation pattern after a few clicks and try to match on-the-fly with already captured navigational patterns. If they were matched, we recommend the most relevant pages in the matched cluster. When page content is not obtainable, the available clusters are based solely on access history, and we identify navigational patterns by finding the clusters that contain the last page referenced in the current user's mission. However, in the presence of content, identifying the navigational pattern of the current visitor consists of recognizing the current focused topic of interest to the user. A study in [4] shows that looking on either side of an anchor (i.e., text encapsulated in a *href* tag) for a window of 50 bytes would capture the topic of the linked pages. Based on this study, we consider the anchor clicked by the current user and its neighbourhood on either side as the contextual topic of interest. The captured topics are also represented by a keyword vector which is matched with the keyword vectors of the clusters representing the augmented navigational patterns. From the best match, we get the pages with the best *hub* value and provide them in a recommendation list, ranked by the *hub* values. The *hub* value is chosen for ranking instead of the *authority* value because the *authority* value does not favor newly added pages and disadvantages them. Indeed, newly added pages are not linked from other pages since they were unknown and thus would never have a high value of *authority*. However, newly added pages could certainly link to good authorities.

To avoid supplying a very large list of recommendations, the number of recommendations is adjusted according to the number of links in the current page: we simply make this number proportional to the number of links in the current page. Our goal is to have a different recommendation strategy for different pages based on how many links the page already contains. Our general strategy is to give $\sqrt{n}$ best recommendations (n is the number of links), with a maximum of 10. The limit of 10 is to prevent adding noise and providing too many options. The relevance and importance of recommendations is measured with the *hub* value already computed off-line.

3 Experimental Evaluation

We evaluate our recommendation framework on both a generic web site dataset with all three information channels available (the University of Alberta Department of Computing Science web server, abbreviated as the UofA CS web server) and an application-specific enriched log with only the usage channel available (VIVIDESK$^{(TM)}$ session-based logs). For the UofA CS web server access logs, data were collected for 8 months (Sept. 2002 – Apr. 2003), and partitioned into months. On average, each monthly partition contains more than 40,000 pages, resulting in on average 150,000 links between them. The log of each month averaged more than 200,000 visit sessions, which generated an average of 800,000 missions per month. The modified DC-tree content clustering algorithm generated about 1500 content clusters, which we used to identify the missions per month. For VIVIDESK$^{(TM)}$ logs, data were collected for one and a half years (May 2001 – Sept. 2002), totaling 16024 login sessions. Data are also partitioned into months.

3.1 Methodology

Given the data partitioned per month as described above, we adopt the following empirical evaluation: one or more months data is used for building our models (i.e., training the recommender system), and the following month or months for evaluation. The reason why we divide the data based on a time frame (months) rather than use standard cross-validation on the data set is that we want to measure the prediction ability of our system for the future rather than merely the past. Moreover, the web site evolves over time. More specifically, the idea is that given a session s from a month m, if the recommender system, based on data from month $m-1$ and some prefix of the session s, can recommend pages p_i that contain some of the pages in the suffix of s, then the recommendation is considered accurate. Moreover, the distance in the number of clicks between the suffix of s and the recommended page p_i is considered a gain (i.e., a shortcut). More precisely, we measure the *Recommendation Accuracy* and the *Shortcut Gain* as described below.

Recommendation Accuracy is the ratio of correct recommendations among all recommendations, and the correct recommendation is the one that appears in the suffix of a session from which the prefix triggers the recommendation. As an example, consider that we have S visit sessions in the test log. For each visit session s, we take each page p and generate a recommendation list *R(p)*. *R(p)* is then compared with the remaining portion of s (i.e., the suffix of s). We denote this portion *T(p)* (T stands for Tail). The recommendation accuracy for a given session would be how often *T(p)* and *R(p)* intersect. The general formula for *recommendation accuracy* is defined as:

$$\textit{Recommendation Accuracy} = \frac{\sum_s \frac{\left|\bigcup_p (T(p) \bigcap R(p))\right|}{\left|\bigcup_p R(p)\right|}}{S}$$

