On-line Generation of Suggestions for Web Users

Fabrizio Silvestri
Istituto ISTI - CNR
Pisa – Italy

Ranieri Baraglia
Istituto ISTI - CNR
Pisa – Italy

Paolo Palmerini
Istituto ISTI - CNR
Pisa - Italy

{fabrizio.silvestri,ranieri.baraglia,paolo.palmerini}@isti.cnr.it

Abstract

One important class of Data Mining applications is the so-called "*Web Mining*" that analyzes and extracts important and non-trivial knowledge from Web related data. Typical applications of Web Mining are represented by the personalization or recommender systems. These systems are aimed to extract knowledge from the analysis of historical information of a web server in order to improve the web site expressiveness in terms of readability and content availability. Typically, these systems are made up of two components. One, that is usually executed off-line with respect to the Web server normal operations, analyzes the server access logs in order to find a suitable categorization of users, the other, that is usually executed on-line with respect to the Web server normal operations, classifies the active requests according to the previous off-line analysis. In this paper we propose *SUGGEST 2.0* a recommender system that, differently from those proposed so far, does not make use of any off-line component. Moreover, in the last part of the paper, we analyze the quality of the suggestions generated and the performance of our solution. To this purpose we also introduce a novel quality metric that tries to estimate the effectiveness of a recommender system as the capacity to anticipate user requests that could be issued farther in the future.

Introduction

Web Mining is one of the most important classes of Data Mining (DM) applications Etizioni O. (1996). The World Wide Web: quagmire or gold mine? Communication of the ACM. 33:65-68, November 1996.. The main concern of Web Mining is to discover information "hidden" into Web-related data. More specifically, *Web Usage Mining* (WUM) is the process of extracting knowledge from Web users access data (also called *clikstreams*) by exploiting DM technologies. A typical application of WUM is to improve Web sites usability. This goal may be reached in several ways. The one we follow in this work is typically approached by the so-called *recommender systems* . In this kind of systems the focus is to give Web site users a list of links to pages (i.e. suggestions) that may be considered important for that user. These information are usually obtained by the prior knowledge extracted from the usage history of the site itself. Moreover, it should be desirable to maintain the knowledge base updated with information coming from new requests issued by the Web site users. This last point is rarely reached in current WUM systems. In fact, to the best of our knowledge the system we describe here is the only one that is able to reach this goal. *SUGGEST 2.0*, in fact, operates in an on-line fashion with respect to the Web server activity.

A recommender system is usually structured in two stages Baraglia, R., & Palmerini P. (2002). Suggest: A Web Usage Mining System. Proceedings of the IEEE International Conference on Information Technology: Coding and Computing (ITCC 2002)., Mobasher B., Cooley B., & Srivastava J. (2000). Automatic Personalization Based on Web Usage Mining. Communications of the ACM, 43(8):142–151., Mobasher B., Jain N., Han E.H., & Srivastava J. (1996). Web Mining: Pattern Discovery from World Wide Web Transactions. TR 96-050, University of Minnesota., Silvestri F., Baraglia R., Palmerini P., & Serranò M. (2004). On-line Generation of Suggestions for Web Users. Proceedings of the IEEE International Conference on Information Technology: Coding and Computing (ITCC 2004)., and Yan T.W., Jacobsen M., Garcia-Molina H., & Umeshwar D. (1996). From User Access Patterns to Dynamic Hypertext Linking. Proceedings of the Fifth International World Wide Web Conference.. The first stage analyzes the historical data recorded in a Web server log and builds a knowledge base that will be used in the second stage to generate the personalized content.

The first stage, usually executed *off-line* with respect to the Web server normal operations, proceeds by analyzing the historical data. The main functions carried out in this first stage are: *Preprocessing*; and *Pattern Discovery*. Preprocessing is devoted to data cleaning and session identification, while Pattern Discovery is aimed to build the system knowledge base by using some DM techniques, such as association rules, sequential patterns, clustering or classification.

