Mining Association Rules in Hypertext Databases

José Borges and **Mark Levene**
Department of Computer Science
University College London
Gower Street
London WC1E 6BT, U.K.
Email:{j.borges, mlevene}@cs.ucl.ac.uk

Abstract

In this work we propose a generalisation of the notion of association rule in the context of flat transactions to that of a composite association rule in the context of a structured directed graph, such as the world-wide-web. The techniques proposed aim at finding patterns in the user behaviour when traversing such a hypertext system. We redefine the concepts of confidence and support for composite association rules, and two algorithms to mine such rules are proposed. Extensive experiments with random data were conducted and the results show that, in spite of the worst-case complexity analysis which indicates exponential behaviour, in practice the algorithms' complexity, measured in the number of iterations performed, is linear in the number of nodes traversed.

Introduction

Data Mining and Knowledge Discovery is an active research field whose purpose is the study of tools to cope with the explosive growth in the amount of data being stored. Several data mining techniques have been being studied and among them is the problem of mining association rules in large datasets of flat transactions. The problem was introduced in (Agrawal, Imielinski, & Swami 1993), and has been being intensively studied since then, see for example (Agrawal *et al.* 1996). The standard example aims to find in a large dataset of supermarket sales transactions rules such as "90% of the customers that purchase milk also buy bread".

The explosive growth and widespread use of the World-Wide-Web (WWW) suggests the potential of adapting the concept of association rule to the context of internet commerce, such as online bookstores. The success of such web-sites relies heavily on the quality of the service provided, especially since on the WWW it takes just a mouse-click for a client to move to a competitor site. Therefore, a method to analyse the user patterns of behaviour when browsing online services can be very useful by helping the service provider to understand the users' preferences and to improve the design of the underlying hypertext system accordingly. This idea was first suggested in (Chen, Park, & Yu 1996) wherein a method is proposed to mine access patterns in a web-like environment. The user navigation information is obtained from server log data that is converted into a set of *maximal forward references* and two algorithms are devised to mine the *large reference sequences*. The difference between large reference sequences and large itemsets is that the former consists of consecutive references in a maximal forward reference, and the latest is just an arbitrary combination of items in a transaction.

We propose the generalisation of the concept of association rule rather than pre-processing the server data into a form which is then amenable to analysis by the standard association rule algorithms as in (Chen, Park, & Yu 1996). The semantics of the user's navigation are maintained thus alleviating the danger for over evaluating references when the input is converted to standard association rule form. We formalise the notion of composite association rule, confidence and support in the context of a directed graph such as the WWW, or more generally a hypertext system (Nielsen 1990). Two algorithms are proposed: one is a modification of the directed graph Depth-First-Search algorithm, and the other uses an incremental approach to build the set of composite rules of size $n+1$ from the set of composite rules of size n. Extensive experiments with random data were carried out and the results show that in spite of the complexity analysis which indicates exponential behaviour, in practice the algorithms' complexity, measured in the number of iterations performed, is linear in the number of nodes traversed. However, the performance of the algorithms in terms of CPU time is still exponential in the number of nodes traversed, since for larger graphs the algorithms have to deal with more complex data structures.

The rest of the paper is organised as follows. Section 2 provides a description of the problem, and the formal model of the problem is presented in Section 3. Section 4 then presents two algorithms for mining user patterns. Section 5 presents the experiments results, and related work is discussed in Section 6. Finally, Section 7 presents our concluding remarks.

Problem Description

Log files contain information that characterises user accesses to the server, including the user identification (or their IP address), the URL of the requested page and the date and time of access. With this information we can reconstruct user navigation sessions and build a weighted directed graph that summarises it. The graph describes the user's *view* of the hypertext system and delineates the domain for mining user patterns. Each node in the graph corresponds to a visited page and each arc to a traversed link. We begin with all graph links having a zero weight, and when scanning the user sessions, for each evaluated link, we increment by one the corresponding arc's weight. The result is a graph where the weight of an arc corresponds to the number of times its corresponding link was traversed. Since the model represents a summary of a collection of sessions we lose the concept of an individual session. Therefore, the resulting weighted graph is intended to represent the navigation information of a single user or a group of users with similar interests, and the aim is to mine trails of the user's knowledge of the system as opposed to mining trails of the user's individual sessions. Note that the model can be modified so that the weight of a link is dependent on the history of the sessions by having a node correspond to a sequence of visited pages instead of an individual page.

