2009 Ninth International Conference on Intelligent Systems Design and Applications

An Heuristic Approach to Page Recommendation in Web Usage Mining

Antonio Maratea, Alfredo Petrosino
Department of Applied Science
University of Naples Parthenope
I-80143 Naples, Italy
{antonio.maratea, alfredo.petrosino}@uniparthenope.it

***Abstract*—Personalized Web page recommendation is strongly limited by the nature of web logs, the intrinsic complexity of the problem and the tight efficiency requirements. When tackled by traditional Web Usage Mining techniques, due to the presence of an huge number of meaningful clusters and profiles for visitors of a typical highly rated website, the model-based or distance-based methods tend to make too strong and simplistic assumptions or, conversely, to become excessively complex and slow. In this paper, a heuristic " majority intelligence" strategy is designed, that easily adapts to changing navigational patterns, without the costly need to explicitly individuate them before navigation. The proposed approach mimics human behavior in an unknown environment in presence of many individuals acting in parallel and is able to predict with good accuracy and in real time the next page category visited by a user. The method has been tested on real data coming from users who visited a popular website of generic content. Average accuracy on test sets is good on a 17 class problem and, most remarkably, it remains stable as the web navigation goes on.**

***Index Terms*—Web Usage Mining, Web User Profiling, Web Personalization, Web Recommendation**

I. Introduction

Characterization of human behavior in a digital environment is one of the great challenges of computer science in the new century [2]. "Mining" the web will be essential in future not only to better understand human behavior (and hence human intelligence), but also to allow completely dynamic restructuring of contents and adaptive navigation of the Internet. The explosive growth of web user data in the form of log files has produced an increasing interest of researchers in the last decade and a young but rich literature on Web Usage Mining (see for reference [1], [3], [9]). As instance, personalized recommendation of products, documents, and collaborators has become an important way of meeting user needs in commerce, information provision, and community services, whether on the web, through mobile interfaces, or through traditional desktop interfaces. So, the analysis of web usage data may be useful at different levels and degrees to all categories of actors involved in any role into web usage (web site administrators, content providers, advertisement companies, common users, database and network administrators, etc.).

The "natural"framework in which Web Usage Mining has been studied widely is that of markovian models [12], mainly because navigated pages may be easily represented as a discrete sequence of symbols from a finite alphabet. Markovian models are also the core of predictive web prefetching algorithms (see for example [7], [8]), in which the forthcoming page accesses of a client are predicted on the base of its past accesses, to the purpose of improving cache effectiveness. Unfortunately, the underlying Markov assumption of a unique generative distribution turns out to be too strong and models of order bigger than 1 have a too high complexity [6]. Mixture of markovian models with a limited number of components have shown better performances [2], [5], however even a mixture of a few components is inadequate for the high heterogeneity of big data sets concerning human behavior. Conversely, distance based user profiling algorithms tend to be slow and to produce too fragmented results, as the number of significant profiles may easily be huge. Meta clustering algorithms have been used (for example in [11], [13]) to reduce this phenomenon, but the high number of clusters reflects actual richness of user interests and cannot be reduced significantly without loosing meaningful information. More canonical approaches based on association rules and more challenging machine learning techniques have also been proposed (see for example [1]).

With such an array of different perspectives, Web Usage Mining asks for an array of techniques that needs to be developed and tailored to each specific user requirement. There are anyway many common factors who make the solution of each problem challenging. In practice, available web user data have "always" the form of web logs (most likely because they are so common and easily obtainable), so in a broad sense Web Usage Mining becomes the problem of how to infer meaningful patterns from *log data*. The use of this kind of data implies per se a strong limitation. In practice, almost all Web Usage Mining techniques consider only server side logs, due to the problematic collection of information on the client side. Server logs usually contain data as the client IP address, the used browser, the used operating system, the requested page(s), the date and time of visit and similar. An incomplete list of limitations of server side logs and mining from logs in general follows:

a) IP addresses in log files are not reliable indicators of user identity, because proxies or ISP may mask the individual IP address;

b) the sequence of page requests written in the log is not a reliable indicator of the sequence of page actually visited, because some pages are cached by the browser and/or by the