The second stage, which is usually executed *on-line* with respect to the normal Web server operations, makes use of the extracted knowledge base to enrich the content of a Web site in several forms, such as links to pages, advertisements, hints regarding products or services that are retained to be of interest for the current user.

In the past, several WUM projects have been proposed to foresee users preferences and their navigation behaviors, as well as many recent results improved separately the quality of the personalization or the user profiling phase Frias-Martinez E., & Karamcheti V. (2003). Reduction of User Perceived Latency for a Dynamic and Personalized Site Using Web-Mining Techniques. Proceedings of the WebKDD workshop, pages 47–57., Nakagawa M., & Mobasher B.. A Hybrid Web Personalization Model Based on Site Connectivity. Proceedings of the WebKDD workshop, pages 59–70., Nasraoui O., & Petenes C. (2003). Combining Web Usage Mining and Fuzzy Inference for Website Personalization. Proceedings of the WebKDD workshop, pages 37–46..

Analog Yan T.W., Jacobsen M., Garcia-Molina H., & Umeshwar D. (1996). From User Access Patterns to Dynamic Hypertext Linking. Proceedings of the Fifth International World Wide Web Conference. is one of the first WUM systems proposed. It is structured according to two main components. Past users activity, recorded in server log files, is processed off-line to form clusters of user sessions. Then the on-line component builds active users sessions that are then classified into one of the clusters found in the previous phase. The classification allows to identify pages related to those contained in the active session and to return the requested page annotated with a list of suggestions. The geometrical approach used for clustering is affected by several limitations, related to scalability and to the effectiveness of the results found. Nevertheless, the architectural solution introduced was maintained in several other more recent projects.

The *WebWatcher* system Joachims T., Freitag D., & Mitchell T. (1997). Webwatcher: A tour guide for the world wide web. Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence. is an interface agent for the World Wide Web. It accompanies a user through the pages by suggesting hyperlinks that it believes will be of interest. The system interacts with the user who can fill predefined forms with keywords to specify his interest. To suggest a hyperlink WebWatcher uses a measure of the hyperlink quality that is interpreted as the probability that a user will select that hyperlink. It is based on both keywords specified by a user and associated to each hyperlink selected, and information coming from the hypertext structure. WebWatcher is implemented as a server and operates much like a proxy.

In Mobasher B., Cooley B., & Srivastava J. (2000). Automatic Personalization Based on Web Usage Mining. Communications of the ACM, 43(8):142–151. B. Mobasher *et al.* present *WebPersonalize.* Also in this case, the system provides dynamic recommendations, as a list of hypertext links, to users. The analysis is based on anonymous usage data combined with the structure formed by the hyperlinks of the site. In order to obtain aggregate usage profiles, DM techniques such as clustering, association rules, and sequential pattern discovery are used in the preprocessing phase. Using such techniques, server logs are converted in *session clusters* (i.e. sequence of visited pages) and *pageview clusters* (i.e. set of pages with common usage characteristics). The on-line phase considers the active user sessions in order to find matches among the users activities and the discovered usage profiles. The matching entries are then used to compute a set of suggestions that will be inserted into the last requested page as a list of hypertext links. Recently the same authors proposed in [5] a hybrid personalization model that can dynamically choose between either non-sequential models (like association rules or clustering) or models where the notion of time ordering is maintained (like sequences). The decision of which model to use is based on an analysis of the site connectivity.

In Mobasher B., Cooley R., & Srivastava J. (2000). Automatic personalization based on web usage mining. Communications of the ACM, 43(8):142–151., clusters of URLs are found using the Association Rule based Hypergraphs Partitioning technique. The on-line component of the system finds the cluster that best matches a fixed width sliding window of the current active session, by also taking into account the topology of the site. This component is implemented as a set of CGI scripts that dynamically create customized pages.

