Rule induction for click-stream analysis: set covering and compositional approach

Petr Berka,Vladimír Laš, Tomáš Kočka,

FIS, VŠE nám. W. Churchilla 4, 130 67, Praha

berka@vse.cz, vladalas@hotmail.com, kocka@vse.cz

Abstract: We present a set covering algorithm and a compositional algorithm to describe sequences of www pages visits in click-stream data. The set covering algorithm utilizes the approach of rule specialization like the well known CN2 algorithm, the compositional algorithm is based on our original KEX algorithm, however both algorithms deal with sequences of events (visited pages) instead of sets of attribute-value pairs. The learned rules can be used to predict next page to be viewed by a user or to describe the most typical paths of www pages visitors and the dependencies among the www pages. We have successfully used both algorithms on real data from an internet shop and we mined useful information from the data.

Keywords: click-stream, decision rules, web usage mining

1 Introduction

According to the W3C Web Characterization Activity, click-stream is a sequential series of page view (displays on user's browser at one time) requests. A user session is the click-stream of page views for a single user across the entire web, a server session is a click-stream in a user session for a particular web site. A set of server sessions (visits) is the necessary input for web usage mining tools.

We can categorize web mining into three areas: web content mining, web structure mining and web usage mining [9]. Web usage mining mines the data derived from interactions of the users while browsing the web. The 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 behavior 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 behavior 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 modeling, and (2) learning user navigation patterns [5]. 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 modeling (to capture significant dependencies among various variables in the web domain) [8]. Some systems has already been developed for this area: WebSIFT (that uses clustering, statistical analysis or association rules) [4], WUM (that looks for association rules using some extension of SQL) [7], or WebLogMiner (that combines OLAP and KDD) [10].

The algorithms described in this paper contribute to the area of sequential pattern discovery by learning decision rules to predict next page the user will visit based on his session history (pages visited just before). We describe the algorithms in section 2 and give some experimental results obtained on real web log data of an internet shop in section 3.

2 Learning Rules

2.1 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 as shown in fig. 1. There are two basic ways how to create a single rule (step 1 of the algorithm):

1. by rule generalization, 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. [6]).
2. by rule specialization, 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 [3] or CN4 [2].

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 *And* and Class and $n(Ant)$ is the number of examples that match *Ant* in the data. Fig. 2 shows an simplified version of the KEX algorithm, we developed in the 90th; this system deploys the compositional approach in a generate-test cycle and uses weight *w* (based on validity) to express uncertainty of a rule [1]. Let us mention, that in some broader sense, naive bayesian classifier follows the compositional approach as well.

Set covering algorithm

1. find a rule, that covers some positive examples and no negative example of a given class (concept),
2. remove covered examples from the training set D_{TR},
3. if D_{TR} contains some positive examples (not covered so far) goto 1 else end.

Fig. 1. Set covering algorithm for two class problems

Compositional algorithm

1. add empty rule to the rule set *KB*
2. repeat
   - 2.1. find by rule specialization a rule $Ant \Rightarrow Class$ that fulfils the user given criteria on lengths and validity,
   - 2.2. if this rule significantly improves the set of rules *KB* build so far (we test using the χ^2 test the difference between the rule validity and the result of classification of an example covered by *Ant*) then add the rule to *KB*.

Fig. 2. Simplified version of the KEX rule learning algorithm

2.1 Rule learning algorithms for click-streams

We follow the rule learning approach based on rule specialization in our algorithm as well. 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 page views (pages for short). 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*
p is the validity of the rule, i.e. $p = \frac{n(Ant//page)}{n(Ant)}$.

In the formula above we denote the number the occurrences of a *sequence* in the data by *n*(*sequence*) and a concatenation of two sequences *s*1 and *s*2 by *s*1//*s*2.

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 thus create rules by rule specialization. We check each rule against **its generalization** 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 **generalization is updated** by re-computing the validity by ignoring (removing) sequences that are covered by the newly added rule.

