Web customer modeling for automated session prioritization on high traffic sites

Nicolás Poggi [*], Toni Moreno [†‡], Josep Lluis Berral [*], Ricard Gavaldà [§], Jordi Torres[*†]

February 10, 2007

Abstract

In the Web environment, especially for e-commerce sites, user identification is becoming a major challenge for admission control on high traffic sites. As previously demonstrated by several authors [5, 2], when a web server is overloaded there is a significant loss of throughput when we compare finished sessions and the number of responses per second; longer sessions are usually the ones ending in sales but also the most sensitive to load failures. Although there is some work on session-based admission control [2, 13], this approach does not maximize revenue as it treats all non-logged sessions the same. In this paper we present a novel method for learning to assign priorities to users with higher purchasing intentions. For this, we use traditional machine learning techniques and second order Markov-chain models; we test our approach on access logs obtained from a high-traffic online travel agency. We are able to train a system to estimate the probability of the possible user's purchasing intentions according to its early navigation clicks and other static information. The predictions can be used by admission control systems to prioritize sessions or deny them if no resources are available, thus improving sales throughput per unit of time.

[*]Computer Architecture Department, U. Politècnica de Catalunya, Barcelona, Spain
[†]Barcelona Supercomputing Center, Barcelona, Spain
[‡]Department of Management, U. Politècnica de Catalunya, Barcelona, Spain
[§]Department of Software, U. Politècnica de Catalunya, Barcelona, Spain

Keywords: Web traffic prediction, navigation patterns, machine learning, data mining, admission control, resource management, autonomic computing.

1 Introduction

During the recent years there have been important changes in the way websites work. There has been a shift from originally serving mainly static content to fully dynamic sites. Dynamic applications for the web have a huge demand on CPU power, opposed to network bandwidth that has been the traditional bottleneck of the web. Not only web programming languages have become more complex over the years, but also user requirements. Websites now rely on technologies such as AJAX to make pages more interactive, XML based languages such as SOAP to exchange B2B information among companies, and SSL for security. These technologies let sites improve user experience and privacy, but also increase the demand for CPU power [4].

Due to the rising complexity of these systems, an attractive solution is to try to make the system components able to manage themselves. This can be solved using proposals from the Autonomic Computing research area, that draw in an enormous diversity of fields within and beyond the boundaries of traditional research in computer science [6]. Our approach touches on the user modeling area, and adds some form of machine learning based on historical observations.

This trend towards more autonomic web applications has two major implications in the user modeling area. On one hand, web user modeling techniques must be enriched and adapted to use the data generated and logged by the dynamic websites, and to capture relevant features in order to properly account for the actual behavior or the user in the site. On the other hand, the increased demand for CPU processing and other resources in this kind of dynamic applications presents a scenario where user modeling can be applied in order to make a more efficient use of the available resources. This paper proposes a novel approach consisting of generating a model for web user behavior in a real, complex website and using it to support decisions regarding the allocation of the available resources, based on a revenue-related metrics. In our user models, we try to understand how to best capture the features that make a customer more likely to make a purchase, and therefore more attractive -from the point of view of maximizing revenues- to maintain in the

system even in the case of a severe overload. In this sense, we are proposing a user-adaptive policy for admission control and session prioritization.

System overload is a common situation and its incidence is growing along with the increase of complex applications that demand more and more resources. Improving the infrastructure of a website might not be simple; for cost reasons, scalability problems or because some peaks are infrequent, websites might not be able to adapt rapidly in hardware to user fluctuations. Almost no company can plan the dimension of their systems according to the resource consumption in rare peaks, and therefore overload does occur. To the analyzed travel agencys website, depending if holidays are approaching or if there is a media marketed campaign e.g. of airline companies, traffic increases substantially. These fluctuations create peak loads and user count can quintuple for a couple of days. When a server is overloaded what happens typically is that it won't server any connection as resources get locked and a race condition occurs.