In Wu K.L., Yu P.S., and Ballman A. (1998). Speedtracer: A Web Usage Mining and Analysis Tool. IBM Systems Journal, 37(1). *SpeedTracer*, a usage mining and analysis tool, is described. Its goal is to understand the surfing behavior of users. Also in this case, the exploring of the server log entries does the analysis. The main characteristic of SpeedTracer is that it does not require cookies or user registration for session identification. In fact, it uses five kind of information: IP, Timestamp, URL of the requested page, Referral, and Agent to identify user sessions. The application uses innovative inference algorithms to reconstruct user traversal paths and identify user sessions. Advanced mining algorithms uncover users movements through a Web site. The final result is a collection of valuable browsing patterns that help webmasters to better understand user behavior. SpeedTracer generates three types of statistics: user-based, path-based and group-based. User-based statistics pinpoint reference counts by user and durations of access. Path-based statistics identify frequent traversal paths in Web presentations. Group-based statistics provide information on groups of Web site pages most frequently visited.

PageGather Perkowitz M., & Etzioni O. (1999). Adaptive Web Sites: Conceptual Cluster Mining. Proceedings of the International Joint Conference on Artificial Intelligence, pages 264–269. is a personalization system concerned with the creation of index pages containing topic-focused links. Using automatically extracted users access patterns enhances the site organization and presentation. The system takes as input a Web server log and a conceptual description of each page at a site and it uses clustering to discover pages that are visited together. To perform this task a co-occurrence matrix M is built where each element M_{ij} is defined as the conditional probability that page i is visited, given that page j has been visited in the same session. A threshold for M_{ij} allows to prune those entries regarding pages having a low probability of being accessed together. The directed acyclic graph associated with M is then partitioned to find the graph cliques. Finally, cliques are merged to originate the clusters. This solution introduced the hypotheses that users behave coherently during their navigation, i.e. pages within the same session are in general conceptually related. This assumption is called visit coherence. The generated static index pages are kept in a separate "Suggestion Section" of the site.

As we have observed above, in most of the previous proposed systems a two-tier architecture is used. The main limitation of this approach is the loosely coupled integration of the WUM system with the Web server ordinary activity. Indeed, the use of two components leads to the disadvantage of having an "asynchronous cooperation" between the components themselves. The off-line component has to be periodically performed in order to keep the patterns up-to-dated, but the frequency of the updates is a problem that has to be solved on a case specific basis. To overcome this problem, in this work we propose *SUGGEST 2.0*: a WUM system whose main difference with respect to other solutions, is the fact that it is composed of only one on-line component that is able to update incrementally and automatically the knowledge obtained from historical data, and to generate on-line personalized content. Moreover, we used a novel effectiveness measure based on the assumption that a good recommender system should suggest pages to users, in order to reduce the average length of a session.

It should be noted that the integration of the functionalities of the off-line and on-line components into a single one, poses other problems in terms of overall system performance, and response time. The system, in fact, should be able to generate personalization without too much overhead, and the knowledge mined by the single component has to be comparable with that mined by using two distinct (on-line/off-line) components. It must be said that this work is, actually, an extension of a previously published paper Silvestri F., Baraglia R., Palmerini P., & Serranò M. (2004). On-line Generation of Suggestions for Web Users. Proceedings of the IEEE International Conference on Information Technology: Coding and Computing (ITCC 2004)..

The rest of the paper is organized as follows. In Section The SUGGEST 2.0 system the overall architecture of our system is presented along with a complete description of the algorithms proposed and the results of the theoretical analysis we made on them. Section SUGGEST 2.0 Evaluation shows the results of the experiments conducted to evaluate the systems. Finally Section Conclusions concludes this paper and describes some future work we intend to do to further refine the system proposed.