We must keep in mind, however, that log files do not always reflect the exact user behaviour due to the use of various levels of cache and the difficulty in identifying individual users.

Definition 1 A *hypertext system* is a database of pages connected by oriented links that provide a non-sequential method to access information.

We are assuming that our hypertext system only contains elementary pages, where an *elementary page* is a page that encompasses a single concept or idea, such that we cannot decompose its content without loss of meaning. We also assume that: (i) every link which refers to a page refers to the page as a whole and not to a subset of its content; (ii) the hypertext system does not contain loops, i.e., links in which the source page coincides with the target page; (iii) there is at most one link between any source and target page.

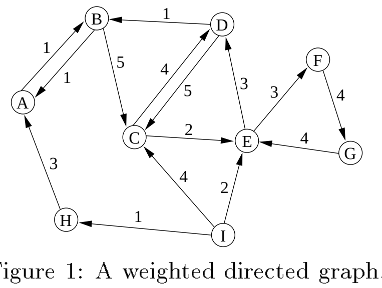

Figure 1: A weighted directed graph.

Formal Model

We have a directed graph $G = (N, E)$ that consists of a set of *nodes*, N and a set of *arcs*, E. A node, $A \in N$, is the entity that corresponds to a page. An arc is a pair $(A, B) \in E$, and is the entity that corresponds to a link; for simplicity we assume there is at most one link between any two pages. Given a link (A, B) we call A the *source* and B the *target* node. We do not assume that the graph is connected. In addition the graph is weighted and the weight of an arc expresses the number of times the user traversed the corresponding link.

The basic interaction a user can have with a hypertext system is to choose a link among all those available in a page. To capture the user's preferences among the available links in a page we introduce the concept of a *single association rule*, which is a statement such as "when a user reaches node A he will next choose link $A \rightarrow B$ with a certain probability". The probability of the link being chosen is defined as the *confidence* of the rule (see Definition 3).

Definition 2 A *single association rule* is an expression such as $(A \rightarrow B)$, where $A, B \in N$ and $(A, B) \in E$, meaning that when the user reaches node A he will next choose the link (A, B), with certain confidence.

A single association rule $r = (A \rightarrow B)$ holds with confidence $\mathcal{C}_r$, where $\mathcal{C}_r$ is the proportion of traversals with A as a source node that had B as the target node.

Definition 3 $|(A, B)|$ represents the number of times link (A, B) was traversed and $\sum_{\{x|(A,x)\in E\}} |(A, x)|$ the number of times the user traversed all the links that have A as their source node. A single association rule $r = (A \rightarrow B)$ holds with *confidence* $\mathcal{C}_r$, where $\mathcal{C}_r$ is:

$$\mathcal{C}_r = \frac{|(A,B)|}{\sum\limits_{\{x|(A,x)\in E\}} |(A,x)|}$$

To detect the relevant rules we define the concept of *support*, since a rule that holds with full confidence is interesting only if the corresponding link was traversed a significant number of times, relative to the average.

Definition 4 A single association rule $r = (A \rightarrow B)$ holds with *support* $\mathcal{S}_r$, where $\mathcal{S}_r$ is defined as the number of times the link (A, B) was traversed over the average of times all the links in the graph were traversed:

$$\mathcal{S}_r = \frac{|(A,B)|}{\frac{\sum\limits_{\{i|(x_i,x_{i+1})\in E\}} |(x_i,x_{i+1})|}{|E|}},$$

where $|E|$ is the number of arcs in E.

