DETECTING PARALLEL BROWSING TO IMPROVE WEB PREDICTIVE MODELING

Geoffray Bonnin, Armelle Brun and Anne Boyer
LORIA – KIWI Team, B.P. 239, 54506 Vandœuvre-Lès-Nancy Cedex, France

Keywords: Parallel browsing, Web recommendation, Predictive modeling.

Abstract: Present-day web browsers possess several features that facilitate browsing tasks. Among these features, one of the most useful is the possibility of using tabs. Nowadays, it is very common for web users to use several tabs and to switch from one to another while navigating. Taking into account parallel browsing is thus becoming very important in the frame of web usage mining. Although many studies about web users' navigational behavior have been conducted, few of these studies deal with parallel browsing. This paper is dedicated to such a study. Taking into account parallel browsing involves to have some information about when tab switches are performed in user sessions. However, navigation logs usually do not contain such informations and parallel sessions appear in a mixed fashion. Therefore, we propose to get this information in an implicit way. We thus propose the TABAKO model, which is able to detect tab switches in raw navigation logs and to benefit from such a knowledge in order to improve the quality of web recommendations.

1 INTRODUCTION

Predicting web users' future paths is one of the most important tasks in Web Usage Mining (WUM). WUM can be defined as "the process of applying data mining techniques to the discovery of usage patterns from web data" (Srivastava et al., 2000). Web predictive modeling is useful for many purposes such as web page research (Tan and Kumar, 2002), latency reduction (Schechter et al., 1998), arrangement of the links in a website (Chi et al., 1998), web recommendation (Nakagawa and Mobasher, 2003), etc. The most popular techniques used in this domain are association rules (Agrawal et al., 1993), sequential patterns (Agrawal and Srikant, 1995) and Markov models (Pitkow and Pirolli, 1999). These techniques modelize users' sessions by exploiting a training corpus, processed from navigation logs. These logs usually consist in sequences of resources along with timestamps and users' IP addresses, or ids. The usual challenge is to provide a model that is a tradeoff between predictive accuracy, coverage, space and time complexity (Pitkow and Pirolli, 1999; Deshpande and Karypis, 2004).

Although widely studied, this domain is constantly evolving, and is perpetually challenging. Indeed, the technology used to surf on the web becomes more ergonomic and sophisticated everyday, which involves changes in the usual users' behaviors. In particular, present-day browsers provide interfaces called *tabs* that allow several pages to be contained in the same window and to switch from one to another, which is commonly referred to as *tabbing*. Although this phenomenon is very common in current web users' behavior, very few approaches have dealt with it. In several works, it is assumed that user sessions include several *tasks* that can be characterized (Jin et al., 2005; Gündüz and Özsu, 2003). In one session, a user can perform several tasks by tabbing, which is referred to as parallel browsing. Tabbing actions are not stored in the navigation logs, only the time-ordered actions are stored, and when parallel browsing is performed the resulting session consists in a mixed sequence of resources (mixed tasks). On the client side, tabbing can be categorized into two categories: inter-site and intra-site tabbing. On the server side, only intra-site tabbing can occur. Regarding inter-site tabbing, it is easy to detect which pages correspond to which task, as one can simply consider the url associated with the resources. In the case of intra-site navigation, it is more difficult to detect tabbing. For instance, the typical way to perform such an activity is to click on a link using the middle button of the mouse, or to hold down the CTRL key while clicking on a link. As neither this action nor tab switching are indicated in the navigation logs, determining

Bonnin G., Brun A. and Boyer A..
DETECTING PARALLEL BROWSING TO IMPROVE WEB PREDICTIVE MODELING.
DOI: 10.5220/0003115905040509
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2010), pages 504-509
ISBN: 978-989-8425-28-7

which resource within a session corresponds to which task is a difficult problem. State-of-the-art predictive models do not take into account tabbing, they usually simply assume that one session is made up of one single task. We assume that the prediction ability of these models should be increased if the tasks were extracted from the sessions and considered for prediction. In this paper we aim at discriminating between the different tasks mixed within a session. We propose to tackle this issue by using sequence alignment algorithms. We first propose an algorithm to extract tasks from sessions. We then propose a model that exploits the extracted tasks in order to perform resource recommendation.