To prevent serving no users, session admission control systems [5, 2] allow only a certain number of session so that the site can work for some users. In this case, firewalls and load balancers should only accept sessions if there is available capacity. Denied sessions could be forwarded to a static server with the proper explanation message to retry later. Session admission control maximizes throughput in terms of an established variable, such as the number of finished sessions or their revenue. If we take the number of sales per minute as a metric, sessions with higher purchase probability should be accepted or prioritized when the load balancer has to choose between more than one session. Websites offering premium services can also use this technique to send most profitable users to premium servers or to increase their bandwidth. Our early experiments showed that the user's intention to as site can be predicted from its early navigation clicks. In this context, a profit-based admission control policy leveraging user models can help to avoid the revenue to drop from having to delay or drop connections. Defining admission policies based on information generated from user behavior models can contribute to devising cost-effective infrastructures, and it seems to be a promising application field for user modeling.

In this paper we present a method for learning, from the analysis of session logs, how to assign priorities to customers, in this case, according to their probability of purchasing in the current session. We have gathered session logs from a high-traffic online travel agency that makes use of the above mentioned state-of-the-art web technologies. Our approach consists in using

the server log files to learn models that can be used to make predictions about each user's future behavior, with the objective of assigning a priority value to every customer based on the expected revenue that s/he will generate, which in our case essentially depends on whether s/he will make a purchase. Our proposal combines static information (time of access, URL, session ID, and whether a user is a registered customer) and dynamic information (the path followed by the user in the host's web graph). For dynamic information we use Markov-models, closely related to other models such as Customer behavior Graphs [9] and finite-state machines, among others. For static information we can use several of the traditional models in machine learning, e.g. C4.5 decision trees, Naïve-Bayes, or linear regressions.

We have developed a program that extracts the static information from session logs, and creates two second-order Markov chains, one for purchasing users and another for non-purchasing users. Each entry on the log (user action) is then passed through both Markov models and their probabilities are added as new variables to the static information. The dataset formed in this way is used to train a predictive model or predictor using the WEKA machine learning package [14], which therefore takes into account both static and dynamic information. After the predictor is built, incoming sessions can be run against the predictor, which will produce a probability on the users' purchasing intentions.

We obtain promising results in the sense that the predictions made with our system are far from random: While in real traffic only 2% of the sessions end in purchase, this percentage rises to about 16% among those sessions predicted to buy, a 8x increase. Furthermore, the learning system is adaptive, in the sense that if we further restrict the allowed number of admissible sessions over the baseline value, the precision grows accordingly. As web infrastructure capacity is limited, the probabilities produced by our system can be used by load balancer applications to prioritize sessions; the number of allowed session can be set dynamically by modifying the threshold by modifying the lower limit for the purchasing intentions.

The rest of the paper is organized as follows. Next section reviews some of the most relevant related work. Section 3 describes our approach, detailing the information contained in the log files, the methodology that we follow to generate the training data that we need in order to learn the models, and the architecture of the system. Section 4 describes the set of experiments that have been performed using our method with the available data and discusses the results obtained. Finally, Section 5 presents some conclusions

and discusses possible future work.

2 Related Work

The present work lies in the intersection between research in user modeling, machine learning, resource management and autonomic computing. Although there is extensive literature in each of the individual topics, few works address the automated learning of user models for and efficient and autonomic resource management. Recent works on web user prediction [1, 10, 8, 15] have focused on web catching or prefetching of web documents, to reduce latency of served pages and improve the cache hit rate. Another studied approach is to model users for link prediction generating navigational tours and next-link suggestions to users. The mentioned approaches are best suited for large and mostly static web pages, where users navigate vast information such as an encyclopedia. Prefetching a dynamic page that includes DB transactions might be too costly in the case of a miss or user exit. Other authors [11, 1] focus on dynamic content adaptation, where the page adapts to the type of user; it could include images, colors an even products or links. The results obtained from our work could be suited for dynamic content adaptation, however we are focusing this study on user prioritization.

Path analysis [12, 3] and Customer Behavior Model Graphs (CBMG) such as [9] are similar to our dynamic part of the approach, where we use Markov chains. Menascé et al. [9] propose to build the CBMG using k-means clustering algorithm, creating a probability matrix for the possible paths from a state. What we try to accomplish in this paper is not to predict what the next click will be, but rather to detect the user's intentions when visiting the site, and in particular whether s/he will eventually buy. Session-based admission control has been widely studied [5, 2, 13]; however, it treats all non logged users with equal priority, and we believe that it could benefit from the extra information from our predictor when peak loads occur.

