Generating Intelligent Links to Web Pages by Mining Access Patterns of Individuals and the Community

By Benjamin Lambert and Omid Fatemieh

Abstract

The goal of this project is to make use of Web site access logs to make intelligent link recommendations for an organization's Web site. Several methods are used to make recommendations with varied parameters. Each method is evaluated based on whether automatically recommended pages are accessed later in the same browsing session. Several evaluation metrics are used to compare the methods. Our best results show a recommendation accuracy of around 60%. These techniques are applied and implemented with existing data sets as well as new data from UIUC CS department's Web site.

1. Introduction

No Web site can claim to be perfect. Some sites are more easily navigated than others and users may have varying preferences as to how a site is laid out. An ideal Web site would need to be constantly updated so that the most relevant information for each user is always the easiest to access.

The goal of this research is to investigate how we can make an organization's Web site most easily used without the constant intervention of the site's administrator. To make the site more easily used we attempt to automatically provide convenient links to "recommended" Web pages. Several techniques are used to provide recommended links to pages linked to from the current page as well as other pages on an organization's Web site.

Since we would like to recommend pages without any user or administrator effort, we must rely on other data sources to infer which pages a given user is likely to prefer. The two main sources of information utilized respectively capture information about both the community's patterns and individual patterns: the organization's Web site access log and individual user history.

There are various scenarios in which a user may benefit from good recommendations. For example, a user may know the page they are looking for, but have to click several links before arriving at the desired page. In this case, we would like to recognize frequently occurring sequences of page requests. Then if it is detected that a user has begun to follow one of these sequences, the terminal page of the sequence should be recommended.

Another scenario where a user may benefit from recommendations is when the user does not know what page they are looking for. In this case, we would like to use the community's collective behavior to recommend pages. For example, the most frequently clicked links show which are the most popular pages.

The benefits of providing this service can be seen with a simple example. If the Computer Science Department has a page listing seminar announcements buried many links from the main page and many users access this page on the day a popular researcher is speaking, the Web server will likely receive many requests for the pages that link the main page to the particular seminar's page. If the server detects this sequence of requests occurring more frequently than other sequences, it can provide a convenient link to the content. If the pattern is very common, the server might display a link directly to the seminar page from the main page. If the pattern is only moderately common, it might provide a link from the other pages *en route* to the seminar page.

This tool would be useful to both Web browsers and Web content publishers. Web browsers would benefit from the convenience of getting good quality page recommendations. These recommendations could save the users' time and recommend pages that users would not otherwise encounter. Web publishers benefit by providing the easy access to in-demand content dynamically and conveniently without effort.

2. Related Research

The focus of much of the previous work on mining Web access patterns has been on Web personalization [1][2] which is not the center of our focus. However, there has also been some closely related work to what we have done. Masseglia et al. [3] analyze access log files with data mining techniques such as mining association rules and sequential patterns. After interesting patterns are discovered, they use the patterns to customize dynamically the hypertext organization. This is done by adding navigation links to each page based on the results of the data mining. Similar work is done on the Web site logs of the Wharton Business School at University of Pennsylvania [4]. A survey of the previous work in this area has been done by Eirinaki et al. [5]. Gunduz et al. [8] proposed a clustering-based method that calculates the similarity of browsing sessions based on the amount of time spent viewing similar pages.

Another general theme of research on Web server log analysis for Web page recommendations focuses on probabilistic methods, specifically first-order and higher-order Markov models. Davison [7] surveys Markov-based probabilistic techniques for Web page recommendations. Deshpande et al. [9] show that higher-order Markov models are impractical for many applications due to the large number of states and increased space and runtime requirements. In Section 3.3 a dynamic Markov model is proposed which models temporal trends of interest shared among many users of the Web site.

Our work differs from previous work in that we attempt to model both user and community browsing patterns in real-time. The other achievement in our work is that by combining different methods which incorporate different users' needs, we make highly dynamic and useful recommendations to users.

3. Our approach

To automatically generate Web page recommendations several techniques are used. These techniques include: dynamic first-order Markov model, truncated dynamic first-order Markov model, trained machine learning classifiers, and sequential pattern mining. Each technique has its strengths and weaknesses. For example, the sequential pattern mining algorithms are better able to recommend pages that are several clicks away from the current page whereas the Markov models are better able to recommend the links on the current page. The machine learning algorithms are more easily extensible in that the feature types can easily be changed.