i.The SUGGEST 2.0 system

The main goal of a recommender system is to provide users with useful suggestions about pages they may find of their interest.
The first version of the system, *SUGGEST 1.0*, was organized as a two-tier system composed by an offline module which carried out the first stage and an online classification module which carried out the second stage. In the new version, hereinafter referred to as *SUGGEST 2.0*, we merged the two modules into a single one that performs exactly the same operations but in a complete on-line fashion. *SUGGEST 2.0* is implemented as a module of the Apache Thau R. (1996). Design Considerations for the Apache Server API. Computer Networks and ISDN Systems, 28(7–11): 1113-1122. Web server, allowing an easy deployment on potentially any kind of Web site currently up and running, without any modification of the site itself. Since *SUGGEST 2.0* is required to work completely on-line with respect to the stream of user requests, we had to completely re-engineer the previous system. In addition we developed a novel on-line graph-partitioning algorithm that we used to incrementally adjust the clustering structure used as the system knowledge base.
Schematically, *SUGGEST 2.0* works as follows: once a new request arrives at the server, the underlying knowledge base is updated, and a list of suggestions is appended to the requested page. Collapsing the off-line and on-line modules into a single one pushed aside the asynchronous cooperation problem, i.e. the need to estimate the update frequency of the knowledge base. In **Algorithm 1** the steps carried out by *SUGGEST 2.0* are presented. At each step *SUGGEST 2.0* identifies the URL *u* requested and the session to which the user belongs. By using the session identifier it then retrieves the identifier of the URL *v* the user is coming from. According to the current session characteristics the knowledge base is updated, and the suggestions are generated. The session identifiers, along with the identifiers of the pages accessed within the sessions themselves, are stored into a simple mapping array. The session identifier is used as key to access such array. The URLs to URLs-identifiers mapping is stored into a string trie data structure. The trie is statically built at the initialization phase during which the list of all the possible URLs managed by the system is read. This is obviously possible since *SUGGEST 2.0* is designed to work just with the statically generated pages so that the filling of the list is feasible since we can know in advance the URLs of all the pages that are in a Web site.
All the steps carried out by *SUGGEST 2.0* are based on a graph-theoretic model which represents the aggregate information about navigational sessions. To extract information about navigational patterns, our algorithm models the usage information as a complete graph $G = (V,E)$. The set V of vertices contains the identifiers of the different pages hosted on the Web server. The set of edges E is weighted using the following relation:

$$W_{ij} = N_{ij} / \max\{N_i, N_j\}$$

where N_{ij} is the number of sessions containing both pages i and j, N_i and N_j are the number of sessions containing only page i or page j, respectively. The main property that we used in devising the above formula is called *visit coherence.* This important concept, introduced in Perkowitz M., & Etzioni O. (1999). Adaptive Web Sites: Conceptual Cluster Mining. Proceedings of the International Joint Conference on Artificial Intelligence, pages 264–269., hypothesizes that users behaviour is coherent during their navigation. That means, that pages within the same session are conceptually related. The equation above respects this hypothesis and extends it by trying to discriminate internal pages from the so-called index pages. Index pages are those that, generally, do not contain useful contents and are only used as a starting point for a browsing session. Index pages are very likely to be visited with any other page and nevertheless are of little interest as potential suggestions. For instance, there are many Web pages that contain a link labeled "*home*" which makes users to return at the root of the site. Such root page should not be considered important and thus should not be suggested to any users. To reduce the weight carried on by those pages, we divided N_{ij} by the maximum occurrence of the two pages. The effect obtained is to reduce the relative importance of links involving index pages. The data structure we used to store the weights is an adjacency matrix M where each entry M_{ij} contains the value W_{ij} computed according to the above formula.

Algorithm 1. The operations performed by *SUGGEST 2.0*.

