A New and Versatile Method for Association Generation

Amihood Amir[*] Ronen Feldman[**] Reuven Kashi[***]
Georgia Tech and Bar-Ilan University · Bar-Ilan University · Bar-Ilan University

Abstract. Current algorithms for finding associations among the attributes describing data in a database have a number of shortcomings:

1. Applications that require associations with very small support have prohibitively large running times.
2. They assume a *static* database. Some applications require generating associations in real-time from a dynamic database, where transactions are constantly being added and deleted. There are no existing algorithms to accomodate such applications.
3. They can only find associations of the type where a conjunction of attributes implies a conjunction of different attributes. It turns out that there are many cases where a conjunction of attributes implies another conjunction only provided the *exclusion* of certain attributes. To our knowledge, there is no current algorithm that can generate such *excluding associations.*

We present a novel method for association generation, that answers all three above desiderata. Our method is inherently different from all existing algorithms, and especially suitable to textual databases with binary attributes. At the heart of our algorithm lies the use of subword trees for quick indexing into the required database statistics. We tested our algorithm on the Reuters-22173 database with satisfactory results.

[*] Department of Mathematics and Computer Science, Bar-Ilan University, 52900 Ramat-Gan, Israel, (972-3)531-8770; amir@cs.biu.ac.il; Partially supported by NSF grant CCR-92-23699 and the Israel Ministry of Science and the Arts grant 6297.

[**] Department of Mathematics and Computer Science, Bar-Ilan University, 52900 Ramat-Gan, Israel, (972-3)531-8629; feldman@cs.biu.ac.il; Partially supported by the Israel Ministry of Science and the Arts grant 8615.

[***] Department of Mathematics and Computer Science, Bar-Ilan University, 52900 Ramat-Gan, Israel, (972-3)531-7529; kashi@cs.biu.ac.il.

1 Introduction

Algorithms for finding association rules appear in [2, 1, 11, 8, 9, 13]. These algorithms find *covers*, i.e. sets with large enough support. They then check the confidence on suitable partitions of the covers. In the worst case these algorithms have exponential time complexity. In reality, though, their running time on the tested databases is manageable. The reasons for this are the heuristics they implement. The heuristics are all based on the support. The speed of the algorithm is proportional to the size of the support. The higher the support – the faster the algorithm. For very small supports these algorithms break down.

Consider the following two application domains that require mining for associations. The first is a scenario that involves finding associations between labels of articles in textual collections [5, 6, 7]. Another important example is in a medical database where transactions are medical records, the type of associations needed are those between treatment and outcome, symptoms and procedures, demographic attributes and medical outcome, initial symptoms and length of hospital stay.

Both of these important applications have needs that are not answered by the existing algorithms.

1. **Small Support:** Medicine is replete with rare but lethal cases. Under these circumstances it is better to err on the side of safety and consider associations with small support. What is needed, then, is a completely new method of generating associations in a *support independent* manner.
   There is no current association generation algorithm that can handle tiny supports.
2. **Dynamic Databases:** In a fast-changing database (growing or shrinking), the user may need results in real time as new records accumulate or change and the association needs to reflect all known data.
   In [4] two methods for real time generation of associations in incremental databases were developed. They were tested and compared on the Reuters-22173 database using the KDT data mining system. While the methods of [4] performed quite well on the Reuter-22173 test database, they suffer from two main drawbacks. (1) They are both support dependent, and (2) They assume only an *increasing* database, but can not handle a *decreasing* database.
3. **General Associations:** In our definition of association we followed Agrawal, Imielinski and Swami [1]. Their definition of an association rule means that a conjunction of attributes implies a conjunction of other attributes, $A_1 \wedge A_2 \wedge \cdots \wedge A_i \Rightarrow B_1 \wedge \cdots \wedge B_j$. Recently, Mannila and Toivonen [10] discuss the theoretical option of having a general boolean formula implying another general boolean formula. Many such formulae are meaningless from a data mining perspective. For example, $A_1 \wedge A_2 \wedge \cdots \wedge A_i \Rightarrow \neg B_1 \wedge \cdots \wedge \neg B_j$ for all subsets $\{B_1, \ldots, B_j\}$ where it is not the case that $A_1 \wedge A_2 \wedge \cdots \wedge A_i \Rightarrow$

B_ℓ, $\ell = 1, \ldots, j$. It is clearly not worthwhile to seek such meaningless associations, especially considering the fact that their number is exponential in the number of attributes (587 in the Reuters-22173, for example).