The following sections describe each method formally. Techniques for evaluation and results are described in Sections 4 and 5.

3.1 Preliminaries

From a Web server's perspective, page requests are just a sequence of requests from various IP addresses at irregular intervals. The Web server has no internal concept of a user, what that user is looking for, nor if an extended period without any requests means the user is reading a page or is no longer using the site. Thus, it is necessary for us to define a few concepts of which the Web server is unaware.

Definition. *Browsing Session. A Web browsing session is defined to be the longest sequence of requests from the same IP address where each request is made within k seconds of the previous request.*

For most of the experiments described in this paper we set the session length to be 600 seconds or 10 minutes. The intuition is that most Web pages on the CS departmental Web site do not take more than 10 minutes to read in entirety and a 10 minute period of inactivity indicates the user is no longer actively using the Web site.

It should be noted that this definition does not imply that a browsing session is a single person or that the browser is human. A session is defined for an IP address, but NAT networking may mean that many people are browsing the site using the same IP address. Many requests may also not be from humans. Such requests likely are made by Web crawlers which systematically request all the pages on a site so that they can be index and later searched.

In the experiments described in this paper, Web requests made by crawlers are filtered from the data. In the data cleaning stage, crawlers are filtered in two ways. One is by ignoring requests that do not come from a browser-declared browser type of *"Mozilla"* (this type includes Internet Explorer, Netscape, Firefox as well as other popular browsers); crawlers often give themselves away as *"Alexa crawler"* or *"Google bot"*. The second is by using the convention that Web crawlers download the file *"robots.txt"* from a Web server before crawling the server. The *"robots.txt"* file lists parts of the Web site that are not permitted to be crawled. Thus, we also filter crawlers by ignoring all IP addresses that request the *"robots.txt"* file.

3.2 Dynamic First-Order Markov Model

The first attempt to make recommendations is based on a continuously updating Markov model. In this model, every time a user transitions from one page to another page during a session the frequency of that transition is incremented.

The Markovian assumption for this first-order model is that the next requested page depends on the previously requested page. In other words, the next requested page depends on the page the user is currently looking at. To choose recommended pages, the pages that have the highest probability given the current page are selected. To make more than one recommendation, the *k* pages with highest conditional probability are recommended.

$$\arg\max_{recd \in W} p(recd \mid curr) \quad where \quad p(recd \mid curr) = \frac{\#\,req.\ \ for\ \ curr\ \ then\ \ recd}{\#\ \ req.\ \ for\ \ curr}$$

This is slightly different than a regular Markov model in that it is continuously updated. The model begins with equal probabilities for all transitions and updates the probabilities to reflect each the probability of each transition based on all the data seen so far.

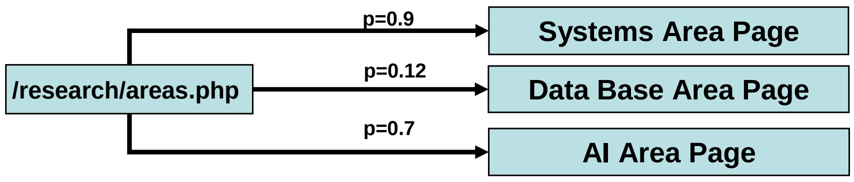


Figure 1. First Order Website Markov model.

3.3 Truncated Dynamic First-Order Markov Model

The standard first-order model described in the previous section cumulatively models the browsing behavior on the Web site from when it beings recording page requests. This approach is able to model long-term browsing patterns that do not change over time. However, this approach cannot effectively capture ephemeral browsing patterns. For example "bursty" request patterns such as those to upcoming seminars or admissions pages during admission season are not effectively modeled. In an attempt to model very recent browsing patterns, we also maintain a model of the page requests for the previous *t* seconds.

In order to do this, we maintain both a transition matrix for all the transition probabilities (counts) and a transition queue of all of the transitions that occurred during that last *t* seconds. Every time we make a recommendation, we check the tail of the queue to determine if the least recent transition occurred in the last *t* seconds. If it did not, the transition count in the transition matrix is decremented for the outdated transition. Each new page request is added to the head of the queue along with the time the request was made. With this model, the parameter *t* can be varied to change the window size of the activity modeled.