This definition leads to values of support distributed around the value 1, meaning that rules with support greater than 1 consist of links traversed more than the average. Given the probability distribution of the weights we can test the probability of having a certain level of support.

In the example of Figure 1 we have $\mathcal{C}_{(C\rightarrow E)} = \frac{2}{2+4} = 0.33$ and $\mathcal{S}_{(C\rightarrow E)} = 2/(\frac{43}{15}) = 0.70$.

We will now extend the preceding concepts to the general case of rules with any number of links. The concept of trail plays an important role in our search for patterns on the user behaviour; following is the definition of trail according to (Buckley & Harary 1990).

Definition 5 A *trail* $(v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n)$ is an alternative sequence of nodes and arcs such that $v_i \in N, 1 \leq i \leq n$ and every $e_i = (v_{i-1}, v_i) \in E, 2 \leq i \leq n$ are distinct.

Observation 1 Since we are assuming that the graph does not contain any two links with the same source and target node a trail can be characterised solely by its node set. From now on a trail $(v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n)$ will be referred to as $(v_1, v_2, \ldots, v_n)$.

We now generalise the concept of a rule to *composite association rule*, which is a statement such as "when a user browses the hypertext system he will traverse the trail $A_1, A_2, \ldots, A_n$ with a certain probability".

Definition 6 A *composite association rule* is an expression of the form $[(A_1 \rightarrow A_2) \wedge \ldots \wedge (A_{n-1} \rightarrow A_n)]$ where $A_1, A_2, \ldots, A_{n-1}, A_n \in N$ and $\{(A_i, A_{i+1}) | i = 1, \ldots, n-1\} \in E$, meaning that when a user traverses the hypertext system there is a certain confidence that he will follow the trail $(A_1, A_2, \ldots, A_{n-1}, A_n)$.

Definition 7 For a composite association rule $r = [(A_1 \rightarrow A_2) \wedge \ldots \wedge (A_{n-1} \rightarrow A_n)]$ *confidence* $\mathcal{C}_r$ is defined as the product of the confidences of all the corresponding single rules:

$$\mathcal{C}_r = \prod_{i=1}^{n-1} \mathcal{C}_{(A_i \rightarrow A_{i+1})}$$

Definition 8 A composite association rule $r = [(A_1 \rightarrow A_2) \wedge \ldots \wedge (A_{n-1} \rightarrow A_n)]$ holds with *support* $\mathcal{S}_r$, where $\mathcal{S}_r$ represents the average of times the links of the rule were traversed over the average of times all the links of the graph were traversed.

$$\mathcal{S}_r = \frac{\dfrac{\sum_{i=1}^{n-1} |(A_i, A_{i+1})|}{n-1}}{\dfrac{\sum_{i|\{(x_i, x_{i+1})\} \in E} |(x_i, x_{i+1})|}{|E|}}$$

Observation 2 As can be verified when a composite association rule has only one link (it is a single association rule) Definition 4 is a special case of Definition 8.

In example of Figure 1 $\mathcal{C}_{(B \rightarrow C \rightarrow D)} = \frac{5}{5+1} \times \frac{4}{2+4} = 0.55$ and $\mathcal{S}_{(B \rightarrow C \rightarrow D)} = (\frac{5+4}{2}) / (\frac{43}{15}) = 1.57$.

Definition 9 The *monotonic support*, $\mathcal{S}_m$, of a composite association rule is defined as the minimum value of support among all the single rules that constitute the composite rule.

Definition 9 has the advantage of potentially improving the algorithms performance, since for the same support threshold, rules tend to be shorter with this definition. On the other hand, Definition 8 has the advantage of capturing rules that are globally good but have some of its links with support bellow the threshold.

In the example of Figure 1 trail $H \rightarrow A \rightarrow B \rightarrow C \rightarrow D$ has $\mathcal{S}_r = 1.13$ and $\mathcal{S}_m = 0.35$. Therefore, if we have support threshold $\mathcal{S} = 1$ the trail is a rule only with the non-monotonic definition. Note that the definition of confidence is itself monotonic.