The rest of this paper is organized as follows. We are first interested in works related to parallel browsing and task-level modeling. We then present our proposition to extract linear sessions from raw logs in which sessions are mixed. Then, we introduce the model we propose to perform recommendation. Conclusion and perspectives are put forward in the last section.

2 RELATED WORK

Web predictive modeling has been widely studied in the past decade. In this section, we first present a brief overview of the most usual models. We then turn to the works related to parallel browsing and multi-task modeling.

2.1 Web Predictive Modeling Overview

Association Rules and Sequential Patterns. One way of exploiting past actions to predict future ones, is the use of association rules. Association rules have been initially used for mining supermarket basket (Agrawal et al., 1993) to extract information about purchased items dependencies. An association rule is an expression of the form $X \rightarrow Y$, where X and Y are sets of items. X is called the antecedent and Y the consequent. An association rule means that, in one transaction, when users have purchased all resources in X then there is a high probability that they will purchase Y. Using association rules in the frame of web usage modeling thus enables to take into account non-ordered sets of resources in the history. Sequential patterns (Agrawal and Srikant, 1995; Lu et al., 2005) are the sequential form of association rules, and are thus more constrained than association rules.

In both approaches, statistical models are built using a training corpus by counting the sets of resources or sequences of resources. Then, during the prediction step, all possible antecedents in current user's navigation history are compared to the antecedents in the model. If some antecedents match, then the corresponding consequents are recommended.

Sequential patterns can be contiguous (Jianyong et al., 2007) or non contiguous (Ayres et al., 2002). Looking for non contiguous sequences of unlimited sizes induces a huge amount of combinations. So the step of pattern discovery has to limit the size of the patterns to discover. A sliding window with a fixed size is usually used during the pattern discovery step as well as during the recommendation step. However, the time complexity induced is still high. As they induce less combinations, the time complexity of contiguous sequential patterns is lower.

Markov Models. In the frame of web navigation Markov chains are used to predict the next resource according to the present state (the k previously browsed resources), which is referred to as Markov models of order k or k^{th}-order Markov models.

Markov models are built in the same way association rules and sequential patterns are, *i.e.* by browsing a training corpus and counting sequences of size $k+1$. The prediction step is similar to the one of sequential patterns too: the previous actions are compared to the states in the model, and if some state matches, then the most probable corresponding resource is predicted.

The performance of Markov models can be evaluated in terms of coverage, accuracy and state perplexity. Coverage is the percentage of cases where a state of the model matches the current history. When a large enough training data is available, higher order Markov models provide a higher accuracy; however moving to higher orders usually involves a higher state complexity and a lower coverage, as no large enough training data can be found in order to set values to each possible states. Thus, the use of Markov models involves a tradeoff between accuracy, state complexity and coverage (Pitkow and Pirolli, 1999). One way to provide both high accuracy and high coverage is to use several Markov models having various orders. For example, one can try to provide a recommendation using a Markov model of order 3 (high accuracy), and if no matching can be performed, try a Markov model of order 2 (lower accuracy but higher coverage), and so on, until a recommendation can be provided. In the worst case, a Markov model of order 0 is used, which corresponds to the overall probability of one single resource, without considering previous resources. Using such a scheme, a full coverage can be reached, while providing a good accuracy in the recommendations. This scheme is called the all-k^{th}-

order Markov model, and is one of the best performing predictive models of the state-of-the-art (Deshpande and Karypis, 2004). Notice that under the same pruning conditions, it is similar to sequential patterns in which patterns are contiguous.