Before computing the weights W, it is necessary to identify user sessions. In *SUGGEST 1.0* we were only able to guess the current user session by applying an heuristic based on the IP addresses and time-stamps of the past user requests. The main drawbacks of this approach are due to the presence of users behind proxies of NATs (Network Address Translators). In this case, in fact, those users appear as a single one coming from the NAT (or gateway) machine. In *SUGGEST 2.0* user sessions are identified by means of cookies stored on the client side. Cookies contain the keys to identify the user sessions. Once our module has retrieved a key, it is used to access a table that contains the corresponding session id. The computational cost of such operation is thus relatively small. In fact, a hash table is used to store and retrieve the keys allowing this phase to be carried out in time $O(1)$.
As in the original formulation, *SUGGEST 2.0* finds groups of strongly correlated pages by partitioning the graph according to its connected components. The algorithm performed in this phase of *SUGGEST 2.0* is shown in **Algorithm 2**. *SUGGEST 2.0* uses a modified version of the well known incremental connected components algorithm. We start from *u* a *Depth First Search* (DFS) on the graph induced by *M* and we search for the connected component reachable from *u*. Once the component is found, we check if there are any nodes not considered in the visit. If so, a previously connected component has been split and it needs to be identified. We simply apply again the DFS, starting from one of the nodes not visited. In the worst case, when all the URLs are in the same cluster, the cost of this algorithm will be linear in the number of edges of the complete graph *G*.To reduce the contributions of poorly represented link, the incremental computation of the connected components is driven by two threshold parameters. Aim of these thresholds is to limit the number of edges to visit by:

1. filtering those W_{ij} smaller than a constant value. We called this value *minfreq*. Links (i.e. elements M_{ij} of M) whose values are smaller than *minfreq* are poorly correlated and thus not considered by the connected components algorithm;
2. considering only components of size greater than a fixed number of nodes, namely *minclustersize*. All the components having smaller than *minclustersize* nodes are discarded because considered not sufficiently significant.

In general, the incremental connected component problem can be solved using an algorithm working in $O(|V|+|E|A)$ time, where $A=\alpha(|E|, |V|)$ is the inverse of the Ackermann's function. This is the case in which we have the entire graph and we would incrementally compute the connected component by adding one edge at a time. Our case is slightly different. In fact, we do not deal only with edge addition, but also with edge deletion operations. Moreover, depending on the value chosen for *minfreq*, the number of clusters and their size will vary, inducing a variation in the number of edges considered in the clusters restructuring phase. Figure 1 shows the results of the steps of our algorithm where the parameter *minclustersize* was set equal to 4. As it can be seen, the clusters correspond to the graph connected components. Note that in the picture we showed just the edges having weight greater than zero while in the actual implementation all the edges are present, also those having weight equal to zero. However, in those cases, and also in the cases where the edges have weight smaller than *minfreq*, we cannot consider the edge to be present when we look for the graph's connected components.

Figure 1. **A clustering example using *minclustersize* = 4. Cluster 2 contains three nodes and thus it is discarded as the singletons on the lower right.**

After the clustering step, we have to construct the suitable suggestions list. This is built in a straightforward manner by finding the cluster which have the largest intersection with the *PageWindow* corresponding to the current session (see **Algorithm 3**). The final suggestions are composed by the most relevant pages in the cluster, according to the order determined by the clustering phase. The cost of this algorithm is proportional to the *PageWindow* size and thus is constant (*O(1)*).

Algorithm 2. The clustering phase: Cluster(M, L, page_id$_u$).

Algorithm 3. The suggestions building phase: Create_Suggestions(PageWindow, L, page_id$_u$).

i.*SUGGEST 2.0* Evaluation

In order to measure the effectiveness and the efficiency of our system we conducted several tests. The first measure is the most difficult to define because it implies to be able to measure how much a suggestion has been useful for the user. In Baraglia, R., & Palmerini P. (2002). Suggest: A Web Usage Mining System. Proceedings of the IEEE International Conference on Information Technology: Coding and Computing (ITCC 2002). we introduced a parameter that permits us to quantify the effectiveness - i.e. *the quality* - of the suggestions. Such measure was based on the intersection of real sessions with the corresponding set of suggestions. For every session S_i composed by n_i pages there is a set of suggestions R_i, generated by the module in response to the requests in S_i. The intersection between S_i and R_i is:

$$\omega_i^{old} = \frac{|\{p \in S_i \mid p \in R_i\}|}{n_i}$$

The main drawback of this measure was that we were not able to capture the potential impact of the suggestions on the user navigational session. For example, if a page that the user would have visited at the end of the session is instead suggested at the beginning of the session, the suggestion in this case could help the user finding a shorter way to what s/he is looking for. Therefore we extend the above expression taking into account the distance of the generated suggestions from the pages visited during the session. For every user session S_i, we split the session into two halves. The first half S_{1i} is used to generate a set of suggestions R_{1i} , the second half is used to measure the intersection with the suggestions. For every page p_k that belongs to the intersection $S_{2i} \cap R_{1i}$ and appears in position k within S_{2i}, we add a weight $f(k)$. We choose f so that more importance is given to pages visited at the end of the session.
A different form for f can be chosen. For example to have the same coverage measure as used in Nakagawa M., & Mobasher B.. A Hybrid Web Personalization Model Based on Site Connectivity. Proceedings of the WebKDD workshop, pages 59–70. it is sufficient to take $f(k)=1$, or any constant value. It is also possible does not increase linearly the pages importance by taking $f(k)=k^2$.
In conclusion, for the whole session log, the measure of the suggestions quality is given by

$$\Omega = \sum_{i=1}^{N_S} \frac{\sum_{k=1}^{n_1/2} \langle p_k \in S_{2i} \cap R_{1i} \rangle \frac{f(k)}{F}}{N_S}$$

where N_S is the number of sessions and $<expr>$ is the truth function equal to *1* if *expr* evaluates to *true*, *0* otherwise. *F* is simply a normalization factor on the weights. We choose to take *f(k)=k* assuming, in this way, that the weights assigned to pages into the considered session increase linearly when the position occupied by a page into the session *PageWindow* increases. Another important feature of this measure is that it can also evaluate the ability of SUGGEST 2.0 in finding a good way to reduce the computational load on the Web server side. In fact, $1 - \Omega$ is a measure of the number of additional accesses to pages (maybe unimportant) that a user would have done without the support given by *SUGGEST 2.0*.

Starting from real life access log files, we generated requests to an Apache server running *SUGGEST 2.0* and recorded the suggestions generated for every navigation session contained in the log files. We considered three real life access log files publicly available: NASA, USASK and BERKLEY containing the requests made to three different Web servers. We report in **Table 1** the main features of such datasets.

Dataset	**Time Window**	**N_S**
NASA	27 days	19K
USASK	180 days	10K
BERK	22 days	22K

Table 1. Access log files used to measure the quality of suggestions.

For each dataset we measured *Ω*. The results obtained are reported in **Figure 1**. In all curves, varying the *minfreq* parameter, we measure *Ω* for the suggestions generated by *SUGGEST 2.0*.We also plotted the curve relative to suggestions generated by a random suggestion generator (rnd in **Figure 1**). As it was expected, the random generator performs poorly and the intersection between a random suggestion and a real session is almost null. On the other hand, suggestions generated by *SUGGEST 2.0* show a higher quality, which, in all dataset, reaches a maximum for *minfreq=0.2*.

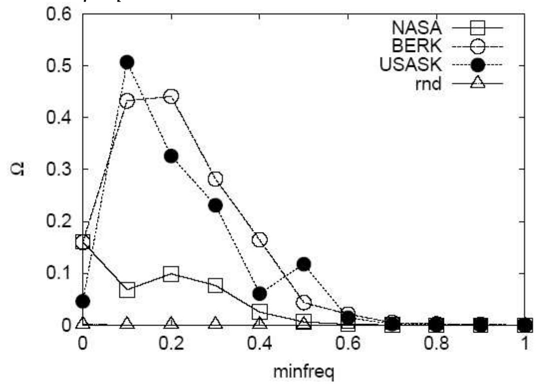

Figure 1. Suggestions coverage for the NASA, BERK, USASK access log files, varying minfreq. We also plotted the result of a random suggestions generator.