We now define some useful concepts which clarify the description of our techniques. Our approach aims at finding all the trails in the graph with support and confidence above specified thresholds, $\mathcal{C}$ and $\mathcal{S}$ respectively. An initial trail is expanded with neighbour links until the values of its characteristics drop to values bellow the respective thresholds.

Definition 10 Given a trail $t = (v_1, \ldots, v_n)$, a *neighbour link* is a link that when appended to the trail t the resulting trail t' complies with Definition 5.

Note that a trail can have cycles but every arc in the trail is distinct. Definition 10 can be specialised to two types of neighbour links as follows.

Definition 11 Given a trail $t = (v_1, \ldots, v_n)$, a *backward neighbour* is a link (x, v_1), where $x \in N$, $x \neq v_n$ and $(x, v_1) \in E$, with $(x, v_1) \notin t$; a *forward neighbour* is a link (v_n, y), where $y \in N$ and $(v_n, y) \in E$, with $(v_n, y) \notin t$.

Algorithms for Mining Composite Association Rules

We now present two algorithms to mine composite association rules that implement the non-monotonic definition of support, however the extension to the monotonic definition is straightforward. Note that when using its non-monotonic definition, support can not be used as a pruning criterion. The intermediate output of the algorithms is the set of all trails with confidence above the threshold, i.e., the candidate rules (CR), wherein those which have support above the threshold are the rules.

We now explain the notation used in the algorithm description. Given a trail x, $forw\_neigh(x)$ refers to its next non-visited forward neighbour and $last\_link(x)$ to the last link in x. Given a trail x and a link e, $x + e$ denotes the concatenation of x and e. Finally $x[i]$ refers to the i^{th} link of trail x.

A Modified Depth-First Search Algorithm

This algorithm is a special case of a directed graph DFS. From each link in the graph a depth first search tree is built in which each branch corresponds to a candidate rule. Each DFS tree identifies all the candidate rules that have the root link of the tree as their first link. The exploration of a branch ends when it finds a node such that all its links have already been marked as visited, or when the confidence of the corresponding trail drops to a value bellow the threshold $\mathcal{C}$. In addition each branch exploration has is own independent list of visited links, since it corresponds to a new trail.

Algorithm 1.1 (Modified_DFS(G,$\mathcal{C}$))
1. **begin**
2. **for** each $\{e \in E : \mathcal{C}_e \geq \mathcal{C}\}$
3. Explore(e, $\mathcal{C}_e$);
4. **end for**
5. **end.**

Algorithm 1.2 (Explore($trail$, $\mathcal{C}_{trail}$))
begin
  $CR := CR \cup trail$;
  for each ($fn := forw\_neigh(trail)$)
    if ($\mathcal{C}_{fn} \times \mathcal{C}_{trail} \geq \mathcal{C}$) **then**
      Explore($trail + fn$,$\mathcal{C}_{trail}$);
  end for
end.

An Incremental Step Algorithm

This algorithm takes advantage of the principle that every subtrail contained in a trail with confidence above the threshold has itself the confidence above the threshold. This means that a rule with n links can be obtained by combining the rules with $n-1$ links. The algorithm starts with the set of trails with $n = 1$ links, CR_1, that have confidence above the threshold $\mathcal{C}$, and recursively builds the set of trails with $n+1$ links, CR_{n+1}, from the set of trails with n links, CR_n.

Algorithm 2 (Incremental_Step(G, $\mathcal{C}$))
begin
  for each $e \in E$
    if $\mathcal{C}_e \geq \mathcal{C}$ **then**
      $CR_1 := CR_1 \cup e$;
  end for
  $i = 1$;
  repeat
    for each $r \in CR_i$
      for each $\{x \in CR_i : x \neq r \wedge (x[j] = r[j+1] : 1 \leq j \leq i)\}$
        if $\mathcal{C}_r \times \mathcal{C}_{last\_link(x)} \geq \mathcal{C}$ **then**
          $CR_{i+1} := CR_{i+1} \cup (t + last\_link(x))$;
      end for
    end for
    $i = i + 1$;
  until ($CR_{i+1} = \emptyset$)
end.