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.252

proxy and may be navigated backward or forward without asking for a reload. For the same reason, any reading of the time spent on each page from the web log is unreliable;
c) it is difficult to isolate sessions, as the same IP address may correspond to different user in different times, or conversely different IP address may correspond to the same user connected at different times [15]. In literature a timeout of 30 minutes for isolating sessions is pretty common [10];
d) data from log files need to be converted in numerical form and cleaned before any processing [4], but there is not an universally accepted coding schema or cleaning rule. Data may be grouped for thematic categories or be detailed down to URL, time of visit may be reported or not, "short" sessions can be removed or not (often using crude cut off thresholds...) and so on.

Apart the previous list of what may even seem *technical* problems, many *semantical* problems affect analysis of web logs:
e) a user may have different interest in different visits to the same website or a number of parallel interests in the same visit (and behave like 2 or N others);
f) as a very crowded website may easily have millions of visits per day, the number of meaningful and "true" clusters of users is necessarily huge. People that visit a website may be classified in hundreds of thousands of categories/interests. The readability and hence the usefulness of clustering decreases exponentially when the number of clusters grows so much;
g) the complexity and the "thematic spread" of a website is another unconsidered factor that may multiply user profiles: a very specific website is likely to attract very specialized users that will likely have a limited number of navigation patterns, while a general website (or a hub website) that includes i.e. sport, news, music, technology will attract thousands of different people with heterogeneous interests and a hub website for example will be characterized by very short *relevant* visits;
h) auxiliary variables that could be used to further characterize users habits, like sex, age, education are not available in log files and the ones that are available, like time of visit or used O.S., are often not accounted for;
i) some visits may be performed by crawlers, robots, spiders or malicious software and not being performed by any human. These are difficult to isolate;
j) many websites have dynamically constructed pages that may update very fast ad whose log may become obsolete in a very short time;
k) navigation is already intrinsically biased by advertisement, website structure, presentation logic, marketing rules application;
l) the absence of auxiliary data makes the validation step of any clustering method extremely difficult, clusters may only be validated by human experts or through internal measure of goodness (see [10] for a possible approach).

In summary, we can say with confidence that the sources of technical and semantical variability in this problem are overwhelming and unlikely to be reduced unless other kind of data are used alternatively or in addiction to web logs. In the current scenario, the use of a heuristic is both viable and reasonable for at least a partial solution attempt.

The specific problem approached in this paper is how to predict the next page category visited by a user in a single session, given the list of his previously visited page categories. The granularity of the problem is at page category level, instead of single page level, as our focus is on the user behavior and not on the website management [2]. In [10] page categorization is suggested as a general pre processing step that helps reducing complexity and improve data management. The page categories may be manually given by a website administrator [2], [10] or may be built automatically (see for example [14], [16]). We propose a heuristic approach that mimics human behavior in an unknown environment crowded by other humans. For prediction, the proposed algorithm "borrows" the most voted next category from a sample of users with a similar navigation history. In this way a " majority intelligence" strategy is implemented, that easily adapts to changing navigation patterns without the need to explicitly individuate the patterns before computation.

Our prediction algorithm is supposed to work server side on a structured website. While navigation of a user goes on, relative log data can be accessed and converted into categories (in the simplest case it may suffice to access a lookup table). After that, the next page navigation category is predicted and suggested to the user. Within this predicted category, most popular contents may be proposed following any wished criteria.

Web sessionalization, that is the problem of how session are actually built starting from log data, is a challenging problem with a rich specific literature [19], [17], [18], [15] that will not be discussed further in this paper. In the dataset we used as benchmark [2], a time based heuristic for sessionalization is used.

The paper is organized as follows: in next section the proposed algorithm and the meaning of its parameters are explained. In section 3 the used data are briefly described and the results of the experiments summarized. Finally, in section 4 main conclusions are drawn and future work is outlined.