3.4 Trained Machine Learning Classifier

A natural approach to learning Web browsing patterns is to learn a classifier that outputs a page to be recommended. This classifier could take as input any number of things including: recently browsed pages, user preferences, and site meta-information. To begin with, we trained a classifier with the names of all pages previously visited this session as input. In an online fashion the learning algorithm updates its linear function every time it makes an incorrect recommendation. The linear function provides an activation level for each possible recommendation, so we can recommend the top *k* pages. However, the linear function is updated every time the top recommendation is not correct.

Details about the learning algorithms are beyond the scope of this paper. The algorithms used are perceptron, winnow, and naïve Bayes with the default parameters of the SNoW learning architecture. For each of the experiments described in this paper, a single classifier is trained with training examples where the target classes are any page on the Web site. The current page is included as a feature. An alternative architecture would be to learn one classifier per page on the Web site rather than one large classifier for all pages on the Web site.

All the results shown in this paper for this learning architecture are based on a user's browsing history. So it is the page names (or unique identifiers of) that constitute the features of each example. In addition to just using the names of previously viewed pages, we also used the *fex* feature extraction tool to also use combinations of recently visited pages as features.

The two additional combinations of features we uses are called by SNoW: bigrams and sparse collocations. *Bigrams* are ordered pairs of consecutive page requests. Sparse collocations of requests are conjunctions of pages requested in the current browsing session. For example, the browsing sequence A, B, C produces the bigram features: {(A,B), (B,C)} and the spare collocations {(A,B), (A,C), (B,C)} (where the order does not matter). Thus each training and testing example consists of a generated list of features based on the names of recently requested pages and the target class is the next requested page.

Features can also easily be generated from the pages contents of each page. For example, each distinct word on a recently visited page could be a feature. Intuitively, the contents of each page should help to make recommendations because a Web page may serve several purposes and words on the page may be more indicative than the viewing of the page itself. Experiments contained in this paper do not utilize the page contents due to limitations of the *fex* and *SNoW* software: trained and testing example file are prohibitively large in *fex* format.

3.5 Sequential Pattern Mining

The techniques described in the previous sections make recommendation for what will be the next page in a sequence of page requests. However, those techniques do not attempt to recommend the pages that the user is ultimately looking for. For example, if the sequence A→B→C→D→E is repeated in the log by many users, page E should be recommended to users from page A.

For this project, we divide the entire log into disjoint browsing sessions and treat each sequence of user page requests as analogous to a biological sequence in which the same pattern may occur many times in the same sequence. Thus the problem can be also be visualized as a biological pattern mining problem, with Webpage names instead of biological bases.

Web page recommendations are constructed from mined sequential patterns of length two. These patterns are ordered pairs of page names. The pairs occur in order in the original sequences with or without other pages in between. The PrefixSpan algorithm [6] is used to determine the most common sequential patterns of length two. Recommendations for page X are given by taking the most frequent patterns (i.e. those with greatest support) of length two that begin with X.

4. Evaluation and Results

The best way to evaluate the utility of the recommended links would be to incorporate them into a popular Web site and monitor the behavior of users that go to the Web site to see what percentage of them click on the recommended links. However, access to such a Web server was unavailable. Instead Web server logs were used, with the expectation that users will often eventually find the page they are looking for even if not immediately with the assistance of a recommendation system.

The general idea is to divide the entire log into two sets: training set and testing set. After this, our model is trained using the training data and the testing set is used for performing evaluations. The assumption is that the testing data will reflect actual user behavior in the future.

The data that we used for our evaluation is described in Section 4.1. As the evaluation process is slightly different for each approach, we describe them separately for each of our approaches in sections 4.2, 4.3, and 4.4.

4.1. Data

The Web server logs are long lists of Web page requests. Each request is represented by IP, time, requested page, browser type, referring page, etc. We used the following Web server logs for our evaluations:

- CS Department Web server logs from Dec 6, 2004 to Feb 28, 2005.
- NASA Kennedy Space Center Web server log collected over July and August 1995.
- CS Department Web server logs from Nov 1, 2003 to Nov 11, 2003.