For small values of the *minfreq* parameter, good values are obtained for the quality of the suggestions, therefore a wide margin is left for further improvements. In order to measure the impact of the *SUGGEST 2.0* module on the overall performance of the HTTP server, we plotted the throughput (see **Figure 2**), i.e. number of HTPP requests served per time unit, and the total time needed to serve one single request (see **Figure 3**).

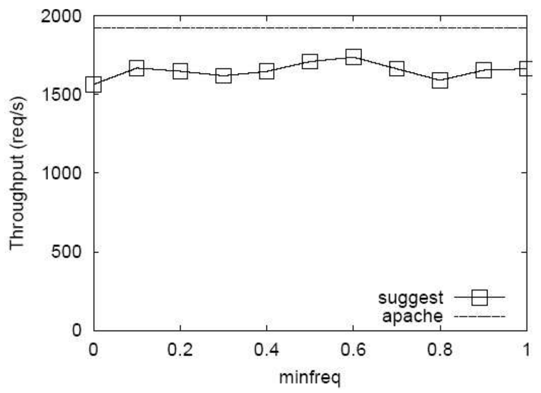

Figure 2. Impact of *SUGGEST 2.0* on the throughput of the Apache Web server.

As we can see from the **Figures 2** and **3**, the impact of *SUGGEST 2.0* on the Apache Web server is relatively limited, both in terms of throughout and in terms of the time needed to serve a single request. This limited impact on performance makes *SUGGEST 2*.0 suitable to be adopted in real life production servers. It must be noticed, however, that in its current version *SUGGEST 2.0* allocates a buffer of memory whose dimension is quadratic in the number of Web sites pages. This might be a severe limitation in sites whose number of pages can be of many thousands.

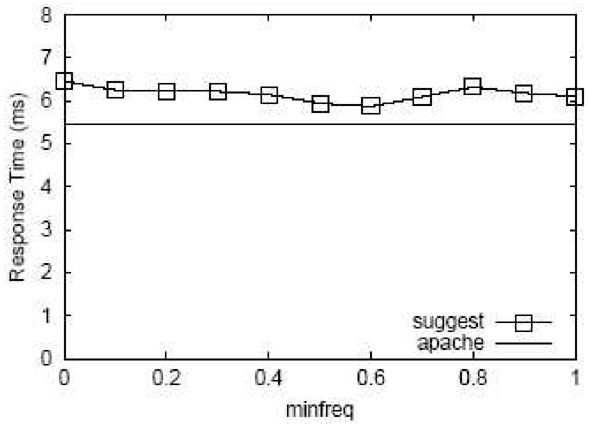

Figure 3. Apache Web server response time with and without the *SUGGEST 2.0* module.

ii.Conclusions

Web personalization can effectively be achieved with data mining techniques. An off-line extraction of knowledge from historical data, and an on-line personalization engine compose typical WUM systems. The on-line component first understands users behavior as HTTP requests arrive at the Web server, and then personalizes the content of the page requested accordingly. Such asynchronous architecture presents the problem of how to maintain updated the knowledge extracted in the off-line phase. We propose an architecture for a *WUM* system which is composed by a single on-line component, tightly integrated in the functionalities of the Apache Web server. Our system, called *SUGGEST 2.0*, processes the requests arriving at a Web server and generates a set of suggestions to Web pages correlated to the one requested. When requests arrive at the *SUGGEST 2.0* module it incrementally updates a graph representation of the Web site based on the active user sessions and classifies the active session using a graph partitioning algorithm. At the end of these operations it finally generates the suggestions, by adding at the bottom of the requested page, a set of links to potentially interesting pages. We experimentally evaluated *SUGGEST 2.0* performance by measuring the quality of its suggestions and the impact on Apache throughput and response time. To this purpose we introduced a new quality measure aimed at measuring the ability of the proposed recommender system to anticipate the requests eventually made by the users. The metric is also able to measure the ability of *SUGGEST 2.0* to reduce the computational load of the underlying HTTP server. The reduction of the load is, in fact, obtained since *SUGGEST 2.0*, in some sense, is able to give users hint on the pages which they would access quite far in the future.