II. Proposed method

The purpose of the proposed method is to give an efficient prediction algorithm that may be used for online recommendations to users of a website. The granularity of the problem is at page category level, instead of single page level. An often underrated benefit of the use of page categories is the avoidance of problems due to pages updated very frequently and/or with a very short lifetime. The proposed method tries to mimic the human behavior in an unknown environment crowded by other humans: the visitor that has browsed L categories "asks" to a random sample of K other visitors that have exactly his same navigation history what category to go next and then uses a voting schema to take a decision. If no one is present with exactly his same navigation history, he "asks" to visitors that have exactly his same navigation history on the most recent $L-1$ browsed categories and so

on recursively, until he looks for visitors that browsed just his current category. In case of parity of votes, he randomly chooses between the ex aequo categories. Order of visits turned out to be very important for reliable prediction and, conversely, comparison of unordered lists of previously visited categories turned out not to be enough for a good prediction (see next section). Free parameters are the minimum and maximum size of support, $Kmin$ and $Kmax$, that is the minimum and maximum number of "opinions" to consider for validating voting and $MaxIter$, that is the maximum number of trial for an exact match in navigation history before resizing the querying sequence to the $L-1$ most recent categories. The proposed algorithm constructs dynamically and efficiently the pattern while navigation goes on. A flow chart schema is depicted in figure 1.

Input parameters of the algorithm are: the query sequence (suppose of length L), the reference (training) data, minimum and maximum size of support and maximum number of trials. Reference data may be grouped on the base of size t, thus avoiding rejection of too many samples because of incompatible length.

The first step of the algorithm is to extract randomly a sequence from the reference dataset compatible with the query sequence and suitable to give a prediction (both conditions imply that to be accepted, the extracted sequence must have at least length $L+1$) and to compare it element-wise with the query sequence: if their first L symbols match, the selected sequence becomes "trusted" and its $L+1^{th}$ symbol is took as a valid suggestion; if they don't match, even for a single symbol, the sequence is not "trusted" and it is rejected. The process is repeated until the maximum number of trials or the maximum size of support is not reached and when it proceeds, the vector of suggestions is incrementally built and kept in memory. When the cycle stops, no guarantee of having reached the minimum support is given, so this condition must be checked. If minimum support was not reached, instead of merely repeating the process, the search is done shortening the query sequence and the reference data so that their most remote pages are no more considered. To do so, a recursive call to the function is needed. The recursion stops when the query sequence has reached length one, that means only current page is searched in reference data and for a sequence to be trusted it is enough to have visited this same page once. The vector of suggestions may hence have different lengths (actually a different number of non zero elements) following the number of "supporters" of each query sequence. If there are in the data *at least* $Kmin$ sequences matching with the query sequence on *at least* the last most recently visited page, then the suggestions vector is guaranteed to have length between $Kmin$ and $Kmax$, otherwise it may be shorter or even be null. By convention we indicated with 0 the value corresponding to "no prediction available" . Finally, once the suggestion vector has been built, the next step is to compute voting and to choose a winner (this step can be performed only if the suggestions vector is not null). In the case that two or more "candidates" get the same score, a random choice between them is performed.

The computational complexity of the algorithm is strictly related to the value of parameters (or in other terms to sampling size). In the worst case of no reference sequence matching the query sequence complexity is $O(MaxIter \cdot L^2)$. To show how this value was obtained, conditions in figure 1 have been numbered from 1 to 4. In the worst case loop guarded by condition (1) always stops after $MaxIter$ iterations and in each cycle of loop (1), the loop guarded by condition (2) always stops after L iterations. So the computational cost of completing the two nested loops (1) and (2) is $O(MaxIter \cdot L)$. In the worst case the recursive condition (3) is always verified and so the two nested loops (1) and (2) are repeated L times while the query sequence shortens to length 1 (condition 4 is needed to stop recursion). Cost in the second iteration is hence $O(MaxIter \cdot (L-1))$; cost in the third iteration is $O(MaxIter \cdot (L-2))$ and so on. Total complexity is $O(MaxIter \cdot (L+1) \cdot L/2) \approx O(MaxIter \cdot L^2)$.