Experiments

To assess the performance of the algorithms we conducted extensive experiments with random data. The generation of random data consisted in two steps: the characterisation of the hypertext system graph (number of nodes, N and the average number of links per node, i.e., graph density or GD); and the characterisation of the user interaction with it (weight of each arc). To generate the arcs weight we considered two probability distributions that are known to characterise a great variety of problems, the Uniform distribution and Poisson distribution. Several tests were performed for different values of the parameters that characterise the probability distributions and the graph.

With the DFS algorithm an iteration corresponds to the process of checking if a forward neighbour of a trail can be augmented to the current trail. With the Incremental_Step algorithm an iteration corresponds to the evaluation of two trails in the CR set to check if they can form a trail for the CR_{x+1} set.

Performance Analysis

The experiments show that the chosen probability distribution and respective parameters do not have significant influence on the performance of the algorithms.

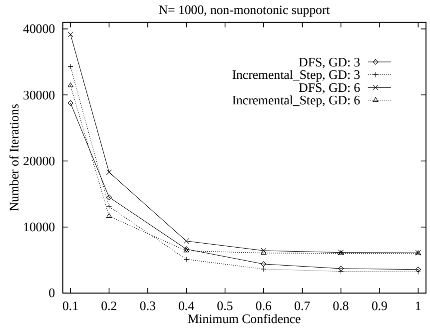

Figure 2: Number of iterations for varying confidence.

Figure 2 represents the number of iterations of the algorithms for a graph with 1000 nodes and two different values for the graph density. We can see how the performance is significantly affected by the confidence threshold, especially for values below 0.4, due to a much larger number of rules and to a larger average size of the rules. The CPU execution time follows a similar trend.

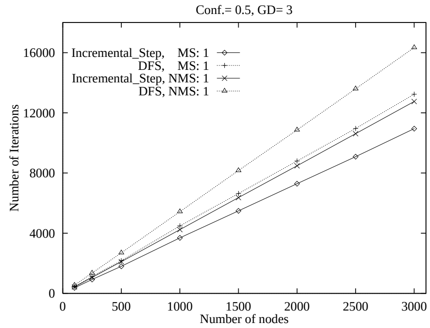

Figure 3: Number of iterations for varying graph size.

Figure 3 shows that despite of the exponential worst case time complexity of the algorithms, in practice both algorithms complexity, measured in the number of iterations performed, is linear in the number of nodes. We can also see that the monotonic support definition, (MS), leads to a lower number of iterations than the non-monotonic definition,(NMS), with both algorithms.

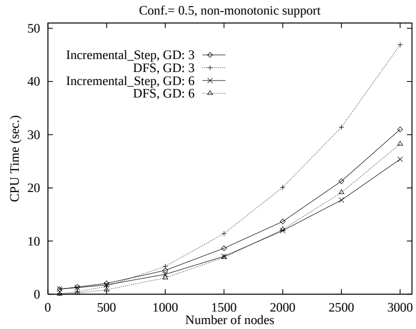

Figure 4: CPU time for varying graph size.

In Figure 4 we see that there is not a direct correspondence between the number of iterations and CPU time, since the same iteration takes longer for larger graphs due to the fact that it has to manipulate more complex data structures. In addition we can also say that for a fixed number of nodes when the graph density increases the performance (in terms of CPU time) improves although the number of iterations also increases. This fact can be explained since the higher the graph density the more rules we have, but these have shorter length, and iterations that deal with small rules are significantly faster. In Figure 5 we see how the size of the largest rule increases when the confidence threshold decreases and how the graph density and monotonic support definition affect the rules size.