One of the major drawback of the all-k^{th}-order Markov model is its low robustness to navigation mistakes and parallel browsing. Indeed, it takes into account only strictly contiguous sequences, and if a given user makes parallel navigations or goes to an unwanted resource, the model will not be able to capture the real path and will reduce the size of the history considered. Moreover, when the model does not match the complete history, the farthest resources are discarded, and the closest resources are always considered while some of them may be navigation mistakes and should be discarded.

2.2 Parallel Browsing and Multi-task Modeling

To the best of our knowledge, parallel browsing has not yet been directly studied in the frame of web predictive modeling. However some works have dealt with related interests.

In (Weinreich et al., 2006), a 195 days study of user browsing behavior is presented. The authors compare their study to two other user browsing behavior studies, that date back to 1995 and 1997 (no other similar study has been performed during the nine previous years). The results show important changes in users' behavior within these nine years, such as a decreased use of the back button from 30% in the mid-nineties to less than 15% currently. The use of multiple browsing windows[1] has increased from less than 1% to more than 10%. Related to our purpose, in this study the action of opening a new window is only registered when it is done from the menu item. Moreover, this study also mentions that tabs are used by participants, but no figure about their frequency is provided. In (Viermetz et al., 2006), a clicktree model is proposed, in which all the possible tabbed paths of user logs are stored into a tree. Using this model, they found that tabbing is performed by users from 4% to 85% of the time, which is a quite large range. Moreover, handling such clicktrees involves huge time and space complexities and is not appropriate for common web usage mining applications such as web recommendation.

(Jin et al., 2005) propose a web recommendation system able to discover task-level patterns. The authors use probabilistic latent semantic analysis to characterize users' navigational tasks. They then use a bayesian updating to compute the probability of each task being performed according to a given active session. Then recommendations are computed using a maximum entropy model in which one of the features uses a first-order Markov model. So, the resulting model can detect task changes within one session and may be considered as a viable model for parallel browsing data, as it implicitly recognizes the different tasks. In the same spirit, (Bonnin et al., 2009) propose a skipping-based recommender that implicitly takes into account parallel browsing. This model is based on a discontiguous version of Markov models and computes recommendations based on a weighted combination of subhistories, *i.e.* small subsequences of user's current session. In this paper we go a step farther and identify more explicitly which resources in a user session correspond to which tasks.

3 EXTRACTING LINEAR SESSIONS

We now focus on the detection of tasks within sessions. We first define several concepts:

- We call a *task* a typical sequence of resources that can have several slight variations;
- We call *linear session* a session in which only one task is performed. Several different linear sessions may correspond to the same task while one linear session cannot correspond to different tasks;
- We call *nonlinear session* a session in which several tasks are mixed;
- Given two observed sessions X and Y, we say that X is a *subsession* of Y if X is included in Y;
- We define an *interrupted session* as a linear session that was abruptly stopped.

We now propose a new algorithm to extract linear sessions. A first solution would be to make the assumption that if a session X is a subsession of another session Y, then Y is not a linear session. However, this way to do is not sufficient as X may simply correspond to an interrupted session. Thus, the assumption on which is based our proposition for the extraction of linear sessions is the following: if two sessions X and Y do not correspond to the same task, and are both subsessions of a third session Z, then Z is not a linear session.

To extract linear sessions, a comparison of all the sessions is performed, and the corresponding non-

[1] From the point of view of parallel browsing, opening a new window can be considered as being similar to opening a new tab.

linear sessions are suppressed whenever the aforementioned situation is encountered. The corresponding algorithm is detailed in Algorithm 1. This extraction algorithm lies on two additional algorithms: sameTask(X,Y) that indicates whether two sessions X and Y are similar enough to be considered as belonging to the same task, and isSubsession(X,Y) that indicates whether X is a subsession of Y. These algorithms are defined in the following sections.