III. Data, experiments and results

Data used in the experiments are taken from a well-known dataset freely downloadable from the UCI KDD repository[1]. They come from users who visited the *msnbc* website on the whole day of september 28, 1999 and are already coded by category. Each category may include from 10 to 5000 different pages and is represented with a number between 1 and 17. No information on the time of stay on each page or on the total duration of the session is present. Length of session is measured in terms of number of different categories visited and goes from 1 to more than 500. There are nearby one million visits recorded. Each row represents a single visit and its average length is 5.7. The 17 categories are: *frontpage, news, tech, local, opinion, on-air, misc, weather, health, living, business, sports, summary, bbs (bulletin board service), travel, msn-news* and *msn-sports*.

To have reference sequences of the same length, we extracted a random sample of 10000 sequences from the *msnbc* dataset of length 2, another random sample of the same size of sequences of length 3 and so on up to length 6 (remember that the average length is 5.7). Each of these 5 blocks of data was randomly split in training set and test set, following the 0.7/0.3 proportion.

The used approach is supervised. To explain how we estimated accuracy, suppose we have an observed sequence of length 5, for example: (1, 5, 4, 2, 2). We can use its first 4 symbols as input of the algorithm and compute the predicted fifth symbol. As we know the true fifth symbol, we are able to evaluate reliably if they match. The estimation of accuracy (done with the above explained method) has been performed for each considered sequence length. The length of sequence has shown not to have a large impact on accuracy (as can be seen in Figure 2).

For each sequence of length L in the test set, its subsequence of length $L-1$ built with its first $L-1$ symbols was used

[1] http://kdd.ics.uci.edu/databases/msnbc/msnbc.data.html

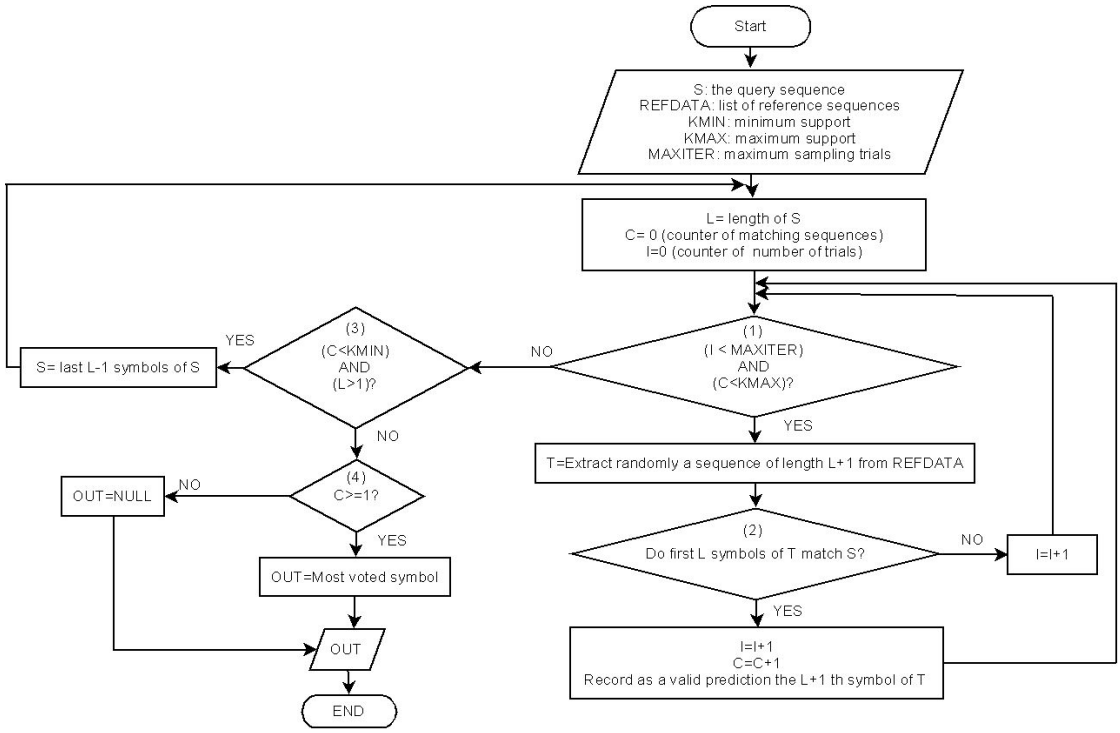

Fig. 1. The Trust-The-Unknown algorithm in a flow chart representation. OBJECTIVE: Given a sequence S of page categories already visited by a single client of length L and a collection of visits at least of length $L+1$ performed by other users as reference, to compute the most voted next page for the considered client.

