MSc Łukasz Mikulski[1]
Department of Formal Languages and Concurrency
Faculty of Mathematics and Computer Science
Nicolaus Copernicus University

MSc Paweł Weichbroth
Department of Information-Technology Management
Faculty of Management and Economics
Gdańsk University of Technology

Discovering Patterns of Web Page Visits from the Association Rules Viewpoint

Abstract

The popularity of the Internet results from the almost unlimited resources of information stored in it. At the same time, Internet portals have become a widespread source of information and note very large number of visits. The list of web pages opened by users is stored in web servers' log files. Extraction of knowledge on the navigation paths of users has become carefully analyzed problem. Currently, there are a number of algorithms that are used for this purpose. The formal representation of knowledge discovered from databases are association rules. Their extraction requires to find all the "frequent" sets. The contribution from the authors is an extension of the Apriori algorithm by adding capacitance of generating association rules during the search of "frequent" sets. Furthermore, the implication of the obtained knowledge is a graphical representation, showing the patterns of users' visits. To this purpose the method of mind mapping is used.

Keywords: knowledge extraction, association rules, patterns of users, data mining.

Introduction

The development of global computer networks, the spread of personal computers and high availability led to an increase in the number of Internet users visiting websites. The success of web services mainly lies in the information - its availability and timelines.

The evolution of the process of generating content on websites from a static to a dynamic approach, allow for easier updates. Information regarding the order in which pages are opened, may be used to gain traffic and predict the behaviour of the positioning of content. Data exploration and associated techniques are now the subject of many studies [2-4].

In the literature of the data mining, analysis of web services is defined as the exploration of Internet resources (Web Mining). Based on the data form, there can be distinguished: an analysis of site content (Web Content Mining), an analysis of the structure of the service (Web Structure Mining) and an analysis of how the service is used by users (Web Usage Mining) [4]. Typically, each click on a link corresponds to the visualization of the HTML document. In this context, session represents a

[1] The research supported by Ministry of Science and Higher Education of Poland, grant N N206 258035.

sequence of clicks in a single service. The session history can be used for online prediction of the sequence of opened pages.

The aim of the authors work is extraction the traffic patterns of onet.pl visitors. Navigation paths of users' activity has been presented in terms of association rules. For this purpose the RuleMiner application, which operates on the basis of the Apriori algorithm [1], was implemented. The algorithm has been extended to the simultaneous generation of association rules. The study was carried out on a server log file. This gave a basis for formulating proposals for the discovered patterns.

Article consists of three parts. The first is a description of the algorithm Apriori and the program RuleMiner whereas the second is the characteristics of research organizations; the results of the study are presented in the third part.

Definitions

Input to the problem is a sequence $(P_i)_{i=1..M}$ consisting subsets of a universe U. Sets P_i correspond to individual user sessions, and elements of these collections, as well as elements of the whole universe U, are single requests for specific subpage. We will be interested only in such collections $A \subseteq U$, which appear as subsets of P_i often enough. Formally, we will describe the frequency using the value of the **support supp(A)**=$|\{i ; A \subseteq P_i\}| / M$, and the sets, whose support exceeds the arbitrarily fixed level **minsupp**, we called **frequent sets**.

Apriori algorithm, used here, works recursively – a growing collections A are generated using smaller collections, especially the subsets of A of cardinality $|A|-1$, hereafter called **hipersubsets**. Formally, a set B is a hipersubset of a set A if and only if $\exists_{a \in A} A = B \cup \{a\}$.

These definitions allow us to make the first important, albeit simple, property of frequent sets. It has been used for the justification of the correctness of the application.

LEMMA 1. ***Each subset of the frequent set is a frequent set.***

Proof:
Let A be an arbitrary frequent set and $B \subseteq A$ be a subset of A.
The set A is frequent, so $\text{supp}(A) = |\{i ; A \subseteq P_i\}| / M \geq \text{minsupp}$. Since B is a subset of A, we have that $A \subseteq P_i \Rightarrow B \subseteq P_i$, so

$$\text{supp}(B) = |\{i ; B \subseteq P_i\}| / M \geq |\{i ; A \subseteq P_i\}| / M \geq \text{minsupp},$$

and hence the set B is also a frequent set, which ends the proof. □

In particular, from this lemma follows that each hipersubset of frequent set is a frequent set.