Algorithm 1: Extraction of the linear sessions.

```
Data: a list S of sessions
Result: A sublist of S with only linear sessions
for each session X in S do
    for each session Y after X in the list S do
        if NOT sameTask(X,Y) then
            for each session Z in S, Z ≠ X and Z ≠ Y do
                if isSubsession(X,Z) and isSubsession(Y,Z)
                then remove Z from S;
            end
        end
    end
end
```

3.1 Discriminating between Tasks

We propose to use a global alignment algorithm, as used for instance in (Gündüz and Özsu, 2003). The algorithm used by Gnduz and zsu, the FastLSA algorithm, is an enhancement in terms of storage space of the Needleman-Wunsch algorithm, which is one of the standard global alignment algorithms. However, the time complexity of the FastLSA algorithm is higher than the standard Needleman-Wunsch algorithm. Storage space is not problematic when computing the best alignment of two sessions, as the size of a session is usually small. We thus used the standard Needleman-Wunsch algorithm. Further details on this algorithm can be obtained in (Needleman and Wunsch, 1970).

Then, X is considered as belonging to same task as Y if their global alignment score value is close to the size of X (or equivalently Y).

3.2 Identifying Subsessions

Another element of our linear session extraction algorithm is how to determine whether a given session X is a subsession of a given session Y. A first way to determine this would be to simply check whether each element of X can be found in Y in the same order. However, as mentioned in the previous section, the sessions associated with one given task may show slight differences, and using such a strategy would be too restrictive. We thus propose to use an adapted local alignment algorithm, based on the Smith-Waterman algorithm (Smith and Waterman, 1981). We chose the Smith-Waterman algorithm because it is one of the standard local alignment algorithms.

The classical local alignment problem aims at finding the best alignment between two arbitrary subsequences of the input sequences X and Y. Our aim in this paper is a little bit different. Indeed, our goal is to detect overlapping of subsessions. The problem is thus to determine whether X can be aligned with a discontiguous subsequence of Y in such a way that any number of insertions can be applied between the elements of X without penalizing the final alignment score. We propose to apply a simple modification to the Smith-Waterman algorithm by penalizing only the insertions between the elements of Y, while not decreasing the score when insertions are performed between the elements of X. The resulting algorithm, detailed in Algorithm 2, is asymmetric. As a result, X can be considered as being a subsession of Y if the local alignment score value is close to the size of X.

Algorithm 2: Modified version of the Smith-Waterman algorithm.

Data: two sequences X and Y of sizes m and n
Result: score β of the optimal alignment
set $F(i,0) = 0$ for all $i = 0,1,...,m$;
set $F(0,j) = 0$ for all $j = 1,2,...,n$;
$\beta \leftarrow 0$;
for $i = 1$ **to** m **do**
  for $j = i$ **to** n **do**

$$F(i,j) \leftarrow max \begin{cases} 0 \\ F(i-1,j-1) + s(x_i,y_j) \\ F(i-1,j) - d \\ F(i,j-1) \end{cases} ;$$

    if $F(i,j) > \beta$ **then** $\beta \leftarrow F(i,j)$
  end
end

We propose the match function and mismatch penalties to have to following values: $d = -1$ and $s(x_i,y_j) = 1$ if $x_i = y_j$, and -2 else. In this fashion, if an insertion is performed on X and then another is performed on Y, the penalty is the same as for a mismatch (substitution).

4 THE TABAKO RECOMMENDER

We now present our Tabbing-based recommender. As mentioned in section 2, the all-k^{th}-order Markov model is one of the best performing model of the state-of-the-art. However, one of its major drawback is that it only takes into account contiguous se-

quences, and cannot handle efficiently parallel browsing and navigation mistakes. Nevertheless, when applied to linear sessions, it is the most appropriate algorithm. Indeed, it provides a good accuracy additionally to a full coverage, and a lower time complexity compared to the other state-of-the art models, especially open sequential patterns. So, we assume that if we are able to detect the tasks that have been mixed within a session and ignore navigation mistakes, then the all-k^{th}-order Markov model will still be applied to linear sessions, and thus will provide a good accuracy.