The underlying vision of the approach that we are presenting in this paper is consistent with that of autonomic computing, i.e. building systems that can manage themselves according to high-level objectives from their administrators. This vision is presented in [7].

3 Our Approach

3.1 What's in a log file?

As mentioned before, we try to develop a system that could be adapted easily to different sites and servers. Therefore, we want to assume the least possible about the information that is stored for each transaction, that is, in our generic proposal we only want to use information that can be produced by most dynamic applications. The produced log should be free of static content i.e. images, CSS, javascript or other media files. It should also be cleaned of non user initiated transactions i.e. AJAX autocomplete controls, background checks and offsite requests (B2B communication).

In particular, we assume that each web transaction is a tuple like the following

[date_and_time, session_id, user_id, request_type, ip_address, extra information]

where

- date_and_time: indicate date and time of the transaction
- session_id: is some unique identifier for each session; the session should be provided from the site's dynamic application.
- user_id is some information that (often) identifies the user performing the transaction, allowing us to relate different sessions of the same user at different times. User IDs can be stored on COOKIES, if not present IP addresses can be used although this is not 100% reliable (firewalled users can share the same IP) but it is reasonable approximation.
- request_type: this would generally be the requested URL and query string, although if possible, the dynamic application would input the exact page action, as the action performed is executed after the request is made and the same URL might be used for different purposes according to the situation. Most commercial sites already have these tags for statistics and tracking purposes.
- ip_address: IP address of the requesting client, it will only be used if the user_id is not present.

- extra information: extra information could be present on the log, such as time and memory used to serve the request, type of browser originating the request, that could be useful, but will not be used for the present study.

Our preprocessor produces a transformed log file, with one output line corresponding to each input line, but with different information. In particular, information can be added by

- looking back at transactions from the same session e.g. computing how many transactions have been made before.
- looking at historical information of other sessions by the same or similar users e.g. if the user is a returning customer or if he ever actually purchased from the site before.
- and by looking forward in the session e.g. if the session ended in a sale.

Purchase information is obtained by the request_type (tag) and it is set manually in the preprocessor. At the time of prediction, one can take into account information from the past (same session or previous sessions), but not from the future. Therefore, the information that we collect by looking forward in the log can only be used in the training dataset that we prepare for learning, and the task that we are concerned with is precisely learning to predict that missing information.

3.2 Static information

In our case, for each log entry we produce a new entry in the preprocessed file of the form:

[date_range, time_range, tag, is_logged, returning_customer,
buying_customer, session_length, class]

where

- date_range is some discretization of possible days, decided by the domain expert. For example, we could have 7 values for 7 weekdays, different tags for pre_vacation, vacation, and normal times of the year, etc.

- time_range is some discretization of daytimes, such as morning, afternoon, night.

- tag belongs to a set of categories into which the original request types have been categorized, again decided by a domain expert from the request_type attribute. For us, two particular types of tags are very important:

  - the buying tag indicates that the customer is buying at this moment. Our example goal is to predict whether this session will ever contain a transaction of this type. Other goals could include: the magnitude of the transaction to be made, type of product and margins. In the future our system should be able to combine more than one goal to rate users.
  - the signed-in tag indicates that the user is logging in the system at this time, e.g. with a username and password.

- is_logged Indicates whether or not the user has already signed in the system during this session. Some websites will force the user to log in immediately. Other websites will allow the user to browse for a long time, and maybe login only when ready to buy. Therefore, the relevance can greatly vary from site to site.

- returning_customer Indicates whether this customer has ever visited the website, as indicated by a COOKIE, IP address or by the fact that s/he has logged. This variable can have different values, according to whether it has been seen in the last day, week, month, etc.

- buying_customer Indicates whether this customer has already bought in the past, as this information is considered extremely relevant to predict whether s/he will buy again and to the priority he should be assigned. Again, one could introduce different values or variables to indicate the frequency, amount, time spent since last purchase, etc.

