Mining Click-stream Data With Statistical and Rule-based Methods

Martin Labský, Vladimír Laš, Petr Berka

Department of Information and Knowledge Engineering,
University of Economics, Prague, W. Churchill Sq. 4, 130 67 Praha 3, Czech Republic
{labsky,lasv,berka}@vse.cz

Abstract. We present an analysis of the click-stream data provided for the ECML/PKDD data mining challenge. We primarily focus on predicting the next page that will be visited by a user based on a history of visited pages. We compare results of one statistical and two rule-based methods, and discuss interesting patterns that appear in the data.

1 Introduction

According to the W3C Web Characterisation Activity, click-stream is a sequential series of page view requests, each displayed by user's browser at one time. The ECML/PKDD challenge click-stream data contains *server sessions* for an electronics Internet shop, i.e. click-streams for single users within the shop's site.

We can categorise web mining into three areas: web content mining, web structure mining and web usage mining [16]. Web usage mining mines the data derived from interactions of users while browsing the web. Web usage data includes the data from web server access logs, proxy server logs, browser logs, user profiles, registration data, cookies, user queries etc. A web server log is an important source for performing web usage mining because it explicitly records the browsing behaviour of site visitors. The typical problem (solved in the data preprocessing step) is thus distinguishing among unique users, server sessions episodes etc.

Web usage mining focuses on techniques that could predict user behaviour while the user interacts with the web. The applications of web usage mining could be classified into two main categories: (1) learning user profile or user modelling, and (2) learning user navigation patterns [10]. The methods used for web usage mining are (descriptive) statistical analysis, association rules (to relate pages that are often referenced together), clustering (to build usage clusters or page clusters), classification (to create user profiles), sequential pattern discovery (to find inter-session patterns such that the presence of a set of page views is followed by another set of page views in a time-ordered set of episodes), or dependency modelling (to capture significant dependencies among various variables in the web domain) [13]. Some systems already exist in this area: WebSIFT (uses clustering, statistical analysis and association rules) [4], WUM (looks for association rules using an extended version of SQL) [12], or WebLogMiner (combines OLAP

and KDD) [17]. An overview of web page recommendation systems is presented in [7].

The algorithms described in this paper are motivated by two goals – (1) to predict user's behaviour (i.e. to recommend the next page of interest given previously visited pages), and (2) to find interesting patterns in the visited page sequences. In the following we describe predictors of the next page type visited (e.g. product detail, FAQ, advice) and of the next product type of interest (e.g. digital cameras or zoom lenses). Both predictors can be used by web servers to recommend to users where they could go next. For practical purposes, such predictors might produce several recommendations so that the user spots an interesting page with greater probability. On the other hand, interesting patterns identified in page sequences provide important information both to webmasters and marketing specialists – e.g. about two products often bought together, or about a product detail page often being followed by a visit to a FAQ page. We present two types of algorithms – one statistical, which is appropriate for the next page prediction task, and two rule-based, which are suitable for both page prediction and pattern discovery.

In section 2 we list basic statistics of the data. Section 3 presents results of a statistical Markov N-gram page predictor. Section 4 describes two rule-based approaches, based on a set covering algorithm and a compositional algorithm. Section 5 compares the above and sections 6 and 7 conclude with directions for future research.

2 Click-stream Data

The log file (about 3 millions of records – the traffic of 24 days) contained the usual information: time, IP address, page request URL and referer. In addition to this, the log data contained also a generated session ID so the identification of users was relatively easy – we treated each sequence of pages with the same ID as one session. The whole dataset consists of $522,410$ sessions, of which $318,523$ only contain a single page visit. In the following, we will reduce our dataset to the $203,887$ sessions that contain at least two page visits. The average length of these sessions was 16; the median was 8 and modus 2. A session length histogram is shown in Fig. 1.

For evaluation purposes, we split the $203,887$ sessions into three sets: a training set (the first $100k$ sessions[1]), a test set (the second $60k$ sessions) and a held-out set (the remaining $43,887$ sessions).