However, there are several interesting general cases. One such example is associations of the following form: $A_1 \wedge A_2 \wedge \cdots \wedge A_i \Rightarrow B_1 \wedge \cdots \wedge B_j$ where some of the A_ℓ's are negations. This means that there is sufficient support and confidence for a rule, only provided that certain attributes are *not present.*

Example: $A \wedge B \wedge \neg C \Rightarrow D$ means that there is sufficient support for set $\{A, B, D\}$ but not enough confidence for $A \wedge B \Rightarrow D$. On the other hand, if we *exclude* C, then there is both sufficient support and confidence for the rule. We can conclude that "A and B imply D when C does not occur".

Such *excluding associations* are meaningful and important, yet the current algorithms are incapable of producing them.

In this paper we present a novel idea for generating associations. Our method uses subword tree techniques to mine for associations and is fundamentally different from all previously known algorithms. Our algorithm can handle very small supports, handles dynamic databases, and finds associations with negations.

Our algorithm's worst case time is exponential in the maximum cover size. In practice the size of the maximal cover is rather small (7 in the Reuters-22173 database with support 10, which is equal to 0.045%), while the number of attributes or the record size may be much larger (587 and 26, resp. in the Reuters-22173 database).

In addition, our algorithm reads the database only once. Since the database is normally on external memory, the main time devoted to the algorithm is the I/O time. Consequently, the fact that our algorithm reads the data only once significantly reduces the running time. As a result, the small exponential multiple on the number of machine instructions is not critical. Like all association generation algorithms, our algorithm is very efficient when the data structures it uses reside in memory. The compact representation we use helps keep the extra data structure small relative to the amount of information it provides.

Our method is especially suitable for textual databases, where there is a very large number of attributes, but the length of an individual transaction is small. Under such circumstances it scales reasonably well. For example, if the maximum cover size is 5, then the generated auxilliary databases will be roughly 30 times larger than the initiall database. In addition, we only handle binary attributes, where for every transaction either an attribute exists or it does not.

2 Problem Definition

The following is a formal statement of the problem based on the description of the problem in [2, 3, 11]. Let $I = \{i_1, i_2, ..., i_m\}$ be a set of *attributes*, also called *items*. Let D be a set of variable length transactions over I. Each transaction contains a set of items $\{i_i, i_j, ..., i_k\} \subset I$. A set of items is called an *itemset*. The number of items in an itemset is the *length* (or the *size*) of an itemset. An itemset of length k is referred to as a $k - itemset$.

An *association rule* is an implication of the form $S_1 \Rightarrow S_2$, where $S_1, S_2 \subset I$, and $S_1 \cap S_2 = \emptyset$. S_1 is called the *antecedent* of the rule, and S_2 is called the *consequent* of the rule. We follow the literature [2, 3, 11] in denoting associations as an implication of sets of attributes. A generalization, explored by Mannila and Toivonen [10], is denoting associations as an implication of boolean formulae. Thus, a rule of the form $\{A_1, A_2, \ldots, A_n\} \Rightarrow \{B_1, B_2, \ldots, B_m\}$ will be denoted in [10] as $A_1 \wedge A_2 \wedge \cdots \wedge A_n \Rightarrow B_1 \wedge B_2 \wedge \cdots \wedge B_m$. The formal definition of an association rule appears below.

Each rule has an associated measure of statistical significance called *support*. For an itemset $S \subset I$, the *support* of S is the number of transactions in D that contain the itemset S. (Note that in the literature the support is usually measured as a given *fraction* of the database. However, we deal with extremely small supports and thus choose to define the support absolutely). We denote by $supp(S)$ the *support* of S. The support of the rule $S_1 \Rightarrow S_2$ is defined as $supp(S_1 \cup S_2)$. An itemset that has support greater than or equal to a specified minimum support threshold is called a *covering itemset* or shortly *cover*.