The *Shortcut Gain* measures how many clicks the recommendation allows users to save if the recommendation is followed. Suppose we have a session a, b, c, d, e, and at page b, the system recommends page e; then if we follow this advice, we would save two hops (i.e., pages c and d). There is an issue in measuring this shortcut gain when the recommendation list contains more than one page in the suffix of the session. Should we consider the shortest gain or the longest gain? To solve this problem, we opted to distinguish between *key* pages and *auxiliary* pages. A *key* page is a page that may contain relevant information and in which a user may spend some time. An *auxiliary* page is an intermediary page used for linkage and in which a user would spend a relatively short time. In our experiment, we use a threshold of 30 seconds as this distinction. Given these two types of pages, a shortcut gain is measured as being the smallest jump gain towards a *key* page that has been recommended. If no *key* page is recommended, then it is the longest jump towards an *auxiliary* page. The set of pages in the session we go through with the assistance of the recommender system is called the improved session *s'*. For the total S visit sessions in the test log, *Shortcut Gain* can be computed as:

$$\mathit{Shortcut\ Gain} = \frac{\sum_s \frac{|s| - |s'|}{|s|}}{S}$$

In addition, we compute the *Coverage* of a recommender system, which measures the ability of a system to produce all pages that are likely to be visited by users. The concept is similar to what is called *Recall* in information retrieval. *Coverage* is defined as:

$$\mathit{Recommendation\ Coverage} = \frac{\sum_s \frac{\left|\bigcup_p (T(p) \bigcap R(p))\right|}{\left|\bigcup_p T(p)\right|}}{S}$$

3.2 Experimental Results

Experiments on the UofA CS Web Server Dataset
We first evaluated the performance of our system on the UofA CS web server dataset. Our first experiment varies the *Coverage* to see the tendency of the *Recommendation Accuracy*, as depicted in Figure 4(A). For the purpose of comparison, we also implement an Association Rule Recommender System, the most commonly used approach for web mining based recommender systems, and record its performance in the same figure. As expected, the accuracy decreases when the we increase coverage. However, our system was consistently superior to the *Association Rule* system by at least 30%.

We next varied the *coverage* to test the *Shortcut Gain*, both with our system and with the *Association Rule* System, as illustrated in Figure 4(B).

From Figure 4(B), we can see that in the low boundary where the *Coverage* is lower than 8%, the *Shortcut Gain* of our system is close to that of the *AR* system. With the increase of the *Coverage*, however, our system can achieve an

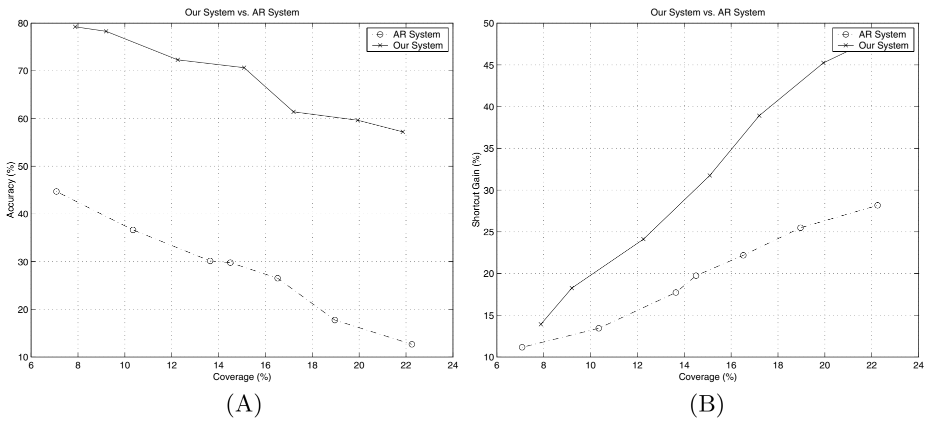

Fig. 4. Performance Comparison: our system vs. *Association Rule* Recommender System. (A): *Recommendation Accuracy* (B): *Shortcut Gain.*

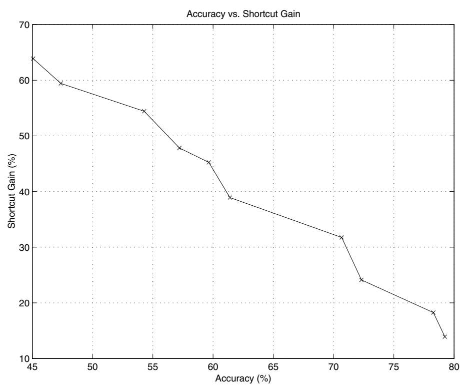

Fig. 5. *Accuracy* vs. *Shortcut Gain*

increasingly superior *Shortcut Gain* than the latter, although the performance of both systems continues to improve.