We identified two interesting types of information in the click-streams: sequences of page types (e.g. detail of a product, shopping cart, product comparison), and sequences of product categories (both the 'list' and 'kategorie' tables combined). For page types, we appended an extra 'end' page to the end of each page sequence, since it seemed important to model which page sequences lead to leaving the web. For product categories, no 'end' pages were appended.

[1] Sessions were sorted by session start time.

Based on the 'list' and 'kategorie' tables, we defined 32 new product categories that grouped similar entries from both tables. Two mapping functions $f1, f2$ were defined to convert the 'list' and 'kategorie' tables into the new categories ($f1 : list[64] \rightarrow categories[32]$; $f2 : kategorie[172] \rightarrow categories[32]$). Analysed product sequences only contained the new categories, and all pages that did not contain product category information were removed from product sequences.

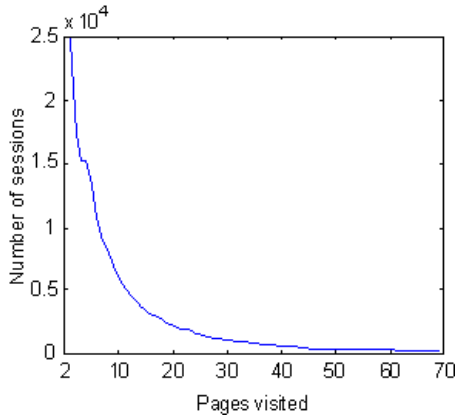

Fig. 1. Session length histogram

3 Markov N-gram Predictor

Statistical approaches to modelling sequences are widely used in areas such as speech recognition, natural language processing or bio-informatics. As our first approach, we trained various *Markov N-gram models* (N-gram models for short) from click-stream sequences. Using these models, we predicted the most likely page following after each history in test data. A model similar to our implementation is described in [6].

3.1 N-gram model

An N-gram model models the probability of an observed page sequence $A_1 A_2 \ldots A_n$ as:

$$P(A_1 A_2 \ldots A_n) = \Pi_{i=1}^{n} P(A_i | A_{i-k} \ldots A_{i-1}) \quad (1)$$

where k is the length of history taken into account. A model with k-token histories is called a $(k+1)$-gram model and obeys the Markov property of limited memory and stationarity [8].

An important decision is how to model the probability $P(A_i|A_{i-k}\dots A_{i-1})$. Based on counts observed in training data, a k-gram probability P_k only conditioned on $(k-1)$-length histories can be estimated as

$$P_k(A_i = c|A_{i-k+1} = a\dots A_{i-1} = b) = \frac{n(a\dots bc)}{n(a\dots b)} \tag{2}$$

where $n(xy)$ is the count of page sequence xy observed in training data. However, choosing just one fixed history length is problematic since long histories suffer from data sparsity and short histories may not contain sufficient information about the next page. It is therefore a common approach to interpolate P_k with lower-order probabilities (i.e. those conditioned on shorter histories). The interpolated (smoothed) k-gram distribution is thus given as a weighted sum of the fixed history length distributions:

$$\begin{aligned} P(A_i|A_{i-k+1}\dots A_{i-1}) = \lambda_0 P_0(A_i) + \lambda_1 P_1(A_i) + \lambda_2 P_2(A_i|A_{i-1}) + \\ + \dots + \lambda_k P_k(A_i|A_{i-k+1}\dots A_{i-1}) \end{aligned} \tag{3}$$

where

$$\sum_{i=0}^{k} \lambda_i = 1, \forall_{i=0}^{k} \lambda_i \geq 0, \lambda_i \leq 1 \tag{4}$$

In Eq. 3, P_0 is a uniform distribution over all existing pages, P_1 is the unigram distribution only based on page frequencies without taking history into account, and the remaining P_i distributions are conditioned on histories of length $i-1$.