The main idea of our **compositional** algorithm is to add (for a particular page to be predicted) rules of growing length of *Ant* – we thus create rules by rule specialization. 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

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

As we assume that most relevant for prediction of occurrence of *page* are pages that are closest to *page*, the specialization of the rule $Ant \Rightarrow page$ is done by adding a new page to the beginning of the sequence *Ant*. Analogously, a generalization of the rule $Ant \Rightarrow page$ is done by removing a page from the beginning of the sequence *Ant*.

The set covering algorithm is shown in Fig. 3, the compositional algorithm is shown in Fig. 4. Both algorithms share the user given input parameters l_{max} (max. length of the sequence *Ant*), n_{min} (min. relative frequency of a page – this parameter set to zero will enable to create a rule that page Y never follows after page X). The parameter l_{max} is also used for data preprocessing; we transform each server session[1] of arbitrary length into a set of episodes of length $l_{max} + 1$ using a sliding window. So e.g. the two sessions

A,B,E
A,B,C,E,D

will be for $l_{max} = 2$ transformed into following set of episodes

Ø,Ø,A
Ø,A,B
A,B,E
Ø,Ø,A
Ø,A,B
A,B,C
B,C,E
C,E,D

The system is implemented in the Borland's Delphi (version 7.0) for Windows (W-98 – W-XP). The minimal configuration of the computer is not set; it's helpful to use computers with processor at least 600 MHz for more extensive analysis. The system doesn't use more than quadruple of the size of an input file in the memory.

We have used the object model of programming whereas each class is stored in particular unit. It's provided lucidity and easy service of the code. The algorithm for searching rules is implemented in particular unit, so others algorithms could be easily added to the program (the source code had to be recompiled because the program is represented by 1 file after compilation).

[1] Recall that a server session corresponds to one visit of one user on the analyzed web site.

Initialization
for each page *page* occurring in the data
1. compute its relative frequency in the data as

$$P = \frac{\text{\# of occurrence of } page \text{ in the input episodes}}{\text{\# of all input episodes}}$$

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 generalization 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 generalization of *AntX* $\Rightarrow$ *page* , compare this generalization 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. 3. The set covering rule learning algorithm for click-stream analysis

The algorithm for searching rules itself is divided to two parts (as you could see above) – the initialization and the main loop. The main loop is implemented as a recursive function that is called for particular rules. At the beginning, the function generates specializations, after it modifies frequency of actual rule and to the end it evaluates the chi-square test for actual rule.

The main loop is called from the initialization with the subsidiary rule * => * which assures that all rules *default* => *page* will be added to the result. The algorithm for searching rules runs in second thread, so found rules could be showed immediately in the core thread of the application - user could stop the algorithm when its continuation isn't needful. The user could restrict the space of searching by set rules that will not be extended more. Pages that don't have required relative frequency are added to a set OTHERS and the system works with this set as with single page.

Found rules could be saved to the file (and loaded back) or exported to Excel. The application enables also to predict a next page in a sequence.

Initialization

for each page *page* occurring in the data
1. compute its relative frequency in the data as

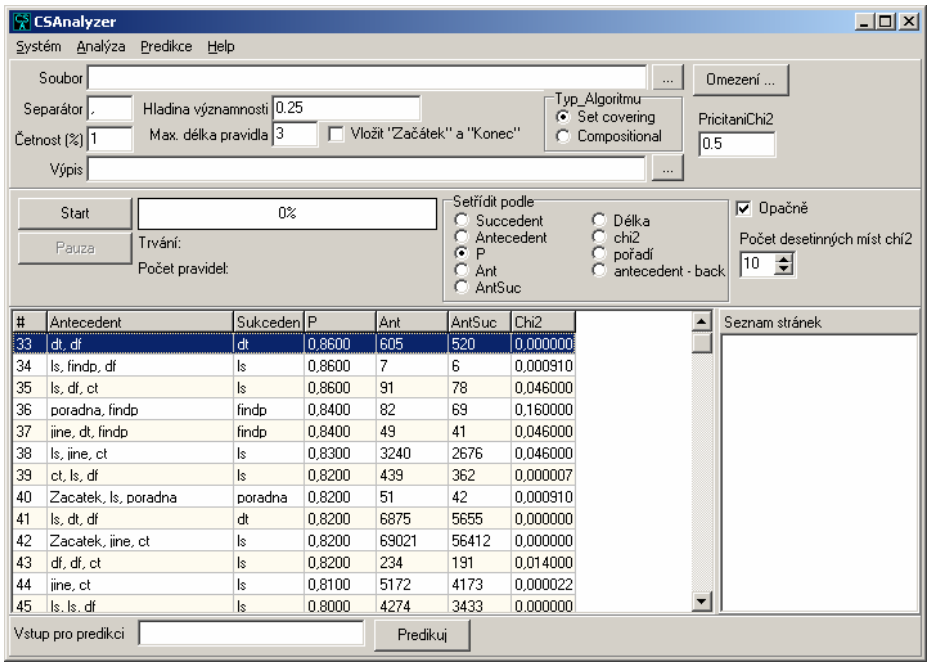
$$P = \frac{\#\text{ of occurrence of } page \text{ in the input episodes}}{\#\text{ of all input episodes}}$$