Figure 5 depicts the relationship of *Recommendation Accuracy* and *Shortcut Gain* in our system. It shows that *Recommendation Accuracy* is inversely proportional to the *Shortcut Gain.* Our study draws the same conclusion from the *Association Rule* recommender system. We argue this is an important property of a usage-based web recommender system, and therefore, how to adjust and balance between the *Accuracy* and *Shortcut Gain* for a web recommender system to achieve the maximum benefit is a question that should be investigated. Some web sites, e.g., those with high link density, may favour a recommender system with high *Accuracy*, while some others may favor a system with high *Shortcut Gain.*

In the above tests, the three distinctive information channels – usage, content, and structure – are provided to, and used in our system. In a second battery of tests we measured the effect of the individual information channels. We first compared three recommender systems, one using all channels, one using only usage and one using only content. We refer to our recommender using the three channels as *Hybrid123*. For this comparison, we implemented an association rule-based usage recommender system as in the previous tests (referred to as *Usage*), as well as a web recommender system based purely on content similarity (referred to as *Content*). The *Usage* system works as follows: an efficient association rule algorithm [5] is applied to the access logs to generate a set of rules. Whenever the pages in the antecedent of an rule have appeared in the user's current session, those pages in its consequence are recommended. For the *Content* system, all pages in the web site are extracted and grouped into clusters solely based on their textual content similarity, using a high-quality content clustering algorithm [18]. If one or more pages in a cluster have been visited, the pages in the same clusters are selected to be recommended. The *Recommendation Accuracy* and *Shortcut Gain* of the three systems are depicted in Figure 6. In the experiment, we varied the *Coverage* to test the trend and consistency of the system quality.

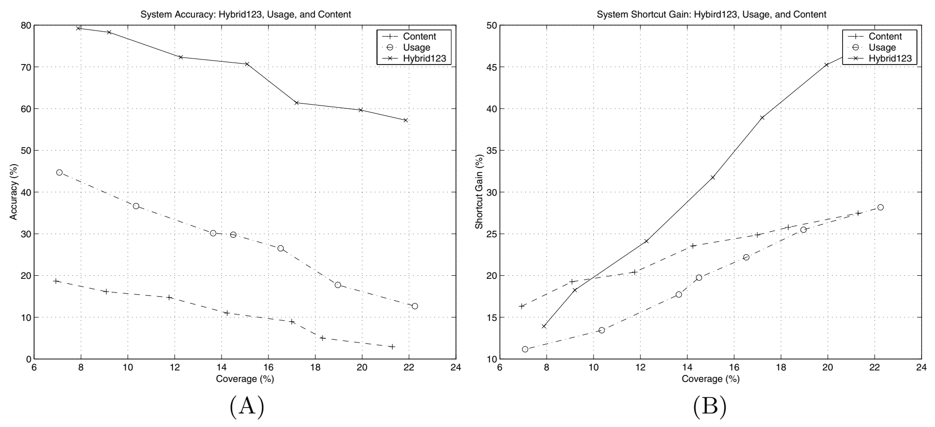

Fig. 6. *Hybrid123*, *Usage*, and *Content*. (A): *Recommendation Accuracy* (B): *Shortcut Gain*.

Figure 6 (A) shows the *Recommendation Accuracy* of the three contenders. As expected, the accuracy decreases when we increase *Coverage*. However, *Hybrid123* is consistently the best among the three systems, superior to *Usage* by at least 30% – while *Usage* always ranks second.

From Figure 6 (B), we can see that in the low boundary, the *Shortcut Gain* of *Content* is the best of the three systems, and the other two are close. With the increase of *Coverage*, the *Shortcut Gain* of all three systems continues to improve, but in different degrees. *Hybrid123* can achieve an increasingly superior *Shortcut Gain* to that of *Usage*, and exceeds *Content* after *Coverage* is larger than about 10%. The major reason that the *Shortcut Gain* improvement of *Content* is lowest is that with the increase of *Coverage*, more and more pages containing only the same terms, but without any logical relationship are selected to be recommended.

In our next experiment, we illustrate the advantage of incorporating web content and web structure information in our system. To do so, we implemented additional two recommender prototypes. The first is similar to *Hybrid123* but is stripped from its connectivity information channel. That is, we do not make use of linkage information to augment and improve the navigational patterns built on usage and content information. We name this hybrid system *Hybrid-3.* The second is also a similar system to *Hybrid123* but does not make use of content information to identify a mission. Rather, the navigational patterns in the system is built upon traditional transactions identified according to the approach in [7]. Then, the patterns are improved with structure information, as with *Hybrid123.* This hybrid system is called *Hybrid-2.* The *Recommendation Accuracy* and *Shortcut Gain* of the three systems are depicted in Figure 7.