The weights λ_i are best determined by an unsupervised EM algorithm [8], which iteratively re-estimates weights using held-out data until convergence. First, all the components P_i $(i = 1\dots k)$ are computed from counts observed in the *training dataset* according to Eq 2. Second, weights are set to some initial non-zero (we chose uniform) values. Third, the probability of a *held-out dataset* is computed using the interpolated distribution in Eq. 3 with the working weights. For each addend in Eq. 3, its share on the total held-out data probability is collected. After running through all held-out data, the new weights are obtained by normalising the shares of each addend. Step three is repeated until none of the λ_i weights changes significantly.

3.2 N-gram results

We trained N-gram models to predict the next page type, and the next product category a user might be interested in.

All N-gram models are only trained using the training data set, and smoothed using the held-out data set. We evaluate the N-gram predictors by comparing the real item in test data with the most likely item that would follow the observed history according to the model. Prediction accuracy is given as the number of correct predictions over all predictions. The best results are compared to rule-based methods in Table 1. The table also includes results achieved on training

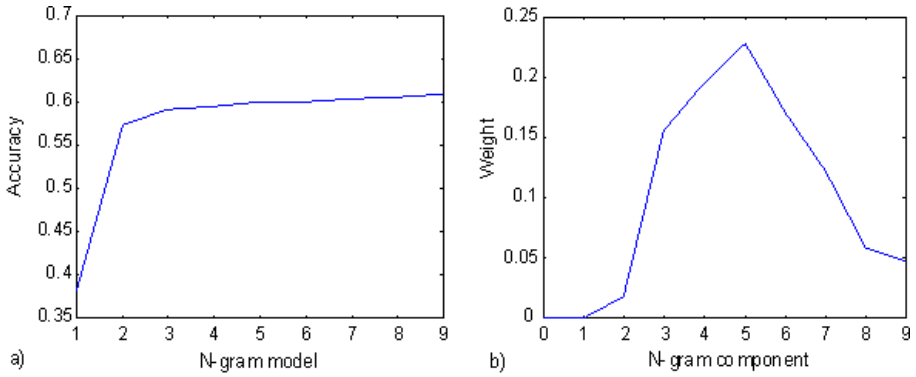

Fig. 2. a. N-gram accuracy for page types **b.** Weights of 9-gram components

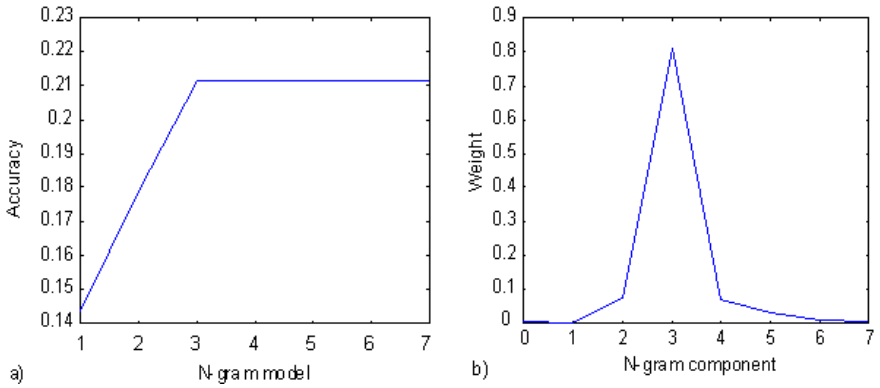

Fig. 3. a. N-gram accuracy for product types **b.** Weights of 7-gram components

data; in these cases, N-gram model weights were estimated on training data as well, which caused the highest-order component to always have a weight of 1.

Accuracy for page types reached 0.61 and is reported in detail in Fig. 2a for N-gram models with N ranging from 1 (a unigram model only smoothed with uniform distribution) to 9 (a 9-gram model smoothed with 9 lower order distributions). We observe that accuracy climbs significantly until the trigram model, thus the chosen page type depends mostly on the two preceding pages. It is also interesting to note the smoothing weights for the 9-gram model in Fig. 2b. Here, the pentagram component's weight reaches a maximum of 23% although its contribution to the overall accuracy is small. We assume this is due to abundance of data – data sparseness is not significant yet for pentagrams and thus we can get large weights for high-order probabilities.