- session_length: the length of the session. We take this to be the number of requests in the session so far. We could additionally add another variable indicating the length (in, say, seconds) of the session, but we have not done so in this version.

- class is the class assigned to this session, that is, what a correct prediction should be for this log entry. In our case, there are two classes: buyer and non-buyer. Note that this is the only information computed by looking forward in the log file, for training purposes.

3.3 Generating Dynamic Information

We call the previous information *static* because it reflects little information about the navigation path of the user in this session. On the other hand, it is reasonable to believe that the sequence of requests made by the user should help in predicting his/her future behavior. We call this sequence the *dynamic information* of the session. In dynamic websites, it can be identified by a sequence of URLs, request_type for our logs. Unfortunately, most machine learning algorithms are not well adapted to dealing with variables that are themselves sequences, and some ad-hoc mechanism has to be designed. We propose here to use Markov chains of variable order.

Recall that a k-th order Markov chain consists of a set of states S and for each state s and path p of length k defines a probability that the next state is s given that the k last visited states are those in path p. In particular, by using a chain-type rule, a Markov chain M assigns a probability $\Pr[p|M]$ to each path p of any length. If the transition probabilities of M are inferred from a set of observed paths, then one can take $\Pr[p|M]$ as an approximation of the probability of path p in the data obtained by forgetting all history before the last k steps.

More precisely, for some parameter k, we create a two order k Markov chain for each of the classes, which models the typical sequences of tags (requests) for each class. In our case, we train two models: one for buyers and one for non-buyers. At prediction time, and given the path followed in the current session, these two chains can be used to compute probabilities $\Pr[p|buyer]$ and $\Pr[p|nonbuyer]$, where p is the sequence of previous k tags in the session. Using Bayes' rule, we can then estimate the converse probabilities $\Pr[buyer|p]$ and $\Pr[nonbuyer|p]$. That is, given that the user has followed this path, the Markov chains guess the probabilities that later in the future s/he buys or doesn't buy.

The *buying* and *non-buying* probabilities can be added as new variables to the static information. The resulting sequence of transformed and enriched log entries can be treated as a dataset where the order of examples is irrelevant and each example is a tuple of simple values (numeric or categorical

values). This is what is needed to apply most machine learning algorithms in the literature, such as Naïve Bayes or Bayesian nets, decision trees, linear and nonlinear regression, and most neural network models, among others.

3.4 Architecture of our system

Figure 1 presents and summarizes the architecture of our proposal.

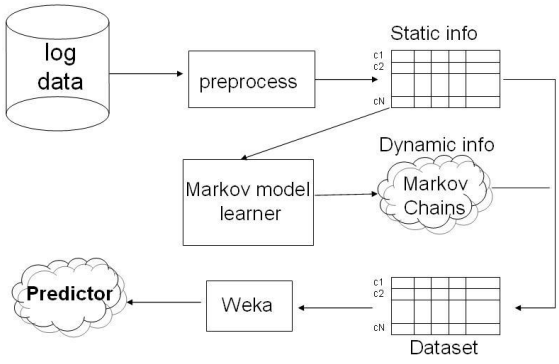

Figure 1: Architecture of our system

The preprocessor is in charge of reading the raw log file and extracting the static information. Additionally, it passes the information on the sequence of tags (or path) followed in each session to the Markov model builder, which produces one Markov chain for each class. These Markov chains are then run again on the preprocessed sessions to estimate a probability that each session belongs to one of the classes. These probabilities are then added as new variables to the transformed log file. Finally, this new log file is passed to one or more machine learning algorithms to produce a predictor model. For building predictors we have used the popular WEKA [14] package, which incorporates dozens of machine learning algorithms ready to use.

4 Experiments

4.1 The Data

The data for the experiment was provided by high traffic Spanish online travel agency. It consisted of about 122,000 transactions collected over approximately 2 days (from 06/29/2006 to 06/30/2006). The web access log was produced by the site's dynamic application; additional code was added at the end of each executing script to log the transaction data after the actions were executed. By doing so, the data is already cleaned and more accurate, as opposed to the access log from a web server. In contrast to the web server, the application can log directly the user session, not only its IP address; allowing us to separate correctly NAT/firewalled users. The additional code also logs the exact page action the user executed, which might be ambiguous from a regular URL in an access log which we call the request_type.