as query sequence of the algorithm and fed together with the training set block of length L as input to the algorithm. The predicted L^{th} symbol computed by the algorithm on the training set was then compared with the known value to evaluate the accuracy. Mean values of accuracy on test sets are depicted in Figure 2.

Computation was done with the following parameter setting for $Kmin, Kmax$ and $NmaxTrial$ respectively: (10,20,60); (20,40,120); (40,80,240); (80,160,480).

The most remarkable result easily readable from figure 2 is that *ordering is essential for prediction accuracy* and that it is not enough to consider visits that just *include* the same categories: a sequence with the same symbols of the query sequence but in different order or with some of them repeated a different number of times should not be used to infer predictions, as it is highly unreliable. Another fact we can again read from figure 2, is that *the prediction accuracy remains quite stable as the sequence length increases*. This fact demonstrates the good adaptability of the method to changing conditions and its ability to capture changing navigation patterns in real time. Finally, the choice of parameters seems not to be critical, as average variation on accuracy on tested values is less than 2% once that a minimum support of at least 20 users has been reached.

The obtained absolute value of average accuracy is circa 65%. While not striking, it is surely a promising result considering the long list of intrinsic biases that affect log data analysis (see introduction). The proposed method is not model based, nor distance based, it is not a variation of clustering and not a form of classification. As the evaluation criteria of a solution is strictly connected to the nature of the used technique (validation of a clustering is completely different from validation of a classification...) many measures of quality that have been proposed in literature are almost technique specific. These measures are not comparable, unless a technique of the same family is used (i.e. clustering vs. clustering). While the *msnbc* data set has been widely used in literature, to best of our knowledge no other measure of accuracy comparable with our (built in the same way) has been published yet on these data that can be used as reference.

The only published techniques that use a notion of accuracy similar to our are in the context of page prefetching literature [7], [8]. In both the cited papers accuracy is defined as "the fraction of the prefetched requests offered to the client that were actually used" and this is similar to "the fraction of recommended pages actually visited". Even if we evaluated accuracy step by step as navigation goes on, while the above papers evaluate accuracy for various combinations of values of parameters, a *rough* comparison is possible. [7] proposes a comparison of different web prefetching techniques: the average value of accuracy among these techniques never goes over the 65% value. [8] uses simulations to study the effect of parameters on his prefetching model and again the obtained values of average accuracy are comparable with ours. Using

as bottom line the performance of a dumb classifier for a 17 class problem, the obtained value is ten times better.

IV. Conclusions

Web Log Mining is a very difficult task due to many intrinsic limitations of web logs and many uncontrollable sources of variation. Due to the high number of meaningful "user profiles"of a typical highly rated website, model or distance based method tend to make too strong and simplistic assumptions or to become excessively complex and slow. In this paper we proposed a heuristic algorithm, called Trust-The-Unknown, that mimics human behavior in an unknown environment. The method has three related free parameters, whose choice turned out not to be critical. Results on real data on prediction of the next page starting from the ordered sequence of previously visited pages on real data are encouraging and certainly deserve attention for future development. The average accuracy on test sets on a 17 class problem is consistent with existing literature and most remarkably *remains stable as navigation goes on*. Future work is about studying estimation strategies for parameters, extending suggestions to more than one category and trying possible extensions to log files with a different structure and granularity.

References

[1] P. Baldi, P. Frasconi, P. Smyth (2003) *Modeling the Internet and the Web*, Wiley Press, USA.

[2] I. Cadez, D. Heckerman, C. Meek, P. Smyth and S. White (2003) "Model based clustering and visualization of navigation patterns on a Web site", *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp.399-424.

[3] S. Chakrabarti (2003) *Mining the Web*, San Francisco, Morgan Kaufman.