Accuracy for product types reached 0.21 for a heptagram model; climbing from 0.14 for a unigram model, as seen in Fig. 3a. Here, we observe a significant increase in accuracy until the trigram model, where further increases stop. This

is confirmed by the superior trigram component weight in Fig. 3b. We can thus conclude that the next product type of interest can be reasonably predicted based just on two preceding products.

4 Rule-based Predictors

4.1 Classical rule learning algorithms

Decision rules in the form

$$Ant \Rightarrow Class$$

where Ant (antecedent, condition) is a conjunction of values of input attributes (called categories or selectors) and $Class$ is a category of class attribute C, are one of most popular formalisms how to express classification models learned from data. The commonly used approach to learning decision rules is the *set covering approach* also called "separate and conquer". The basic idea of this approach is to create a rule that covers some examples of a given class and remove these examples form the training set. This is repeated for all examples not covered so far. There are two basic ways how to create a single rule (step 1 of the algorithm):

1. by rule generalisation, i.e. by removing categories from antecedent of a potential rule (starting from a rule with categories of all input attributes in the antecedent) - this method is used in the AQ algorithms by Michalski (see e.g. [11]).
2. by rule specialisation, i.e. by adding categories to the antecedent of a potential rule (starting from a rule with empty antecedent) – this method is used e.g. in CN2 [5] or CN4 [3].

The other way how to create decision rules is the *compositional approach*. In this approach the covered examples are not removed during learning, so an example can be covered with more rules. Thus more rules can be used during classification. In compositional approach, all applicable rules are used and their particular contributions to classification are combined into the final decision. To do this, some numerical value is usually added to the rule, the simplest one is the rule confidence (also called validity) defined as $n(Ant \wedge Class)/n(Ant)$, where $n(Ant \wedge Class)$ is the number of examples that match both Ant and $Class$ and $n(Ant)$ is the number of examples that match Ant in the data.

4.2 Rule learning algorithms for click-streams

The main difference to conventional rule learning algorithms is due to the fact that instead of unordered set of categories we deal with an ordered sequence of pages. So we are looking for rules in the form

$$Ant \Rightarrow page(p)$$

where Ant is a sequence of pages, $page$ is a page view that directly follows the sequence Ant, and p is the validity of the rule

$$p = \frac{n(Ant//page)}{n(Ant)}.$$

In the formula above we denote the number of occurrences of a sequence in the data by $n(sequence)$ and a concatenation of two sequences $s1$ and $s2$ by $s1//s2$.

We propose two rule learning algorithms for click-streams: a set covering and a compositional one [2]. We follow the rule learning approach based on rule specialisation in our algorithms as well. As we assume that most relevant for prediction of occurrence of a page are pages that are closest to this page, the specialisation of the rule $Ant \Rightarrow page$ is done by adding a new page to the beginning of the sequence Ant. Analogously, a generalisation of the rule $Ant \Rightarrow page$ is done by removing a page from the beginning of the sequence Ant.

The main idea of our *set covering* algorithm is to add (for a particular page to be predicted) rules of growing length of Ant. We check each rule against its *generalisation* created so far. Adding a new rule to the model is determined by χ^2 test that compares the validity of these two rules. If the rule in question is added to the model, its *generalisation is updated* by re-computing the validity by ignoring (removing) sequences that are covered by the newly added rule (Fig. 4).

The main idea of our *compositional* algorithm (Fig. 5) is again to add (for a particular page to be predicted) rules of growing length of Ant. We check each rule against the *results of classification* done by all rules created so far. Adding a new rule to the model is determined by χ^2 test that compares the validity of the rule to be added with the weight of class (predicted page) inferred during classification for a sequence Ant. The weight of a class is computed according to the formula

$$w_1 \oplus w_2 = \frac{w_1 \times w_2}{w_1 \times w_2 + (1 - w_1) \times (1 - w_2)}. \tag{5}$$