We are planning to extend the work proposed in this paper in several directions. First of all we are planning to make *SUGGEST 2.0* running on a real web sites, and to validate the effectiveness of the system by checking the number of times the proposed suggestions are found interesting. This experimentations is interesting since it would measure how many users will find *SUGGEST 2.0* really useful for their needs. Moreover, we are going to extend the suggestion creation phase of *SUGGEST 2.0* by suggesting to the users the most "*interesting*" pages among those present in the cluster selected. This "*interestingness*" property should be evaluated by computing a sort of PageRank™ value of the pages in the site. To this end we think to extend the classical PageRank™ algorithm to evaluate the page relevance using both the information about the site linkage structure, and the information coming from the usage graph is induced, in our system, from the adjacency matrix *M*.

References

[1] Bamshad M., Cooley R., & Srivastava J. (2000). Persomalization Based on Web Usage Mining. Communication of the ACM Vol. 43 No. 8.

[2] Baraglia, R., & Palmerini P. (2002). Suggest: A Web Usage Mining System. Proceedings of the IEEE International Conference on Information Technology: Coding and Computing (ITCC 2002).

[3] Etizioni O. (1996). The World Wide Web: quagmire or gold mine? Communication of the ACM. 33:65-68, November 1996.

[4] Frias-Martinez E., & Karamcheti V. (2003). Reduction of User Perceived Latency for a Dynamic and Personalized Site Using Web-Mining Techniques. Proceedings of the WebKDD workshop, pages 47–57.

[5] Joachims T., Freitag D., & Mitchell T. (1997). Webwatcher: A tour guide for the world wide web. Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence.

[6] Mobasher B., Cooley B., & Srivastava J. (2000). Automatic Personalization Based on Web Usage Mining. Communications of the ACM, 43(8):142–151.
[7] Mobasher B., Jain N., Han E.H., & Srivastava J. (1996). Web Mining: Pattern Discovery from World Wide Web Transactions. TR 96-050, University of Minnesota.
[8] Mobasher B., Cooley R., & Srivastava J. (2000). Automatic personalization based on web usage mining. Communications of the ACM, 43(8):142–151.
[9] Nakagawa M., & Mobasher B.. A Hybrid Web Personalization Model Based on Site Connectivity. Proceedings of the WebKDD workshop, pages 59–70.
[10] Nasraoui O., & Petenes C. (2003). Combining Web Usage Mining and Fuzzy Inference for Website Personalization. Proceedings of the WebKDD workshop, pages 37–46.
[11] Perkowitz M., & Etzioni O. (1999). Adaptive Web Sites: Conceptual Cluster Mining. Proceedings of the International Joint Conference on Artificial Intelligence, pages 264–269.
[12] Silvestri F., Baraglia R., Palmerini P., & Serranò M. (2004). On-line Generation of Suggestions for Web Users. Proceedings of the IEEE International Conference on Information Technology: Coding and Computing (ITCC 2004).
[13] Thau R. (1996). Design Considerations for the Apache Server API. Computer Networks and ISDN Systems, 28(7–11): 1113-1122.
[14] Wu K.L., Yu P.S., and Ballman A. (1998). Speedtracer: A Web Usage Mining and Analysis Tool. IBM Systems Journal, 37(1).
[15] Yan T.W., Jacobsen M., Garcia-Molina H., & Umeshwar D. (1996). From User Access Patterns to Dynamic Hypertext Linking. Proceedings of the Fifth International World Wide Web Conference.

Acknowledgements

This research was supported by both Fondazione Cassa di Risparmio di Pisa within the project ``WebDigger: a data mining environment for the web data analysis'' and Italian Ministry of Research within the project “Technologies and Services for Enhanced Contents Delivery (Legge 449/97, 1999, settore Società dell'Informazione:)**.**