[4] R. Cooley, B. Mobasher and J. Srivastava (1999) "Data Preparation for Mining World Wide Web Browsing Patterns", *J. Knowledge and Information Systems*, vol. 1, no. 1, pp. 5-32.

[5] M. Girolami and Ata Kabán (2004) "Simplicial Mixtures of Markov Chains: Distributed Modelling of Dynamic User Profiles", *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, David S. Touretzky, Sebastian Thrun, Lawrence K. Saul, Bernhard Schlkopf Eds, MIT Press, 2004, pp.9-6.

[6] S. Jespersen, T. B. Pedersen and J. Thorhauge (2003) "Evaluating the markov assumption for web usage mining, in *WIDM 03: Proceedings of the* 5^{th} *ACM international workshop on Web information and data management* , New Orleans, Louisiana, USA, November 7-8 2003.

[7] A. Nanopoulos, D. Katsaros and Y. Manolopoulos (2003) "A Data Mining Algorithm for Generalized Web Prefetching, IEEE Trans. on Knowl. and Data Eng. ,vol. 15, no. 5, pp. 1155-1169.

[8] T. Palpanas and A. Mendelzon (1999) "Web Prefetching Using Partial Match Prediction", in Proceedings of the 4^{th} International Web Caching Workshop, San Diego, California.

[9] R. Kosala, H. Blockeel (2000) "Web Mining Research: A Survey", In *SIGKDD Explorations*, ACM press, vol. 2, no.1, pp.1-15.

[10] G. Pallis, L. Angelis and A. Vakali (2006) "Validation and interpretation of Web users sessions clusters", *Information Processing and Management*, vol.43, no. 5, pp. 1348-1367.

[11] K. A. Smith and A. Ng (2003) "Web page clustering using a self-organizing map of user navigation patterns", *Decision Support Systems*, vol. 35, pp. 245-256.

[12] P. Smyth (1997) "Clustering sequences with hidden Markov models", in: M.C. Mozer, M.I. Jordan, T. Petsche (Eds.), MIT press, *Advances in Neural Information Processing Systems*, vol. 9, pp. 648-654.

[13] K. YongSeog (2007) "Weighted order-dependent clustering and visualization of web navigation patterns", *Decision Support Systems*, vol. 43, pp. 1630-1645.

[14] F. Yongjian, S. Kanwalpreet and S. Ming-Yi (1999) "Generalization-based approach to clustering of web usage sessions", *Lecture Notes In Computer Science*, vol. 1836, pp. 21 - 38.

[15] J. Zhang and A. A. Ghorbani (2004) "The reconstruction of user sessions from a server log using improved timeoriented heuristics", in *CNSR. IEEE Computer Society*, pp. 315-322.

[16] J. Zhu, J. Hong, J. G. Hughes (2004) "Mining conceptual link hierarchies from Web log files for adaptive Web site navigation, *ACM Transactions on Internet Technology*, vol. 4 , no. 2, pp. 185 208.

[17] A. A. A. Elkilany, I. Petrounias (2008) "biTemporal Session Reconstruction for Visited Sessions Retrieval" *in Proceedings of the* 12^{th} *International Conference Information Visualisation*, vol 4, pp. 341-346.

[18] M. Nadjarbashi-Noghani and A. A. Ghorbani (2004) "Improving the Referrer-Based Web Log Session Reconstruction", *in Proceedings of the Second Annual Conference on Communication Networks and Services Research*, pp.286-292.

[19] R. F. Dell, P. E. Romn and J. D. Velsquez (2008) "Web User Session Reconstruction Using Integer Programming", *in Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 1, pp.385-388.