The data was later preprocessed to clean to remove noise from the log. We found that there were a couple of more than 50 transactions sessions, which were identified manually as being produced by automated bots and crawlers. One-click sessions were also cleaned as they would not account for more load, which is the focus of this study. One-click sessions were mainly users that entered the site from a banner of a search page, this could be verified as most request_types were made to landing pages, specially prepared for this purpose.

¿From the data we were able to distinguish 42 different tags or different "pages" accessed by users during their navigation in the site. This corresponded to 21,200 sessions, where 4783 is the number of users who returned; the number of returned users would probably been higher if the analyzed period was for more days. In the website users were not required to login to purchase and the login feature is located after selecting a product, so the login number was very low. Our result does not depend on login information, as we treat all users as being anonymous; login data is used as a complement as is readily available but not guaranteed.

An important feature of the data is that only about 2% of the sessions end in purchase. This percentage is in contrast with some other studies [9] where the fraction of purchases is much higher, but verified to be the norm for most current web companies. For most machine learning algorithms this is a problem: when one of the two classes is so under represented, most algorithms will simply output a model that always says "non-buyer" (with

a 98% accuracy). To force the learning algorithm to pay attention to this minority of buyers, once the log was preprocessed, we filtered out most of the nonbuying sessions and prepared a smaller dataset with about 14,500 transactions of which about 50% led to purchase. This dataset was only used for the initial training phase. A second dataset was prepared containing transactions with the original proportion of 98% of non-buyers, which was used for testing the learned models.

As indicated in Figure 1, two Markov models for buyer and non-buyer sessions were produced, and their predictions were added to the static information, providing the final dataset on which several algorithms from the WEKA package were applied.

4.2 Description of experiments and results

The general goal of the experiments was to test the hypothesis that there is enough information in the processed logfile to make useful predictions about whether a session will end up buying. It is convenient to present our results using confusion matrices, which detail how many instances of each class are classified in that and other classes. In our case, our two classes are buy and non-buy so WEKA's confusion matrices look like

```
buy | non-buy | <-- classified as
----------------------
 X  |    Y    | buy
 Z  |    T    | non-buy
```

Here, X+T is the number of correctly classified instances and Y+Z is the number of incorrectly classified ones. In particular,

- Y is the number of false positives (hits that are classified as buyers but are not eventually followed by a purchase) and
- Z is the number of false negatives (hits that are classified as non-buyers but whose session ends in purchase).

Recall however that our ultimate goal is to use these predictions for prioritizing sessions, and deny the ones with lower priorities when server is under heavy load condition. The meaning of a false positive and a false negative in this context is very different. Rejecting a false negative (Z) session implies a

substantial loss (in revenue), so it is preferable to accept it even at the cost of keeping many false positives (Y) at in the system. Therefore, these two figures should be looked at separately and carefully.

Even more than the % of correctly classified instances, the quantities of interest are the well-known *recall* and *precision* measures, and one specific to our setting: the fraction of transactions predicted that lead to purchase, which we call we call *"%admitted"* since potentially these would be the sessions admitted in the server. More precisely,

- %admitted is (X+Z) / (X+Y+Z+T). This is the quantity that may be limited by the available infrastructure.
- the recall is fraction of real buyers that are admitted, X / (X+Y). A too small recall will make many potential buyers frustrated, so this quantity should be kept minimal.
- the precision is the fraction of predicted buyers that end up buying, X / (X+Z).

At this preliminary stage of research, we decided to use simple but easy-to-use learning methods. More sophisticated methods will be used in the future, once we have developed a better understanding of what kind of results we can expect and where the useful information is.

In all experiments, the 50%buyers-50%non-buyers dataset was used for the training phase, and in a first set of experiments, we used WEKA with static information only. That is, we configured WEKA not to use the variables coming from the Markov models. We tried the following learners:

- `Logistic`, a logistic linear regression.
- `j48`, WEKA's version of the C4.5 decision tree inducer.
- `NaiveBayes`, the well-known Naïve Bayes classifier.