We propose to exploit the algorithms presented in the previous section in order to extract the tasks that have been mixed, and then apply the traditional all-k^{th}-order Markov model on these tasks. The resulting recommendation model is the TAbbing-Based All-K^{th}-Order Markov model (TABAKO model). Compared to the classical all-k^{th}-order Markov model, two major enhancements are provided:

During the Training Step. The linear sessions of the training corpus are first extracted and stored in order to be used during the recommendation step (section 3). Second, the model is built exactly as an all-k^{th}-order Markov model. Thus, instead of using all the training data, only the linear sessions are used. In this fashion, only useful sequences are used to train the model;

During the Recommendation Step. Before looking for a matching between the active user session and one of the states stored in the Markov model, the best overlapping of linear subsessions in the session is computed. In this way, the resources of the session that correspond to the same task are put together and theses tasks can be matched to the all-k^{th}-order Markov model.

The recommendation process is performed in two steps: (1) the extraction of the best overlapping and the extraction of the corresponding linear sessions, and (2) the creation of the recommendation lists by applying the all-k^{th}-order Markov model on the retrieved linear subsessions. The overall recommendation algorithm is presented in Algorithm 3.

Once the best overlapping has been computed, it is possible that some resources in the active session remain unlinked to any of the linear sessions. We then consider that these resources correspond to navigation mistakes and we propose to discard them. In this way, both parallel browsing and noise are handled.

4.1 Extracting the Best Overlapping

In the following, the list of resources that the user has previously browsed in the session will be referred to as the history $h = \langle r_1, ..., r_{i-1} \rangle$. The extraction of the best overlapping is performed according to an iterative process:

Algorithm 3: Recommendation process.

Data: Current user history h
An all-k^{th}-order Markov model M
A list of linear sessions L
Result: A recommendation list R
subhistories $\leftarrow \emptyset$;
$c \leftarrow$bestSubsession(h, L);
while $\|h\| \neq 0$ **and** $c \neq \emptyset$ **do**
  add c to *subhistories*;
  $h \leftarrow h - c$;
  $c \leftarrow$bestSubsession(h, L);
end
for each subhistory c in *subhistories* **do**
  $R \leftarrow$ merge$(R,$ buildList$(c, M))$;
end

1. Find the best subsession c of current user history;
2. Remove the matching elements of c in the history;
3. Reiterate on the remaining history, until no resource remains or no additional subsession can be found.

The search of the best subsession of a given user's history h is performed using the Smith-Waterman algorithm. For each linear session l extracted and stored during the training step, all the prefixes of l of size varying from $\max(\|l\| - 1, \|h\|)$ to 1 are first extracted. Indeed, as the current user session is still open, it cannot be compared to the entire linear sessions. For instance, a user may have browsed 4 resources of a given task with a usual length of 10. Last, the subsession with the highest score is returned and added to the set of sub-histories. This process is detailed in Algorithm 4.

Algorithm 4: Determining the best subsession.

Data: Current user history h
A list of linear sessions L
Result: The subsession that matches h the best
$C \leftarrow \emptyset$;
for each linear session l in L **do**
  for $i \leftarrow \max(\|l\| - 1, \|h\|)$ **downto** 1 **do**
    $l_i \leftarrow$substring(l, i);
    $(c, \beta) \leftarrow$Smith-Waterman(l_i, h);
    add (c, β) to L;
  end
end
return arg max L

4.2 Building the Recommendation Lists

After the determination of the best overlapping, the corresponding list of subsessions is extracted from the

user's active session. Then, for each subsession, a recommendation list is computed using the all-k^{th}-order Markov model. The resulting sublists are then merged according to the following policy: the resources issued from the subsession that has the best alignment score are put on the top of the list. The other recommendations are then appended to the end of the list, except when they are already in the list.