4.3 Rule learning algorithm results

In the first set of experiments we were looking for rules that can be interpreted as interesting by the data providers. We identified as interesting e.g. the rules

```
dp, sb -> sb (Ant: 5174; AntSuc: 4801; P: 93%)
 ct -> end (Ant: 5502; AntSuc: 1759; P: 0.32)
 faq -> help (Ant: 594; AntSuc: 127; P: 0.21).
```

for the sequences concerning page types. In the listing above, `ct` stands for "contact", `Ant` stands for $||Ant||$ and `AntSuc` stands for $||Ant//Suc||$). Among rules concerning product categories we found e.g.

```
loud-speakers -> video + DVD (Ant: 14840, AntSuc: 3785, P: 0.26)
   data cables -> telephones (Ant: 2560, AntSuc: 565, P: 0.22)
 PC peripheries -> telephones (Ant: 8671, AntSuc: 1823, P: 0.21)
```

Initialisation
for each page page occurring in the data
1. compute its relative frequency in the data as P = (no. of occurrences of *page* in the input episodes)/(no. of all input episodes)
2. if $P \geq n_{min}$
   2.1 add $default \Rightarrow page$ into the list of rules $Rules$
   2.2 add *page* into list of pages $Pages$
   2.3 add $default \Rightarrow page$ into list of implications $Impl$

Main loop
while Impl not empty do
1. take first rule $Ant \Rightarrow page$ from $Impl$
2. if length of $Ant < l_{max}$ then
   2.1 for each page *pp* from $Pages$
      2.1.1 find the most specific generalisation of the rule $pp//Ant \Rightarrow page$ in $Rules$ (denote it $AntX \Rightarrow page$)
      2.1.2 compare (using chi2 test) the validity of rules $pp//Ant \Rightarrow page$ and $AntX \Rightarrow page$
   2.2 from all created rules $pp//Ant \Rightarrow page$ select the one with the most significant difference in validity (denote this rule $pp_{best}//Ant \Rightarrow page$)
   2.3 if $pp_{best}//Ant \Rightarrow page$ significantly at a given significance level differs from $AntX \Rightarrow page$ then
      2.3.1 add rule $pp_{best}//Ant \Rightarrow page$ to $Rules$ and $Impl$
      2.3.2 re-compute the validity of rule $AntX \Rightarrow page$ by taking into account only episodes containing $AntX$ and not containing Ant
      2.3.3 recursively update Rules (i.e. find the most specific generalisation of $AntX \Rightarrow page$, compare this generalisation with $AntX \Rightarrow page$, remove $AntX \Rightarrow page$ from $Rules$ if the difference is not significant etc.)
3. remove $Ant \Rightarrow page$ from $Impl$

Fig. 4. The set covering rule learning algorithm for click-stream analysis

The obtained rule sets can directly be used to predict the behaviour of a user. So e.g. for a sequence of pages `dp, sb` the system will predict `sb` as the next page, and for the sequence `loud-speakers` the system will predict `video + DVD`.

In the second set of experiments we were interested in classification accuracy of our rule sets. We have run our algorithms repeatedly for both page sequences (first set of experiments) and product sequences (second set of experiments). When looking at the differences between the set covering and compositional algorithms, we can observe different trade offs between comprehensibility and accuracy: the set covering algorithm offers higher accuracy but lower comprehensibility (larger number of rules) than the compositional algorithm (see Tab. 1). The default accuracy refers to the accuracy of the "zero rule" model, that always predicts the most frequent page. In all experiments we exceeded this "base

Initialisation
for each page page occurring in the data
1. compute its relative frequency in the data as P = (no. of occurrences of *page* in the input episodes)/(no. of all input episodes)
2. if $P \geq n_{min}$
   2.1 add $default \Rightarrow page$ into the list of rules $Rules$
   2.2 add *page* into list of pages $Pages$
   2.3 add $default \Rightarrow page$ into list of implications $Impl$