In order to use the logs for our training and evaluation purposes we had to do the following to make sure the data is clean and ready for training and evaluation:

1. Actual IP addresses were removed for privacy reasons, and were replaced with arbitrary IP addresses.
2. The requests for files of type .jpg, .gif, .css, etc. were discarded because often a single page request triggers many such uninformative requests.

3. Many of the requests to the Web server are actually made by the crawlers. Well-behaved crawlers are supposed to first get the file “robots.txt” on the Web server before crawling. We take advantage of this and discard requests from all IP addresses that request “robots.txt”.
4. Unsuccessful GET requests were discarded. These requests include requests that the return code for them is not 200 (e.g. 404, etc).
5. Consecutive requests for the same page, which we call refreshes, were also deleted from the log.

Finally we divide each user’s activity into possibly several sessions based on the inactivity time between clicks. If a user remains inactive for more than X seconds, we consider his/her click as the start of a new session. We will use the notion of a session frequently during the description of our evaluation techniques.

4.2 The Markov model

To evaluate the utility of Markov model’s automated link suggestions we analyze the clicks on the recommended pages. To achieve this using the logs, we plug the current page in each session into our Markov model recommender and compare the result with the actual next click of the user as recorded in the log. If they are the same, we count it as a hit, otherwise it is a miss. The ratio of the hits to the total number of recommendations is the accuracy. We also measured the accuracy in cases that we recommend two and three pages, in which a recommendation is a hit if the actual next click of the user is among the two or three recommended pages. The results of these measurements are summarized in Figure 2. Recommendations made at the end of a browsing session are counted neither as hits nor misses since the user does not request any additional pages this session.

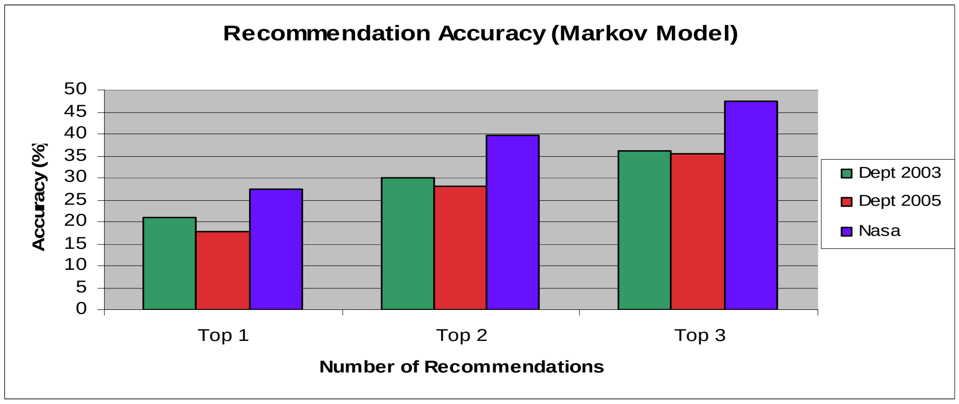


Figure 2. The accuracy of the Markov model recommendations on three data sets.

We also measured the accuracy of the truncated dynamic Markov model with various window sizes. The results are shown in Figure 3 below. These results suggest that the larger the window size the better the recommendations will be. Basically, using the truncated dynamic Markov model would be more useful in cases where temporal trends of interest dominate long-term unchanging browsing trends. The truncated dynamic Markov model may be useful if requests consistent with the long-term browsing trends are separated from those that may reflect short-term trends.

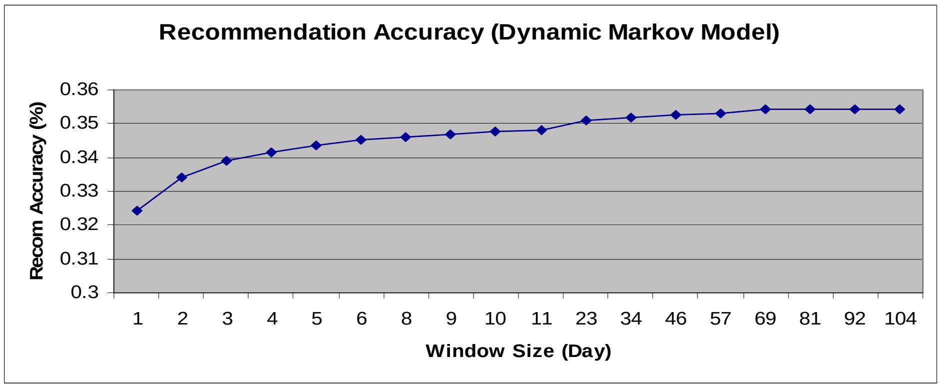


Figure 3. The accuracy of the dynamic Markov model recommendations on the Dept 2005 data set.