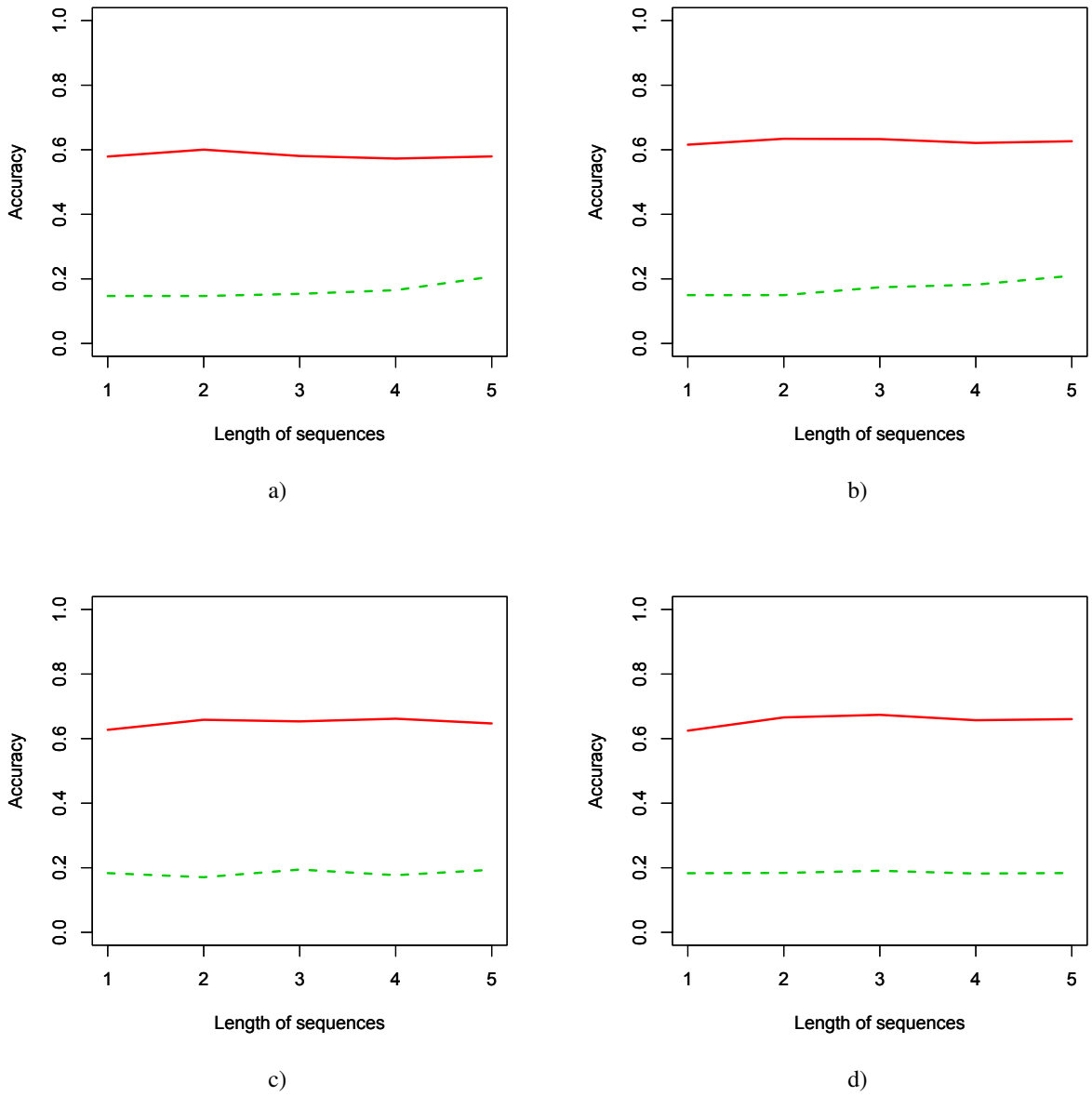

Fig. 2. Average accuracy on test sets for different lengths of query sequences. The x axis shows the length of query sequence (that is up to 5 in order to evaluate prediction accuracy with the true symbol) while the y axis shows the average accuracy of the prediction of the $(x+1)^{th}$ symbol. a) $Kmin = 10, Kmax = 20, MaxIter = 60$; b) $Kmin = 20, Kmax = 40, MaxIter = 120$; c) $Kmin = 40, Kmax = 80, MaxIter = 240$; d) $Kmin = 80, Kmax = 160, MaxIter = 480$. Prediction with ordered sequences (solid line) and prediction with unordered sequences (dashed line).