Main loop
while Impl not empty do
1. take first rule $Ant \Rightarrow page$ from $Impl$
2. if length of $Ant < l_{max}$ then
   2.1 for each page pp from $Pages$
      2.1.1 create the rule $pp//Ant \Rightarrow page$ and compute its validity p
      2.1.2 from all rules in $Rules$ that cover the sequence $pp//Ant$ compute the resulting weight for page as $w^{\oplus}(pp//Ant)$
      2.1.3 compare (using chi2 test) p and $w^{\oplus}(pp//Ant)$
      2.1.4 if p significantly differs from $w^{\oplus}(pp//Ant)$ then
         2.1.4.1 compute w from the formula $w \oplus w^{\oplus}(ppAnt) = p$
         2.1.4.2 add rule $pp//Ant \Rightarrow page(w)$ to $Rules$ and $Impl$
      2.1.5 add $pp//Ant \Rightarrow page(w)$ to $Impl$
   2.2 remove $Ant \Rightarrow page$ from $Impl$

Fig. 5. The compositional rule learning algorithm for click-stream analysis

line". Since both rule-based methods make no use of held-out data, this data was appended to the training data in all cases.

5 A Comparison of N-gram and Rule-based Models

In Table 1, accuracy is computed as the number of correct guesses according to the model and to the observed history, divided by the total number of guesses (all pages from the evaluated data). For the N-gram model, the predicted page is the one with the highest conditional probability given an observed history of pages. For both rule based methods, the predicted page is the page for which the combined weight of all applicable rules (given history) is maximal.

We observe that for both page and product types, the accuracy of the N-gram models on test data is comparable to the set covering algorithm. For page types, the best prediction accuracy of 0.61 was reached by a 9-gram model, while the set covering algorithm achieved the highest accuracy of 0.24 for product types. When applied as a page recommendation system, both predictors should yield more than one recommendation about the next page to let the user choose from several relevant pages. In terms of performance on training data, the 9-gram model achieves 0.67 accuracy, which is however caused by over-fitting the training data. On the other hand, both rule-based algorithms seem to be resistant to over-

Table 1. Empirical comparison of algorithms

page sequences	default	N-gram[a]	set covering		compositional	
	accuracy	accuracy	no. rules	accuracy	no.rules	accuracy
test data	0.40	**0.61**	1088	0.59	371	0.49
training data	0.40	**0.67**	1088	0.60	371	0.50
product sequences						
test data	0.14	0.21	2176	**0.24**	903	0.19
training data	0.14	**0.24**	2176	0.23	903	0.18

[a] 9-gram and 7-gram results are reported for page and product sequences, respectively.

fitting, surprisingly for product sequences they achieve slightly better results on test data than on the training set.

To compare N-gram and both rule based methods, we may view N-gram models as exhaustive sets of weighted rules. Each non-zero probability $P_i(c|a \dots b)$ from Eq. 2, weighted by the corresponding weight λ_i, can be treated as confidence of a corresponding rule $a \dots b \Rightarrow c$. For a particular history $a \dots b$, the weighted confidences of relevant rules are summed to produce a probability for each possible predicted class. This similarity in learnt models seems to be the reason why the set-covering algorithm with large amounts of rules performed comparably to the N-gram models.

Viewing N-gram models as sets of rules, there are however several major distinctions from the rule-based algorithms. First, the number of N-gram "rules" is exhaustive, although we could e.g. remove all "rules" conditioned on histories having count less than a chosen threshold. Second, the contributions of N-gram "rules" are based on confidence (as for the set covering algorithm), however they are further weighted by a constant factor that expresses the reliability of all rules with a certain length of antecedent. To relax this constraint, N-gram weights could be alternatively specified for intervals of history frequencies (referred to as bucketed smoothing) instead of history lengths; in this case frequent histories would receive higher weights as they are more reliable. Yet another difference lies in the method how contributions of multiple rules are combined. In the compositional approach, Eq. 5 is used, whereas a weighted sum in Eq. 3 is used for the N-gram model to yield probability. Last of all, unlike the set covering algorithm, in the N-gram case multiple rules are used during a single classification, as in the compositional approach.

When the analysis goal is user's understanding of learnt models (rather than prediction), a method which learns few comprehensive rules is generally preferable. In our case, this is the compositional approach. The learnt weights of N-gram model components also seem to contain useful information on how long histories still influence user's behaviour.