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. 4. The compositional rule learning algorithm for click-stream analysis



#	Antecedent	Sukceden	P	Ant	AntSuc	Chi2
33	dt, df	dt	0,8600	605	520	0,000000
34	ls, findp, df	ls	0,8600	7	6	0,000910
35	ls, df, ct	ls	0,8600	91	78	0,046000
36	poradna, findp	findp	0,8400	82	69	0,160000
37	jine, dt, findp	findp	0,8400	49	41	0,046000
38	ls, jine, ct	ls	0,8300	3240	2676	0,046000
39	ct, ls, df	ls	0,8200	439	362	0,000007
40	Zacatek, ls, poradna	poradna	0,8200	51	42	0,000910
41	ls, dt, df	dt	0,8200	6875	5655	0,000000
42	Zacatek, jine, ct	ls	0,8200	69021	56412	0,000000
43	df, df, ct	ls	0,8200	234	191	0,014000
44	jine, ct	ls	0,8100	5172	4173	0,000022
45	ls, ls, df	ls	0,8000	4274	3433	0,000000

Fig. 5. Screenshot of the system

3 Experiments

We tested our algorithms on a real data obtained from one Czech internet shop. The log file (about 3 millions of records – the traffic of 24 days) contained the usual information: time, IP address, page request and referee. In addition to this, the log data contained also a generated session ID so the identification of users was relatively easy – we treated as sequence of pages with the same ID as a visit of one user[2]. An example of records in the log file is shown in fig. 4. The log file allowed us to identify two types of information about the visited page: page type and page content. By page type we understand information related to a general structure of the internet shop (detail of a product, shopping chart, product comparison etc), by page content we understand the product (or its category) offered on the page. These two points of view enable us to perform two different types of analyses: analysis of product preferences and analysis of shopping behavior.

```
unix time; IP address; session ID; page request; referee
1074589200;193.179.144.2;1993441e8a0a4d7a4407ed9554b64ed1;/dp
/?id=124;www.google.cz;
1074589201;194.213.35.234;3995b2c0599f1782e2b40582823b1c94;/d
p/?id=182;
1074589202;194.138.39.56;2fd3213f2edaf82b27562d28a2a747aa;/;h
ttp://www.seznam.cz;
1074589233;193.179.144.2;1993441e8a0a4d7a4407ed9554b64ed1;/dp
/?id=148;/dp/?id=124;
1074589245;193.179.144.2;1993441e8a0a4d7a4407ed9554b64ed1;/sb
/;/dp/?id=148;
1074589248;194.138.39.56;2fd3213f2edaf82b27562d28a2a747aa;/co
ntacts/; /;
1074589290;193.179.144.2;1993441e8a0a4d7a4407ed9554b64ed1;/sb
/;/sb/;
```

Fig. 6. Part of the web log

In the first step of data preprocessing we identified particular users by the session ID and we created a file containing sequences of visited pages for each user. So for the log file shown in fig. 4, we created the page-type sequence (session)

```
start, dp, dp, sb, sb, end
```

and the page-content sequence (session)

```
start, 124, 148, end
```

for the user *1993441e8a0a4d7a4407ed9554b64ed1*. In this example, *dp* denotes "detail of product", *sb* denotes "shopping basket", 124 denotes "loud-speaker " and 148 denotes "DVD player". Note, that we added two pages to each sequence found in the data, the `start` page and the `end` page.