By a **confidence** of the disjoint couple of sets (A, B), which can be read B under the condition A, we understand the value of **conf(A,B)**=$\text{supp}(A \cup B)/\text{supp}(A)$. If the confidence exceeds an arbitrarily chosen level of **minconf**, the pair (A,B) will be called an **association rule** and denoted by $A \rightarrow B$; A will be called the predecessor of the associative rule patterns of users B – the successor. It is worth to note that we are interested only in **relevant** association rules, that are such rules $A \rightarrow B$ that a set $A \cup B$ is frequent.

The association rules have both probabilistic and logistic meaning, given by their definition and decription. They also satisfy, fundamental for the algorithm, lemma:

LEMMA 2. ***If A→B∪C is a relevant association rule (A, B, C – pairwise distinct), then also A→B, A→C, A∪B→C and A∪C→B are relevant association rules.***

Proof:

Let A→B∪C be an arbitrary relevant association rule. Then

$\text{minconf} \leq \text{conf}(A,B\cup C) = \text{supp}(A\cup B\cup C)/\text{supp}(A)$.

Let us consider $A\cup B \subseteq A\cup B\cup C$; using lemma 1 we get $\text{supp}(A) \geq \text{supp}(A\cup B) \geq \text{supp}(A\cup B\cup C)$. Therefore,

$\text{conf}(A,B) = \text{supp}(A\cup B)/\text{supp}(A) \geq \text{supp}(A\cup B\cup C)/\text{supp}(A) \geq \text{minconf}$

and the set A∪B is frequent, so A→B is a relevant association rule.

On the other hand,

$\text{conf}(A\cup B,C) = \text{supp}(A\cup B\cup C)/\text{supp}(A\cup B) \geq \text{supp}(A\cup B\cup C)/\text{supp}(A) \geq \text{minconf}$,

and hence, because of a frequence of a set A∪B∪C, A∪B→C is also a relevant association rule.

Proofs for rules A→C and A∪C→B are similar. □

In view of the decisive influence of the frequency of the collection A∪B on the relevance of the association rule A→B, we say that relevant association rule A→B is **associated** with a frequent set A∪B. It was used during designing of data structures responsible for the storage of information on a single frequent set and in the process of generating association rules during determining of frequent sets. This is an important extension of the Apriori algorithm.

Extended Apriori Algorithm

There are many algorithms that can be used to find relevant association rules. A key part of each of them is generating the frequent sets occurring in the studied data. In RuleMiner we used a popular algorithm by R. Agrawal and R. Srikant [1]. Data for this algorithm is a sequence of subsets (P_i) of some universe U; each of the subsets is represented by a sequence of pairs consisting of a name of subset (subset ID) and an element (element ID) of that subset. In addition, an important assumption is made – data is sorted due to the first coordinate. The result is that couples describing a subset of the input appear as a compact fragment.

The algorithm computes successively families L_k of frequent subsets of cardinality k, i.e. the sets of cardinality k that are subsets of many collections appearing in the input. Satisfactory level of frequence is given arbitrarily adopted factor minsupp. As a result, only those subsets whose absolute frequency of occurrence in the output is at least minsupp are generated. In the preparatory step of the algorithm, the family L_1 consisting all frequent sets of cardinality 1 is determined. The next steps are made until the new family of frequent set is not empty, consist of three phases.

In the first phase the algorithm calculates, using the already designated family of frequent sets L_k, a new family of candidate sets C_{k+1}. The candidate set of cardinality k+1 is a union of two frequent sets of cardinality k, which differs by only one element. The algorithm does not specify how to designate new family of candidate sets, naive method of quadratic complexity, due to the size of the family L_k, is to check all pairs in that family of sets.

The second phase is a two-stage verification of candidates. The first criterion allows to reject a candidate set $C\in C_{k+1}$ when there is at least one hipersubset of C which does not belong to the family L_k. This is due to the fact that any subset of a frequent set is frequent (lemma 1). The final verification of the family of candidate sets is carried out by re-input data and calculation of real support for the sets.