A rule $S_1 \Rightarrow S_2$ has a measure of its strength called *confidence* defined as the ratio of number of transactions in D that contain itemset $S_1 \cup S_2$ over number of transactions that contain itemset S_1.

The problem of mining association rules is to generate all rules $S_1 \Rightarrow S_2$ that have both support and confidence greater than or equal to some user specified minimum support (*minsupp*) and minimum confidence (*minconf*) thresholds respectively, i.e. for regular associations:

$$supp(S_1 \cup S_2) \geq minsupp$$

and

$$\frac{supp(S_1 \cup S_2)}{supp(S_1)} \geq minconf.$$

This problem can be decomposed into the following two subproblems:

1. All covers, i.e. itemsets that have the minimum support, are generated.

2. All the rules that have minimum confidence are generated in the following naive way: For every covering itemset S and any $S_2 \subset S$, let $S_1 = S - S_2$. If the rule $S_1 \Rightarrow S_2$ has the minimum confidence, then it is a valid rule.

If all covers and their supports are given, one can generate rules in a brute-force manner, by considering all partitions of the powerset of every cover. Thus, the existing literature concerned itself mainly with generating all covering itemsets and their supports. In section 3 we present an efficient algorithm for generating all covers. In addition, we show an efficient and fast method for generating all associations.

3 A New Approach

A *trie* is a data structure that allows a set of strings to be stored and updated, and allows membership queries. For a formal definition of a trie, see e.g. [14]. We use the *trie* data structure to generate covers. Using the trie enables generating covers independently of support. In addition, the information stored in the trie can be used later for exceedingly fast queries for new types of associations.

3.1 Using the Trie

Our idea is to *preprocess* the database and construct a data structure that encodes necessary information for association generation. Once that data structure is constructed there is really no further need to access the original database. The new data structure can then be used to generate various different types of associations. We also show that updating the new data structure when the database is changed is a fast and simple process.

In the preprocessing phase we construct a trie of the database. Since the database entries are *sets* of attributes and a trie is defined on *strings* some conversion is necessary. The first step is numbering the different attributes. We then consider every set as a string sorted by order of the attribute number.

We would like every node on the trie to be a potential cover. Unfortunately, every subset of a record is a potential cover. It would seem that an exponential space is required, which would make the idea prohibitively expensive for any but a database with very small records. However, the situation turns out to be a lot better than that. Because covers are usually not large (at most 7, in the Reuters-22173 even for the very small support of 10) we can set a relatively small value k as the maximum cover size and only consider subsets of size up to k for inclusion in the trie. Thus, for a record of m attributes, where k is the value taken as the bound of the largest cover, we consider only the $\sum_{i=1}^{k} \binom{m}{i}$ different i-element subsets of the m attributes, for $i = 1, \ldots, k$.

At the colclusion of the preprocessing phase we have a data structure encoding all potential covers and the number of transactions that include those sets among their attributes, with an extremely fast and efficient access method to every cover. The data structure is dynamically updated. Its size is quite manageable (see experimental results) and it encodes data that allows instantaneous recognition of covers, efficient support-independent generation of associations, as well as generation of excluding associations.

4 Association Generation

Many current algorithms generate association rules by the exponential-time naive method of testing the confidence of all subsets of the cover. The following lemma provides an idea for more efficient generation of rules from covers.

Lemma: Let S, S_1, S_2 be *mutually disjoint* sets of attributes.
If $S \Rightarrow S_1 \cup S_2$ is an association rule, *then* also $S \cup S_1 \Rightarrow S_2$ and $S \cup S_2 \Rightarrow S_1$ are association rules.
Proof: Easy by *Venn Diagrams.*

The meaning of the lemma is that once all associations of the form $S \Rightarrow \{a\}$ have been generated, where a is a single attribute, we can narrow down the space of potential attributes of the form $S \Rightarrow \{a, b\}$. In particular, only if *both* associations $S \cup \{a\} \Rightarrow \{b\}$ and $S \cup \{b\} \Rightarrow \{a\}$ exist, is there any chance for $S \Rightarrow \{a, b\}$ to exist. A similar argument holds for larger sets in the right hand side of the association.