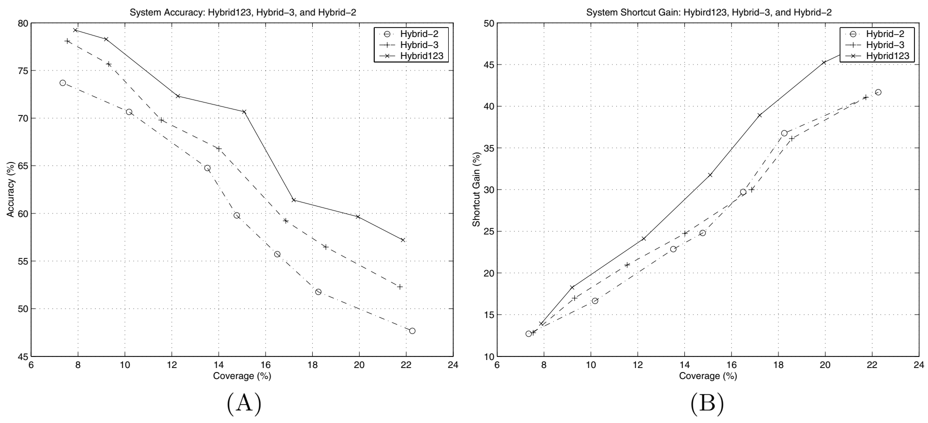

Fig. 7. *Hybrid123*, *Hybrid-3*, and *Hybrid-2.* (A): *Recommendation Accuracy* (B): *Shortcut Gain.*

Figure 7 (A) shows the *Recommendation Accuracy* of the three systems. The consistent best performance of *Hybrid123* illustrates the validity of content and connectivity information to improve recommendations in our hybrid system, and also indicates that content is more useful for recommendation accuracy improvement. The *Shortcut Gains* of the three systems are depicted in Figure 7 (B). We notice that with the increase of *Coverage*, *Hybrid123* can achieve an increasingly superior *Shortcut Gain* compared to both *Hybrid-3* and *Hybrid-2*, while the two systems keep similar performance in terms of *Shortcut Gain.* This figure verifies our justification for using distinctive information channels in building a hybrid recommender system, and shows that content and structure information make a similar contribution to the improvement in *Shortcut Gain* in our system.

In summary, this experiment shows that our system can significantly improve the quality of web site recommendation by combining the three information channels, while each channel included contributes to this improvement.

Experiments on the VIVIDESK Log

We then tested our system on VIVIDESK$^{(TM)}$ log data. As explained before, the visited page content information is not available and we used a more general definition of *mission*, namely the applications used during a VIVIDESK$^{(TM)}$ session. However, VIVIDESK$^{(TM)}$ also records in its logs keystrokes made by users. These text data, while not the real content of pages, can be associated with the visited resources and used to separate sessions into missions. Thus, we implemented two recommender systems: one using the simple definition of mission by means of the applications (*App-Mission*), and one using the extra text data to generate missions (*Text-Mission*). In addition, we implemented the same system but using transactions as defined in [7] (*Tran*) to verify the advantage of missions over transactions. In our reported experiment, we also varied the *Coverage* to see the tendency of the *Recommendation Accuracy* and *Shortcut Gain*.

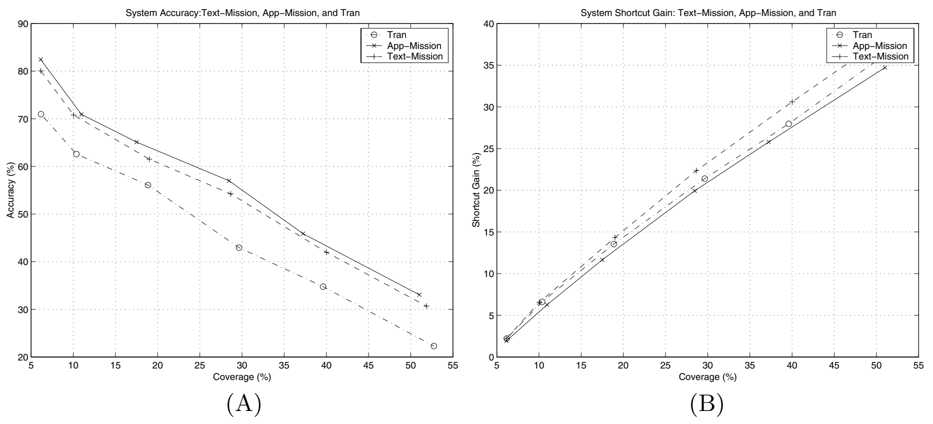

Fig. 8. *Text-Mission*, *App-Mission*, and *Tran*. (A): *Recommendation Accuracy* (B): *Shortcut Gain*.