We used WEKA's default parameters, with the exception of `j48` which was forced to place at least 20 examples in each tree leaf (parameter -M 20) to reduce the size of the tree. The results are given in Figure 2. One can see that there are noticeable, but not drastic, differences in %admitted and recall. But we believe that the really significant figure is that of precision. Precision is practically the same in all three models, around 15% - 17%. In an

	j48 classifier	NB classifier	Logistic
%accuracy	76.5	78.1	72.7
%admitted	25.7	22.9	29.8
%recall	66.9	57.5	68.9
%precision	17.2	16.6	15.3

Figure 2: Models built by different classifiers with static information only

	j48 classifier	NB classifier	Logistic
%accuracy	75.8	77.8	74.2
%admitted	27.0	23.4	28.2
%recall	70.8	58.6	68.1
%precision	17.4	16.6	15.9

Figure 3: Models built by different classifiers with both static and dynamic information

overload situation, if this information about sessions is used, the number of admitted buyers compared to a session without admission control mechanism (or where admitted sessions are chosen at random) would be about 16% to 2% an 8x increase in the proportion of buyers.

In a second set of experiments we wanted to see whether the dynamic information is useful. We performed the same experiments as before but this time letting WEKA use the two variables coming from the two Markov models. The results are given in Figure 3. The results are only marginally better than those obtained with static information only. This has two possible explanations: either dynamic information does not add anything interesting over static one, or we have not yet captured it correctly with our simple Markovian models. At this moment, we are still inclined to believe that there has to be a way of making use of dynamic information. We plan to investigate more refined models in the near future.

In a third set of experiments we wanted to simulate the effect of varying resources on recall and precision. More precisely, we performed the experiments above but forcing the three models to admit a fixed number of users N. Since the number of users admitted in Figures 2 and 3 was in the range $N = 25,000 \ldots 30,000$, we tried situations where less users can be admitted ($N = 10,000$ and N=$5,000$) and more users can be admitted ($N = 30,000$

	N=5,000	N=10,000	N=30,000	N=50,000
%accuracy	91.5	89.2	75.8	60.7
%admitted	4.8	10.0	27.0	43.7
%recall	21.5	44.2	70.8	82.9
%precision	29.8	29.2	17.4	12.6

Figure 4: Models built by the `j48` classifier forcing %admitted to different values.

and $N = 50,000$). The number of admitted users was varied, for the time being, by assigning different weights to false positives and false negatives using WEKA's `CostSensitiveLearning` method. We present the results for the `j48` method only. One can observe that as admission is made harder (N decreases), both recall and precision strictly grow. In other words, as competition gets stronger, our system tends to let in only the most promising sessions. This is an important conclusion for our ultimate goal: that of designing a load balancer that makes use of our predictions to adapt itself to the current workload conditions.

5 Conclusions and Future Work

Websites might become overloaded by certain events such as news events or promotions, as they can potentially reach millions of users. When a peak situation occurs most infrastructures become stalled and throughput is reduced even though there are more users. To prevent this, load admission control mechanisms are used to allow only a certain number of sessions, however current session based admission systems don't differentiate between users and might be denying access to users with the intention to purchase. As a proof of concept, we have taken a dataset from high traffic online travel agency to perform experiments to approximate users purchasing intentions from their navigational patterns.

¿From our experiments, we are able to show that by training a model from previously recorded navigational information on a website, when a peak load situation occurs, by obtaining a sessions purchase information we are able to increase the number of sales per unit of time. In our preliminary experiments, we have increased the precision (or proportion of admitted users that buy)

from 2% to 16%, a factor of 8. The maximum number of allowed users to the site should be flexible, according to the infrastructure's capacity; by assigning different weights to false positives and false negatives, the model can adapt itself dynamically maintaining a reasonable precision. We have also tried to incorporate as input to the learner the predictions generated by two Markov chains modelling users and non-users. The increment, however, was not significant enough to draw final conclusions.

We can conclude that admission control, and resource management in general, is a promising application field for automatically learned user models. Using models of user navigation we have been able to show that in overload situations we can restrict the access to a web application to only a proportion of all the demanding customers while only reducing the revenue that they generate by a factor significantly lower.