4.3 Machine learning

Similar to what we did for the Markov model, to evaluate the utility of the automated link suggestions made by the machine learning approach we analyze the clicks on the recommended pages. Here again we use the logs to measure the accuracy. Hit and missed recommendations are defined in the same way as the way we did for measuring the accuracy of the Markov model. The only major difference here was that we also had to consider the previous clicks of the user in the current session before making the recommendation.

Due to lengthy running time for feature extraction, not all results were available at the time of writing. It is not surprising that using bigram features achieves higher performance than unigram features, since they are a better indication of the "direction" that the user is going in the topology of the Web site. If it is the order of bigrams that translates into performance gains, then it is not surprising that sparse collocations fare worse than bigrams. However, one would expect that unordered conjunctions are rich with information. Perhaps, to the contrary, they add noise and obstructive information to the task.

It may not be surprising that accuracies for this technique are quite low. The classifier is essentially a 2,000 class multi-class classifier where each example has only a handful of active features. A relatively small number of distinct pages are actually recommended. A "pruning" of the learning space would seem to alleviate some of these problems. For example, employing one multi-class classifier per page, with far fewer classes to choose from, would likely generate more accurate classifiers.

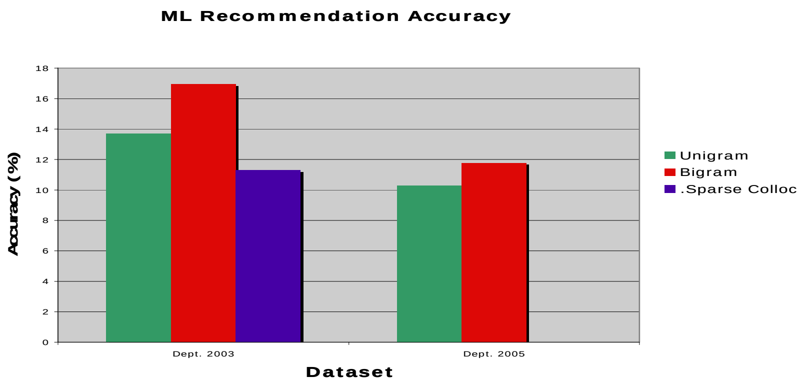
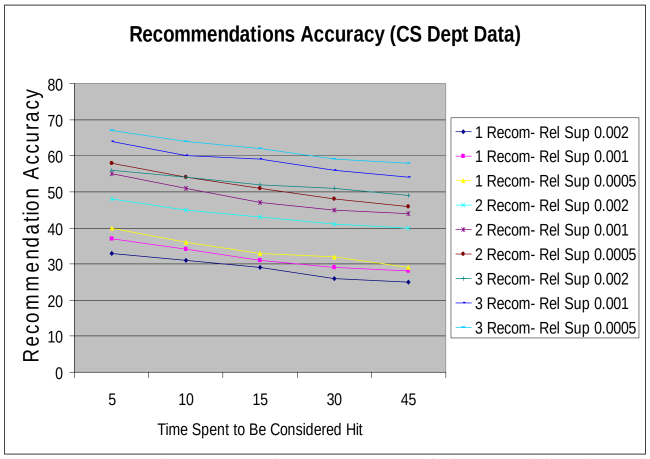


Figure 4. The accuracy of the machine learning approach recommendations on the dept 2003 and dept 2005 data sets.

4.4 Sequential Pattern Mining

To perform an evaluation of the utility of our recommendations based on the sequential pattern mining approach, we used a slightly different method compared to that described in Sections 4.2 and 4.3. Since we use the sequential pattern mining approach to recommend pages that are possibly a few clicks away, we cannot just rely on user's next click to decide whether a recommendation is a hit or not. For this evaluation, the mined patterns get a hit if the user requests the recommended page this session and remains on the page for at least *t* seconds.