5 Excluding Associations

We previously defined the meaning of $supp(S)$ where S is a set of attributes and $supp(S_1 \Rightarrow S_2)$ where $S_1 \Rightarrow S_2$ is a *regular association.*

Let S be a set of attributes $\{A_1, \ldots, A_n\}$. Denote the set $\{\neg A_1, \ldots, \neg A_n\}$ by $\neg S$.

Excluding associations are of the form $S_1 \cup \neg S_2 \Rightarrow S_3$, where S_1, S_2, S_3 are mutually disjoint. Recall that the intuitive meaning of $S_1 \cup \neg S_2 \Rightarrow S_3$ is that the attributes in S_1 imply the attributes in S_3 **provided** that the attributes in S_2 **do not** appear in the transaction.

In the boolean formula notation such a rule will be written as $A_1 \wedge A_2 \wedge \cdots \wedge A_n \wedge \neg B_1 \wedge \neg B_2 \wedge \cdots \wedge \neg B_m \Rightarrow C_1 \wedge C_2 \wedge \cdots \wedge C_\ell$, where $S_1 = \{A_1, \ldots, A_n\}$, $\neg S_2 = \{\neg B_1, \ldots, \neg B_m\}$ and $S_3 = \{C_1, \ldots, C_\ell\}$.

Definition: We say that $S_1 \cup \neg S_2 \Rightarrow S_3$ is an *excluding association* if S_1, S_2, S_3 are mutually disjoint sets of attributes and the following conditions hold:

$$(E_1)\ \frac{supp(S_1 \cup S_3)}{supp(S_1)} < minconf$$

$$(E_2)\ supp(S_1 \cup S_3) - supp(S_1 \cup S_2 \cup S_3) \geq minsupp, \text{ and}$$

$$(E_3)\ \frac{supp(S_1 \cup S_3) - supp(S_1 \cup S_2 \cup S_3)}{supp(S_1) - supp(S_1 \cup S_2)} \geq minconf.$$

In the rest of this paper we will only consider excluding associations where S_2 and S_3 have a single attribute. It should be noted that our algorithm can be generalized to larger sets by considering all subsets of a cover. We did not implement this direction because of time complexity considerations.

An excluding association can be located by a depth first search (DFS) on the trie, in a similar manner to the DFS performed for finding associations. The only difference is that for every path we need to pick out the pairs $\{a\}$ and $\{b\}$ as S_2 and S_3 for which conditions (E_1), (E_2) and (E_3) hold.

6 Experimental Results

We ran our tests on the Reuters-22173 database. The Reuters-22173 text categorization test collection is a set of documents that appeared on the Reuters newswire in 1987. The 22173 documents were assembled and indexed with categories by personnel from Reuters Ltd. and Carnegie Group, Inc. in 1987. Further formatting and data file production was done in 1991 and 1992 by David D. Lewis and Peter Shoemaker. The documents were tagged with 587 attributes. We treat each document as a single transaction, where the attributes are the keywords with which the document is tagged. The size of the Reuters-22173 textual database is 25mb. The size of the Reuters-22173 ASCII attribute set is 1mb.

The platform we used was an UltraSPARC 1000 with 64mb main memory.

6.1 Trie Construction

We constructed three tries. One was a full trie with all subsets of every transaction. The other tries had maximum cover sizes 10 and 7. The following table summarizes the time and space it took to construct each of the tries. In the table below, the first row is the size of the trie, the second row is the time it took to construct the trie, and the third row is the construction time and saving the trie on the disk for future use.

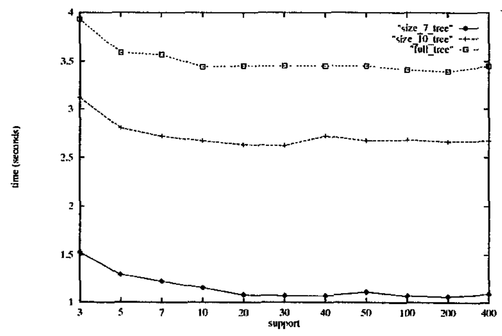
	All Subsets	*Max Cover Size* 10	*Max Cover Size* 7
Space:	4.61 mb	3.57 mb	1.45 mb
Time, construction only:	33.5 sec.	21.8 sec.	10.9 sec.
Time, including save:	35.5 sec.	23.1 sec.	11.6 sec.