6 Future Work

For the N-gram predictor, we plan to implement other smoothing methods, starting with bucketed smoothing based on history frequencies. For the rule-based models, we will analyse how rule learning parameters impact accuracy and compare our algorithms with state-of-the-art methods. Another experiment we would like to perform on the described dataset is predicting the next page based on a richer set of features observed in history. These features would include page types, visited product types, product makes, session length, and the time spent on these pages. A suitable prediction algorithm that would benefit from a large feature set could be a Maximum Entropy Model, which was successfully applied in a web recommendation system described in [9].

7 Conclusion

We presented and compared two approaches for clickstream data analysis – one statistical and two rule-based algorithms. Using these algorithms, we predicted the next page type visited and the next product type of interest. The statistical N-gram algorithm and the set-covering rule-based algorithm achieved comparable prediction accuracies for both page and product types. On the other hand, the compositional rule-based algorithm, which was inferior in terms of prediction accuracy, proved to be suitable for discovering interesting patterns in page sequences. The described algorithms can be applied by web servers to recommend relevant pages to their users, and to identify interesting patterns in their log files.

8 Acknowledgements

The research is partially supported by grant no.201/05/0325 of the Grant Agency of the Czech Republic and by grant no.17/04 of the Internal Grant Agency of the University of Economics.

References

1. Berka P., Ivánek, J.: Automated knowledge acquisition for PROSPECTOR-like expert systems. In: Proc. ECML'94, LNAI 784, Springer 1994, 339-342.
2. Berka P., Laš, V., Kočka T.: Rule induction for click-stream analysis: set covering and compositional approach. In: Proc. IIPMW 2005, Springer 2005, 13-22.
3. Bruha I., Kočková S.: A support for decision making: Cost-sensitive learning system. Artificial Intelligence in Medicine, 6 (1994), 67-82.
4. Cooley R., Tan P.N., Srivastava,J.: Discovery of interesting usage patterns from web data. Tech. Rep. TR 99-022, Univ. of Minnesota, 1999.
5. Clark P., Niblett T.: The CN2 induction algorithm. Machine Learning, 3 (1989), 261-283.

6. Deshpande M., Karypis G.: Selective Markov Models for Predicting Web-Page Accesses. Technical Report 56, University of Minnesota, 2000.
7. Gündüz S., Özsu M.T.: Recommendation Models for User Accesses to Web Pages. In: Proc. ICANN 2003.
8. Jelinek F.: Statistical Methods for Speech Recognition. The MIT Press 1998, 0-262-10066-5.
9. Jin X., Mobasher B., Zhou Y.: A Web Recommendation System Based on Maximum Entropy. In: Proc. IEEE International Conference on Information Technology Coding and Computing, Las Vegas, 2005.
10. Kosala R., Blockeel H.: Web Mining Research: A Survey. In: SIGKDD Explorations, Vol. 2 Issue 1, 2000.
11. Michalski R.S.: On the Quasi-minimal solution of the general covering problem. In: Proc. 5th Int. Symposium on Information Processing FCIP69, Bled, 1969, 125-128.
12. Spiliopoulou M., Faulstich L.: WUM: A tool for web utilization analysis. In: Proc. EDBT Workshop WebDB98, Springer LNCS 1590, 1999.
13. Srivastava J., Cooley R., Deshpande M., Tan P.N.: Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. In: SIGKDD Explorations, Vol. 1 Issue 2, 2000.
14. Witten I. H., Frank E.: Generating Accurate Rule Sets Without Global Optimization. In: Proc. of the 15th Int. Conference on Machine Learning, Morgan Kaufmann, CA, 1998.
15. Witten I. H., Frank E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann 1999, 1-55860-552-5.
16. Zaiane O., Han J.: WebML: Querying the World-Wide Web for resources and knowledge. In: Workshop on Web Information and Data Management WIDM98, Bethesda, 1998, 9-12.
17. Zaiane O., Xin M., Han J.: Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In: Advances in Digital Libraries, 1998.