Using the above definition for a hit, and making one to three recommendations for each page, we got the following accuracy for the recommendations made by the sequential pattern mining algorithm:



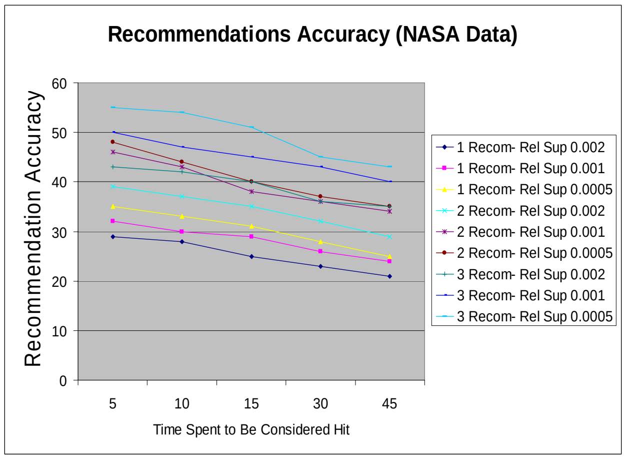


Figure 5. The accuracy of the machine learning approach recommendations on the Dept 2005 and NASA data sets.

The higher accuracies achieved by the sequential pattern mining algorithm does not necessarily mean that this approach works better than the other two. One reason for this is that the evaluation method that we used here was a bit different from what we used for evaluating the Markov model and also the machine learning based approaches. The other reason is that each of these algorithms can predict different types of *next clicks* and a combination of the above approaches is supposed to best meet users' needs.

5. Conclusions

This research attempts to use statistical models, machine learning, and pattern mining to make quality Web page recommendations to browsers of the Web. Attempts to model ephemeral community patterns with a truncated dynamic Markov model demonstrate that ephemeral browsing patterns cannot easily be captured without sacrificing higher accuracy temporally-static patterns, and instead must be separately handled. Additionally, some of the challenges and parameters for making page recommendations via a trained machine learning classifier are demonstrated. Lastly, sequential pattern mining algorithms are used to mine recommendations of a slightly different type with success.

No recommendation system is a panacea for Web browsers. Some systems work well to guide a user to links personally clicked very frequently, or those clicked very frequently by others. Other systems work very well at providing statistically cogent recommendations whereas others will require manual tuning or feature engineering. Our conclusion and recommendation is that a successful recommendation system will require a sophisticated combination of many factors including: community patterns, patterns of community segments, personal preferences, types of links recommendations that are wanted, as well as information intrinsic to the pages and meta-data about the pages.

References

[1] Holland S., Ester M., Kießling W.: Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications. Proc. Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD 2003) 204-216.

[2] M. Baglioni, U. Ferrara, A. Romei, Salvatore Ruggieri, Franco Turini: Preprocessing and Mining Web Log Data for Web Personalization. AI*IA 2003: 237-249.

[3] F. Masseglia , P. Poncelet , M. Teisseire: Using data mining techniques on Web access logs to dynamically improve hypertext structure, ACM SIGWEB Newsletter, v.8 n.3, p.13-19, October 1999

[4] Ramakrishnan Srikant, Yinghui Yang: Mining Web Logs to Improve Website Organization. Proceedings of the tenth international world wide web conference, pages 430-437, Hong Kong, May 2001.

[5] Magdalini Eirinaki, Michalis Vazirgiannis: Web mining for web personalization ACM Transactions on Internet Technology (TOIT), Volume 3 , Issue 1, p. 1-27, February 2003.

[6] J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, " PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth ", Proc. 2001 Int. Conf. on Data Engineering (ICDE'01), Heidelberg, Germany, April 2001.

[7] Brian D. Davison: Learning Web Request Patterns. Web Dynamics 2004: pp 435-460.

[8] Ş. Gündüz and M.T. Özsu. "Recommendation Models for User Accesses to Web Pages", In *Proceedings of 13 Int. Conf. Artificial Neural Networks and 10th Int. Conf. Neural Information Processing*, Istanbul, Turkey, June 2003.

[9] Mukund Deshpande, George Karypis, Selective Markov models for predicting Web page accesses, ACM Transactions on Internet Technology (TOIT), v.4 n.2, p.163-184, May 2004.