In all the following subsections, we start out with an existing trie on disk. The times quoted ***include disk access.***

6.2 Generating All Covers

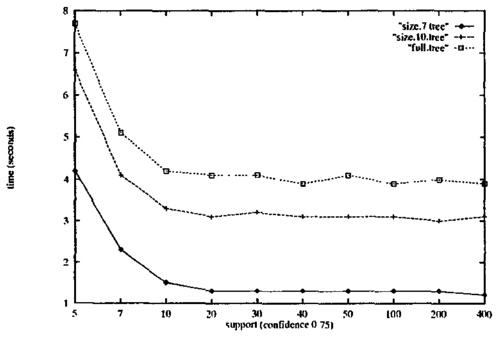
Generating all covers simply implies DFS of the tree, independently of the support. In the graph below one can see the running times when using the tree with maximum cover size 7, the tree for maximum cover size 10, and the full tree. As can be seen, there is practically no difference in the running time for different supports. The only slight rise is where the support is 3. Note that we are talking about ***absolute 3***, every cover that appears even three times was generated! Even the running time for support 3 (0.01%) was very small, 3.93 seconds on the full trie. The reason there is some increment for such a minute support is that now indeed a full DFS took place. The way we constructed the trie allows the scanning to take some advantage of support. The scanning stage does not really scan the entire trie. It only visits the nodes with large enough support.

It is important to note that previous algorithms in the literature did not plot such small supports because of their ***exponential time growth.*** Another important note is that the running time includes disk access to the trie. If the entire trie resides in memory, the running time is several hundred ***miliseconds.***

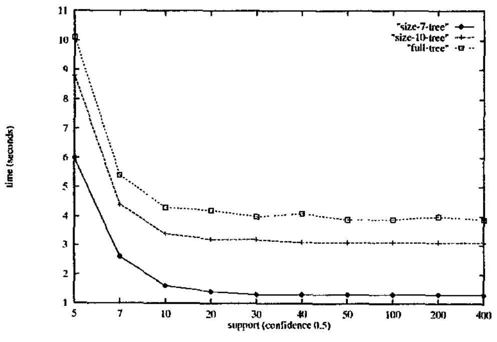


6.3 Generating all Associations

The following graph represents the time it took to generate all associations with various different supports. The confidence ratio taken in all cases was 75%. As can be seen, the graph shows that the support plays no role in the association generation, except for a slight time increase for ridiculously small supports.



The following graph is similar to the one above but the confidence is taken at 50%. Note that for all reasonable supports (even support 10, which is 0.045%) the results seem to be not only support-independent, but also confidence-independent.



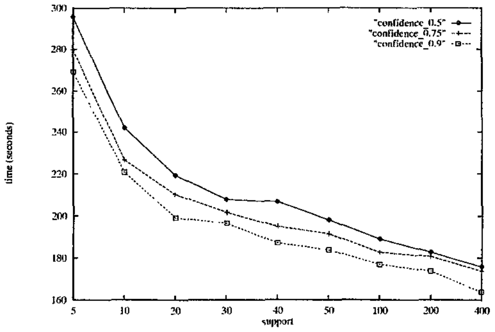
6.4 Generating Excluding Associations

To our knowledge, our algorithm is the only one that can generate excluding associations. The graph below indicates the time it took to generate excluding associations with the different supports. The trie used was the one for maximum cover size 7. The different plots represent the three different confidence ratios used, 90%, 75%, and 50%..

The table below shows the number of associations and the number of excluding associations with a single exclusion and one consequent. The confidence ratio appears as a percentage in parentheses. It should be noted that all excluding associations we count are new. They are not derived from regular associations. In other words, if $S \Rightarrow \{a\}$ then we do not count $S \cup \{\neg b\} \Rightarrow \{a\}$ for any b not in S. This table shows that computing the excluding associations is quite important since there are a significant number of such associations. In addition, the table also shows the running time of our algorithm to produce all regular associations, and all excluding associations with one exclusion and a single consequent.