Last, the third phase of the algorithm iteration, is switching the status of the revised family of candidate sets C_{k+1} to the status of the family of frequent sets L_{k+1}, with simultaneous verification of non-emptiness of this family and preparing for the next step.

In our solution we have added to the second phase of iterative step a generation of the association rules associated with frequent sets found in this step. In the course of the first stage of verifying a candidate set $C \in C_{k+1}$, the association rules founded for its hipersubsets are rewrote (if we associate with a set $C \setminus \{x\}$ the association rule $A \to B$ then with the set C we associate the rule $A \to B \cup \{x\}$). These rules are completed by rules of the form $C \setminus \{x\} \to \{x\}$, where x is an arbitrary element of C, and form a set of candidate rules associated with the set C. Lemma 2 shows that they are all rules theoretically possible to associate with a set C. After second stage of verification, that is the calculation of the actual support for a set C, checking the association rules confidence takes place – those with too low confidence factor are rejected.

Pseudo code:

```
        Variables
        $C_k$ – family of candicate subsets of cardinality $k$
        $L_k$ – family of frequent subsets of cardinality $k$
        F – input data (file, database,…)
---------------------------------------------------------------
        make($L_1$,F)
        verifySupport($L_1$)
        k:=1
        while ($L_k$ not empty) do
        begin
        k++;
        make($C_k$,$L_{k-1}$)
        verifyAndMakeCandidateRules($C_k$,$L_{k-1}$)
        verifySupport($C_k$,F)
        verifyCandidateRules($C_k$)
        $L_k$:=$C_k$
        end
```

Implementation of RuleMiner

The application RuleMiner was implemented in Java. It consists of two classes - the class Main, which is responsible for managing the process of calculation, and the Item class, which stores data of a single frequent set (also a candidate), and association rules set for this set (including the candidate).

The class Item stores in its fields data of a single frequent set. These data are members of the given set (TreeSet), the number of occurrences of this set in the input, the family of hipersubsets represented by the complement (TreeSet), and the list of association rules for this set, together with the support of predecessor of the rule (TreeMap). The size of the two collections are equal to the cardinality of the represented frequent set. The keys of the map appearing as one of the fields of the class Item, are the subsets of the represented frequent set. It potentially causes a risk of the exponential size growing, with respect to the number of elements. In empirical testing the effect was not visible and was offset by the high level of minimal confidence and, using the lemma 2, by saving only part of the structure of the association rules – a representative part that guarantees the occurrence of other rules.

In addition, the class Item provides constructors and methods to manipulate frequent collections. Constructors allows to create a new object that contains a single element or is a combination of two existing objects. The methods can retrieve the information needed in the generation and verification of the candidate sets and association rules as well as save the partial results of the verification candidate sets and associated with them rules.

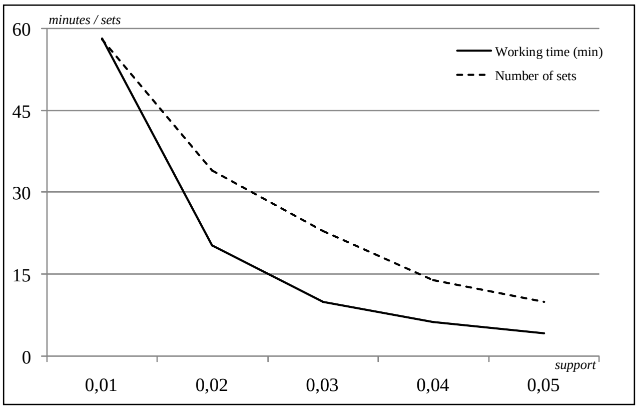
The class Main implements the steps of described above Apriori algorithm. In the course of action only one family of candidate sets and one family of frequent sets are stored (in the TreeSet collection). After each verification of the family of candidates, and changing the status to the status of the frequent sets family, the current results are save to a text file. In this way, we do not need to store in memory all the frequent sets.