5 CONCLUSIONS AND FUTURE WORK

This paper focused on parallel browsing in the frame of web predictive models in order to improve recommendation accuracy. We proposed algorithms to discriminate between linear and nonlinear sessions. These algorithms exploit sequence alignment: global and local alignments. We then proposed a new recommendation model called TABAKO. This model is based on an all-k^{th}-order Markov model and exploits the sequence alignment algorithms to extract linear sessions from the active user's session.

In a future work, we plan to validate this position paper through experiments on a browsing dataset in which tab switching is performed. Such a study can include analysis of the amount of linear sessions obtained when using different parameters on our model, and a comparison to state-the-art models, such as the all-k^{th}-order Markov model and the approaches for obtaining more granularity presented in section 2.

REFERENCES

Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining Association Rules between Sets of Items in Large Databases. In Buneman, P. and Jajodia, S., editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216.

Agrawal, R. and Srikant, R. (1995). Mining Sequential Patterns. In *ICDE'95: Proceedings of the International Conference on Data Engineering*, pages 3–14.

Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. (2002). Sequential Pattern Mining Using a Bitmap Representation. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 429–435.

Bonnin, G., Brun, A., and Boyer, A. (2009). A Low-Order Markov Model integrating Long-Distance Histories for Collaborative Recommender Systems. In *Proceedings of the 13th International Conference on Intelligent user interfaces (IUI)*, pages 57–66.

Chi, E., Pitkow, J., Mackinlay, J., Pirolli, P., Gossweiler, R., and Card, S. (1998). Visualizing the Evolution of Web Ecologies. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 400–407.

Deshpande, M. and Karypis, G. (2004). Selective Markov Models for Predicting Web Page Accesses. *ACM Trans. Internet Technol.*, 4(2):163–184.

Gündüz, S. and Özsu, M. (2003). A Web Page Prediction Model Based on Click-Stream Tree Representation of User Behavior. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–540.

Jianyong, W., Jiawei, H., and Li, C. (2007). Frequent Closed Sequence Mining without Candidate Maintenance. *IEEE Transactions on Knowledge and Data Engineering*, 19(8):1042–1056.

Jin, X., Zhou, Y., and Mobasher, B. (2005). Task-oriented Web User Modeling for Recommendation. In *Proceedings of the 10th International Conference on User Modeling (UM)*, pages 109–118.

Lu, L., Dunham, M., and Meng, Y. (2005). Mining Significant Usage Patterns from Clickstream Data. In *7th International Workshop on Knowledge Discovery on the Web*, pages 1–17.

Nakagawa, M. and Mobasher, B. (2003). Impact of Site Characteristics on Recommendation Models Based On Association Rules and Sequental Patterns. In *Intelligent Techniques for Web Personalization*.

Needleman, S. and Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

Pitkow, J. and Pirolli, P. (1999). Mining Longest Repeating Subsequences to Predict World Wide Web Surfing. In *USITS'99: Proceedings of the 2nd conference on USENIX Symposium on Internet Technologies and Systems*, pages 139–150.

Schechter, S., Krishnan, M., and Smith, M. (1998). Using Path Profiles to Predict HTTP Requests. *Computer Networks and ISDN Systems*, 30(1-7):457–467.

Smith, T. and Waterman, M. (1981). Identification Of Common Molecular Subsequences.

Srivastava, J., Cooley, R., Deshpande, M., and Tan, P. (2000). Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. *SIGKDD Explorations Newsletter*, 1(2):12–23.

Tan, P. and Kumar, V. (2002). Discovery of Web Robot Sessions Based on their Navigational Patterns. *Data Mining Knowledge Discovery*, 6(1):9–35.

Viermetz, M., Stolz, C., Gedov, V., and Skubacz, M. (2006). Relevance and Impact of Tabbed Browsing Behavior on Web Usage Mining. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 262–269.

Weinreich, H., Obendorf, H., Herder, E., and Mayer, M. (2006). Off the Beaten Tracks: Exploring Three Aspects of Web Navigation. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 133–142.