Support:	5	10	20	30	40	50	100	200	400
Regular Associations (90%):	7321	733	241	72	34	24	6	3	0
Excluding Associations (90%):	1047	293	72	26	16	10	0	0	0
Time (seconds, conf: 90%):	269.2	220.8	199.1	196.6	187.1	183.8	176.9	173.8	163.8
Regular Associations (75%):	9899	1070	366	144	62	45	13	6	2
Excluding Associations (75%):	833	288	80	37	17	4	0	0	0
Time (seconds, conf: 75%):	279.7	226.6	210.1	201.6	195.0	191.5	182.6	180.7	173.8
Regular Associations (50%):	13217	1681	617	279	139	91	29	9	3
Excluding Associations (50%):	525	173	81	40	31	24	5	3	0
Time (seconds, conf: 50%):	295.7	242.0	219.3	207.7	206.6	198.1	189.0	182.7	175.8

The running times are plotted on the following graph.



While the time to generate excluding associations is not nearly as good as the time to generate associations, it is still manageable. To our knowledge, this is the first program that can generate excluding associations. Unfortunately, we can not claim that we generate the excluding associations with a support-independent time complexity. However, the situation is not awful. Examining the data, one can see that the ratio of the time it takes for support 100 over the time for support 10, is merely 80%, even though there is an order of magnitude difference between the supports.

7 Conclusions and Open Problems

We have shown a new method for generating associations. Our method enables finding associations with extremely small supports, naturally allows finding covers in growing and shrinking databases and is the first known algorithm to generate excluding associations. At the heart of our method was using subword trees for succinct encoding and efficient retrieval of information. Our algorithm performed satisfactorily on the Reuters-22173 database.

From a theoretical point of view, all existing algorithms, including the ones we presented here, have an exponential bottleneck at some stage. Our algorithm reaches this bottleneck rather fast. It would be an extremely important contribution to develop a method that would only have a polynomial worst-case complexity. We feel that the literature of subword trees may provide answers in this direction. We also feel that further study will allow achieving faster generation of excluding associations.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Database mining: a performance perspective. *IEEE Trans. Knowledge and Data Engineering*, 5(6):914–925, 1993.
2. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD*, pages 207–216, Washington, DC, May 1993.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. 20th Int'l Conf. on VLDB*, Santiago, Chile, Aug 1994.
4. R. Feldman, A. Amir, Y. Aumann, A. Zilberstein, and H. Hirsh. Incremental algorithms for association generation. to appear, First Pacific Conference on Knowledge Discovery, July 1996.
5. R. Feldman and I. Dagan. Knowledge discovery in textual databases. *Proc. 1st Intl. Conf. on Knowledge Discovery and Data Mining*, pages 112–117, 1995.
6. R. Feldman, I. Dagan, and H. Hirsh. Keyword-based browsing and analysis of large document sets. In *Proc. 5th Symp. on Document Analysis and Information Retrieval*, Las Vegas, Nevada, April 1996.
7. R. Feldman, I. Dagan, and W. Kloesgen. Efficient algorithms for mining and manipulating associations in texts. In *Proc. 13th European Meeting on Cybernetics and Systems Research*, Vienna, Austria, April 1996.
8. W. Kloesgen. Problems for knowledge discovery in databases and their treatment in the statistical interpreter explora. *Int'l J. for Intelligent Systems*, 7(7):649–673, 1992.
9. W. Kloesgen. Efficient discovery of interesting statements. *The Journal of Intelligent Information Systems*, 4(1), 1995.
10. H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. *Proc. 2nd Int'l Conference on Knowledge Discovery in Databases*, 1996.
11. H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. *Proc. AAAI Workshop on Knowledge Discovery in Databases*, pages 181–192, 1994.
12. G. Piatetsky-Shapiro and W. J. Frawley, editors. *Knowledge Discovery in Databases.* AAAI Press/MIT Press, 1991.
13. A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. *Proc. 21st Int'l Conf. on VLDB*, 1995.
14. R. Sedgewick. *Algorithms.* Addison-Wesley, second edition, 1988.