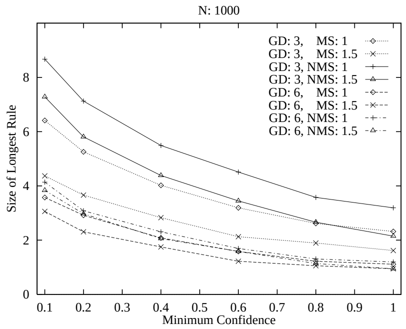

Figure 5: Maximum size of the rules for varying confidence.

Related Work

Techniques for mining association rules between items in large data sets have been an active topic of research since (Agrawal, Imielinski, & Swami 1993). In (Chen, Park, & Yu 1996) the idea of applying these techniques to the problem of mining user access patterns in a web environment was introduced. In their approach log data is converted into a set of maximal forward references, and two algorithms are proposed to find in this set the frequent traversal patterns. In our opinion this innovative approach has two drawbacks. The first is the way backward references are discarded because in general, log data does not provide enough information to identify them unambiguously. The second is the procedure used to build the maximal forward references in which links closer to the root in the resulting traversal tree tend to be over-evaluated. For example, a link from the root to one of its child nodes will appear in all forward references passing through that child, enlarging its support, while this link may have only been traversed once.

The work by Yan *et al.* in (Yan *et al.* 1996) proposes a clustering technique to automatically classify visitors of a web site according to their access patterns, and a method for dynamic link suggestion based on the obtained categories. In (Shahabi *et al.* 1997) the authors propose a profiler to keep record of the user navigation session and in (Zarkesh *et al.* 1997) an algorithm that clusters into classes the users that navigate with similar interests and checks the probability of a new user will become a member of each class.

Our work is also related with another emerging research field, that is the study of techniques to create adaptive web sites (Perkowitz & Etzioni 1997), i.e. sites which automatically improve their presentation by learning from user access patterns.

Concluding Remarks

We have proposed a generalisation of the association rule concept to the context of a hypertext system, such as the WWW, aiming at capturing user patterns when accessing on-line services, such as an electronic bookstore. We have formalised the concept of composite association rule, and redefined support and confidence in this context. Two algorithms to mine composite association rules are proposed and the results of experiments with random data revealed that in practice the algorithms complexity, measured in the number of iterations performed, is linear in the number of nodes of the hypertext system. The explosive growth of the WWW suggests the potential of these concepts.

As future work, we plan to conduct simulations with real data and to develop a more general model to capture access patterns between different groups of users, not necessarily with the same goal in hand.

References

Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; and Verkamo, A. I. 1996. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, 307–328. AAAI/MIT Press.

Agrawal, R.; Imielinski, T.; and Swami, A. 1993. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, 207–216.

Buckley, F., and Harary, F. 1990. *Distance in Graphs*. Addison-Wesley.

Chen, M.-S.; Park, J. S.; and Yu, P. S. 1996. Data mining for path traversal patterns in a web environment. In *Proc. of the 16th Conference on Distributed Computing Systems*, 385–392.

Nielsen, J. 1990. *Hypertext & Hypermedia*. Academic Press.

Perkowitz, M., and Etzioni, O. 1997. Adaptive web sites: an AI challenge. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI'97)*.

Shahabi, C.; Zarkesh, A. M.; Adibi, J.; and Shah, V. 1997. Knowledge discovery from users web-page navigation. In *Proc. of the 7th International Workshop on Research Issues in Data Engineering*, 20–29.

Yan, T. W.; Jacobsen, M.; Garcia-Molina, H.; and Dayal, U. 1996. From user access patterns to dynamic hypertext linking. In *Proc. of the 5th International World Wide Web Conference*, volume 28, 1007–1014.

Zarkesh, A. M.; Adibi, J.; Shahabi, C.; Sadri, R.; and Shah, V. 1997. Analysis and design of server informative www-sites. In *Proc. of the 6th International Conference on Information and Knowledge Management (CIKM'97)*, 254–261.