As future work we plan to further investigate other models, including hidden Markov models, Bayesian Networks, and k-means clustering, to improve predictions. We are also going to explore other classifications i.e. magnitude of the transaction, type of product and profit margins; and their combinations to increase productive efficiency. Before trying our method in production websites for real-time load admission control, we are evaluating the performance of these algorithms in a manager to prioritize user sessions.

6 Acknowledgements

This work is supported by the Ministry of Science and Technology of Spain and the European Union under contract TIN2004-07739-C02-01.

R. Gavaldà is partially supported by the 6th Framework Program of EU through the integrated project DELIS (#001907), by the EU PASCAL Network of Excellence, IST-2002-506778, and by the DGICYT MOISES-BAR project, TIN2005-08832-C03-03

For additional information about the authors, visit the Barcelona eDragon Research Group web site [16].

References

[1] D. Bonino, F. Corno, G. Squillero. *A real-time evolutionary algorithm for Web prediction.* Proceedings of the IEEE/WIC International Conference

on Web Intelligence (WI'03), pp 139-145, October 2003.

[2] L. Cherkasova, P. Phaal. *Session-Based Admission Control: A Mechanism for Peak Load Management of Commercial Web Sites.* IEEE Transactions on Computers, vol. 51, n. 6, pp. 669-685, June 2002.

[3] M. Deshpande, G. Karypis. *Selective Markov models for predicting Web page accesses.* ACM Transactions on Internet Technology (TOIT), v.4 n.2, pp.163-184, May 2004.

[4] J. Guitart, V. Beltran, D. Carrera, J. Torres, E. Ayguadé. *Characterizing secure dynamic web applications scalability.* 19th International Parallel and Distributed Processing Symposium, Denver, Colorado, USA, pp. 166-176, April 4-8 2005.

[5] J. Guitart, D. Carrera, V. Beltran, J. Torres, E. Ayguadé. *Session-Based Adaptive Overload Control for Secure Dynamic Web Applications.* 34th International Conference on Parallel Processing (ICPP'05). Oslo, Norway, June 14-17, 2005, pp. 341-349.

[6] J. Kephart. *Research Challenges of Autonomic Computing.* Proceedings of the 27th International Conference on Software Engineering, St. Louis, Missouri, pp 15-22, May 15-21, 2005.

[7] J. Kephart, D. Chess. *The Vision of Autonomic Computing.* IEEE Computer, January 2003.

[8] B. Lan, S. Bressan, B. C. Ooi, K. Tan. *Rule-Assisted Prefetching in Web-Server Caching.* Proceedings ACM Int. Conference on Information and Knowledge Management (CIKM'00), pp. 504-511, November 2000.

[9] D. A. Menascé , V. A. F. Almeida, R. Fonseca, M. A. Mendes. *A methodology for workload characterization of E-commerce sites.* Proceedings of the 1st ACM conference on Electronic commerce, pp. 119-128, 1999.

[10] A. Nanopoulos, D. Katsaros, Y. Manolopoulos. *Effective Prediction of Web-user Accesses: a Data Mining Approach.* Proceedings WEBKDD Workshop, 2001.

[11] M. Rabinovich, O. Spatschek. *Web caching and replication.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2002.

[12] R. R. Sarukkai. *Link prediction and path analysis using Markov chains.* Computer Networks: The International Journal of Computer and Telecommunications Networking, Volume 33 , Issue 1-6, pp. 366-386, 2000.

[13] Y. Wei, C. Lin, F. Ren, E. Dutkiewicz, R. Raad. *Session Based Differentiated Quality of Service Admission Control for Web Servers.* International Conference on Computer Networks and Mobile Computing (ICCNMC'03), 2003, pp. 112-116.

[14] I. H. Witten, E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition).* Morgan Kaufmann, 2005. Software available from `http://www.cs.waikato.ac.nz/~ml/weka`.

[15] Q. Yang, H. H. Zhang , T. Li. *Mining web logs for prediction models in WWW caching and prefetching.* Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 473-478, 2001.

[16] WebPage. Barcelona eDragon Research Group. Technical University of Catalonia. `http://research.ac.upc.edu/eDragon`, 2006.