As depicted in Figure 8, we notice that *App-Mission* could achieve a higher *Recommendation Accuracy* than simple transaction identification, but lead to a lower *Shortcut Gain*. However, because we can get a much higher *Recommendation Accuracy* with a slight loss of *Shortcut Gain*, we can be confident that mission identification is a better model for user navigational behaviour. The reason why *App-Mission* lead to a lower *Shortcut Gain* is that we identify missions solely based on invoked applications with the absence of content. However, users may need more than one application to fulfill one information need. Thus, identifying missions based on applications may break some interrelationship between web resources across applications. However, bigger jumps are achieved when we used the text entered by the users as means to identify missions. This text is the text entered for instance in HTML form input fields. The *Shortcut Gain* achieved by *Text-Mission* is even higher than the transaction-based approach. This justifies our advocacy of using additional information channels for recommendation improvement.

4 Conclusion

In this paper, we present a framework for a combined web recommender system, in which users' navigational patterns are automatically learned from web usage data and content data. These navigational patterns are then used to generate recommendations based on a user's current status. The items in a recommendation list are ranked according to their importance, which is in turn computed based on web structure information. Our preliminary experiments show that the combination of usage, content, and structure of data in a web recommender system has the potential to improve the quality of the system, as well as to keep the recommendation up-to-date. However, there are various ways to combine these different channels. Our future work in this area will include investigating different methods of combination.

References

1. K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 104–111, Melbourne, AU, 1998.
2. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *The 7th World-Wide Web Conference*, 1998.
3. R. Burke. Hybrid recommender systems: Survey and experiments. In *User Modeling and User-Adapted Interaction*, 2002.
4. S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference*, 1998.
5. M.-S. Chen, J.-S. Park, and P. Yu. Efficient data mining for path traversal patterns. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):209–221, 1998.
6. E. H. Chi, A. rosien, and J. Heer. Lumberjack: Intelligent discovery and analysis of web user traffic composition. In *The Fourth International WEBKDD Workshop: Web Mining for Usage Patterns and User Profiles*, pages 1–15, 2002.
7. R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.
8. X. Fu, J. Budzik, and K. J. Hammond. Mining navigation history for recommendation. In *Intelligent User Interfaces*, pages 106–112, 2000.
9. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230 – 237, 1999.
10. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
11. J. Li and O. R. Zaïane. Combining usage, content, and structure data to improve web site recommendation. In *5th International Conference on Electronic Commerce and Web Technologies (EC-Web 2004)*, 2004.
12. J. Li and O. R. Zaïane. Using distinct information channels for mission-based web recommender system. In *Sixth ACM SIGKDD Workshop on Webmining and Web Analysis (WebKDD 2004)*, pages 35–46, Seattle, WA, USA, August 2004.

13. H. Lieberman. Autonomous interface agents. In *Proceedings of the ACM Conference on Computers and Human Interface, CHI-97*, Atlanta, Georgia, 1997.
14. C. Lin, S. Alvarez, and C. Ruiz. Collaborative recommendation via adaptive association rule mining, 2000.
15. B. Mobasher, H. Dai, T. Luo, Y. Sun, and J. Zhu. Integrating web usage and content mining for more effective personalization. In *EC-Web*, pages 165–176, 2000.
16. M. Nakagawa and B. Mobasher. A hybrid web personalization model based on site connectivity. In *Fifth WebKDD Workshop*, pages 59–70, 2003.
17. O. Nasraoui, H. Frigui, R. Krishnapuram, and A. Joshi. Extracting web user profiles using relational competitive fuzzy clustering. *International Joint Artificial Intelligence Tools*, 9(4), 2000.
18. P. Pantel and D. Lin. Document clustering with committees. In *The 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.
19. M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically synthesizing web pages. In *AAAI/IAAI*, pages 727–732, 1998.
20. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167, 2000.
21. U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.
22. J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.
23. W. Wong and A. Fu. Incremental document clustering for web page classification, 2000.
24. A. L. C. Yi-Hung Wu, Yong-Chuan Chen. Enabling personalized recommendation on the web based on user interests and behaviors. In *11th International Workshop on research Issues in Data Engineering*, 2001.