Finally, it is worth noting that the algorithm is well suited for parallelization. Between each of the steps occurring in pseudocode synchronization is needed, however, steps (1), (2), (7), (8), (9) and (10) can be performed in parallel. In steps (1) and (9) parallelization can be obtained by the method of the input decomposition (with summing the results during the synchronization or synchronized objects in a common memory). In steps (2) and (10) we can decompose the families of considered there candidate or frequent sets – computations are performed for each of these collections independently (a similar method can be used in step (9), but decomposition of the input seems to be more appropriate). Other steps, (7) and (8), operate with the family of candidate sets C_k from the current iteration and the family of frequent sets L_{k-1}, generated in the previous iteration. Here, the only efficient way of decomposition seems to be splitting the family L_{k-1}, with synchronized access to objects stored as the family C_k.

The presented algorithm was used to find all the frequent sets, stored in a server log file. It has no application in taxonomy and does not operate with "moving windows". In its implementation may occur two problems.

The first concerns the nature of resources in the process of transformation of data, which takes place during each pass through the log file. The program was launched in Eclipse environment (Java virtual machine version 1.6.0) on IBM PC class computer with an Intel Core2 Quad processor (2.40 GHz) and 2 GB of memory. Figure 1 shows the execution times of RuleMiner in relation to the number of found frequent sets for five different levels of support.

Fig. 1: The effectiveness of the program RuleMiner



source: own study

The second problem concerns of the generating candidate sets and the calculation of support for these collections. Both processes are of fundamental importance for the algorithm efficiency. In RuleMiner both issues have been solved.

Survey Organization

Only a preliminary test was carrued out. We used observation as the method, since events were registered without interference of the researcher. In this paper, the hypothesis about existence patterns of the users' behaviors of the portal onet.pl was verified. Investigations were divided in three stages: (1) the choice of variables and cleaning the data, (2) finding frequent item with specified minimum support for service and (3) extraction of association rules based on found frequent items.

A WWW server's log file contains full history of requested access to files, kept on server. The majority of WWW servers record activity in log files using Common Log Format schema defined by CERN and NCSA as part of http protocol. According to this standard, a log entry contains: client IP address, user identifier, access time, request method, URL page accessed, the protocol used for data transmission, an error code and the number of bytes transmitted.

Primitive data, received from the onet.pl portal, include fields respectively: (1) session time, (2) session identifier, (3) user identifier, (4) service name, (5) the address of html file path in frame of service. The first step of extraction of knowledge process is data preparation which include cleaning, transformation and selection variables. The number of numeric variables in the processed log file was reduced from six to two: (2) session identifier and (5) the address of html file path. We call this process data cleaning.

Knowledge Extraction

After the data cleaning, the file log was analyzed in *RuleMiner* program. The goal of the second stage in experiment, was finding frequent items. Support was defined arbitrarily on five different levels. The required level of support is every time defined by researcher. In context of experiment, the element of item represents exactly defined html document. Data analysis shows which items have the highest support suitably:

- *{[www]}* 80,36%,
- *{[email]/cnp/login.html.php3}* 27,53%
- *{[email]/np/dynamic/folder.html}* 26,29%
- *{[email]/np/dynamic/folder.html/open.html}* 15,43%
- *{[sport]/wc_volleyball/news.html}* 13,17%
- *{[info]/world/item.html}* 12,70%
- *{[info]/country/item.html}* 12,68%
- *{[email]/np/dynamic/folder.html/delete.html}* 11,96%
- *{[sport]/ski_jumps/ski_jumps/news.html}* 10,01%.

Items are dependent one to another – they occurr together with different elements or very seldom separately.

For support on level 1%, based on found frequent items, RuleMiner generated five types of association rules. The number of rules depends on cardinality of sets. For two items program generated 87 rules, for three 308, for four 581 for five 411 and for six

119 association rules. We used mind mapping methods for graphic representation of them.

Discovered Users' Patterns

An association rule, being logical statement, is written in form "if-then". The number of variables describes the level of discovered knowledge of users' navigation paths (table 1). For assumed minimum support and confidence, it was found that the number of accessed pages in one session did not exceed six. This fact implies kinds of generated association rules. The number of variables used to build a rule is in range <2;6>. This thesis has been confirmed in different sources [6].