In the second step we excluded sessions of length 1. This is a general recommendation as sessions of length 1 are usually created by web bots crawling the web space and collecting pages. Moreover, as we are interested in prediction next page in a click-stream, we need sequences of length 2 and more. This reduces the number of sessions (sequences) to 200 000; the average number of visits by one user (the average length of session) was 16; the median was 8 and modus 2.

In the third step of data preprocessing, we created 2 basic files that entered to the analyses. In the first one, there were saved sequences of page type regardless of the product offered on this page. In the second one, there were saved sequences of page content (sequences of products) regardless of the page type (pages without products were excluded from these sequences). During this step, the products were divided to 30 categories.

[2] Because there was no possibility how to identify two sessions of one user we consider each session with distinct ID as one user.

The modeling consisted in searching for rules using the algorithm described in section 2. The algorithm for searching was running repeatedly with various input parameters.

The first set of experiments was performed for the sequences of page types. Among the obtained results we can find the rule

```
dp, sb -> sb (Ant: 5174; AntSuc: 4801; P: 93%)
```

that covers the session of user *1993441e8a0a4d7a4407ed9554b64ed1* (see above). Another interesting rules were e.g.

```
 ct -> end  (Ant: 5502; AntSuc: 1759; P: 32%)
faq -> help (Ant: 594; AntSuc: 127; P: 21%)
```

In the listing above, *ct* stands for "contact", *Ant* stands for ||*Ant*|| and *AntSuc* stands for ||*Ant*//*Suc*||.

The second set of experiments was performed for the sequences of products (page contents). For the request of the data provider, we generated all rules of length 2 for the sequences of products. From these rules passes of users between products were seen very well. Examples of such rules are:

```
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)
```

The models obtained in both sets of experiments can directly be used to predict the behavior of an 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`.

4 Conclusions

We present two new algorithms to learn decision rules for web usage mining, set-covering and compositional one. Although we developed our algorithms within a project of click-stream analysis, it can be used more generally, to predict occurrence of an event in a sequence of any types of events (e.g. transactions on banking accounts, or network intrusion). Our experiments with a real data of an internet shop showed usefulness of the proposed approach. In our future work, we plan to perform a thorough comparison of both approaches.

References

[1] Berka,P. - Ivánek,J.: Automated knowledge acquisition for PROSPECTOR-like expert systems. In. (Bergadano, de Raedt eds.) Proc. ECML'94, Springer 1994, 339-342.

[2] Bruha I., Kočková S. A support for decision making: Cost-sensitive learning system. *Artificial Intelligence in Medicine*, 6 (1994), 67-82.

[3] Clark P., Niblett T. The CN2 induction algorithm. *Machine Learning*, 3 (1989), 261-283.

[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] Kosala R., Blockeel H.: Web Mining Research: A Survey. *SIGKDD Explorations,* Vol. 2 Issue 1, 2000.

[6] Michalski R.S. On the Quasi-minimal solution of the general covering problem. In: Proc. 5th Int. Symposium on Information Processing FCIP'69, Bled, Yugoslavia, 1969, 125-128.

[7] Spiliopoulou,M. - Faulstich,L.: WUM: A tool for web utilization analysis. In Proc. EDBT Workshop WebDB'98, Springer LNCS 1590, 1999.

[8] Srivastava J., Cooley R., Deshpande M., Tan P-N. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. *SIGKDD Explorations*, Vol. 1 Issue 2, 2000.

[9] Zaiane O., Han J.: WebML: Querying the World-Wide Web for resources and knowledge. In: Workshop on Web Information and Data Management WIDM'98, Bethesda, 1998, 9-12.

[10] [Zaine,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.