Table 1: Two-element association rules chosen for the *{[www]}* item

No.	Association rule	Support	Confidence
1	[[email]/login] → [[www]/]	0,275	0,823
2	[[info]/world/] → [[www]/]	0,127	0,978
3	[[sport]/ski_jumping] → [[www]/]	0,1	0,964
4	[dating] → [[www]/]	0,038	0,936
5	[tv] → [[www]/]	0,013	0,883

source: own study

It was observed that 95% rules with highest support, concerned exclusively three services: [www], [email] and [dating]. Rules with the highest support concern [www] item.

The method of logical interpretation of rules is identical. For example, rule number 5 shows that, if the [tv] page was accessed, in 88% of sessions, the [www] was accessed too. We can suppose that remaining 12% is the direct access to [tv] item, which took place from omission the [www] item, which is the main page of the portal. It takes place when user sends the URL address (http://onet.pl/tv/) of the web page to the other user or open the page directly, for example using the Favorites option available in every web browser. Support (fraction) shows that 1,3% of sessions contain requests of two items: [www] and [tv]. Taking into consideration the size of this paper, interpretation of acquired results was limited to formulating general conclusions.

Conducted survey revealed two essential users' patterns~~ (fig. 3)~~. First pattern applies to users, interested in sport. This pattern is represented by the rule: {[sport]/football/, [sport]/ski_jumping/} → {[www]} with support 2,8% and confidence of 94,8%. Second pattern consists of three services: [business], [info] and [www]. We can say for sure that the group of users seeking current information about stock market and news from the world has been identified.

Similar to work [3], association rules can be graphically presented as a tree pattern.

Extracted association rules also pointed out on internal dependences in perspective of one service. For example, for email service or dating service, we can notice paths between folders in the service, discovered by the actions taken by users like: open, send, delete, new, log in or log out.

Summary

Presented in the paper extension of the Apriori algorithm enables to extract essential association rules during the process of finding frequent items. We used the idea of Apriori algorithm which assumes that to find objects with specific size it is enough to find objects one element smaller. Apriori itself finds only frequent items but implemented algorithm in program RuleMiner also extract association rules.

This approach to the given problem enables to limit the number of keeping items and association rules in operating memory each time. After each verification, partial results are saved to file. It results that the busy memory becomes free. In addition, the number of keeping

association rules simultaneously is indispensable restricted to minimum. It is achieved by defining high confidence as well as getting rid of the rules which cannot be extracted on basis others. Presented solution and properties of frequent items was formally defined and well-founded.

In carried out survey, for every given support, we managed to achieve satisfied results. Support and confidence were two parameters on which we decided what knowledge presented as association rule is valuable enough. Only 5% of them was classified valuable. The survey had experimental character and set a base for developing functionality of the program as well as strengthen the cooperation with the onet.pl portal.

References

1. Agrawal R., Srikant, R., *Fast algorithms for mining association rules*, [in:] *Proceedings of the Twentieth International Conference on Very Large Data Bases*, pp. 487-499, Morgan Kaufmann, San Francisco 1994.
2. Hatonen K., Boulicaut J. F., Klemettinen M., Miettinen M., Mason C., *Comprehensive Log Compression with frequent patterns*, DaWaK 2003, LNCS 2737, pp. 360-370, Springer-Verlag Berlin, 2003.
3. Ivancsy R., Vajk I., *Frequent pattern mining in web log data*, Acta Polytechnica Hungarica, Vol. 3, No. 1, pp. 77-90, 2006.
4. Kosala R., Blockel H., *Web mining research: A survey*, [in:] „Newsletter of the Special Interest Group (SIG) on Knowledge Discovery and Data Mininig" SIGKDD: GKDD Explorations, 2000, nr 1.
5. Weichbroth P., Korczak J., *Data mining*, [in:] Business informatics. Part One. Propaedeutics of computer science. Information technologies, red. Jerzy Korczak, Wrocław 2006 [in polish].
6. http://www.webuser.co.uk/news/81267.html?aff