International Journal of Computational Intelligence Systems, Vol. 5, No. 3 (June, 2012), 434-451

Mining Hierarchical Negative Association Rules

David Taniar [1], Wenny Rahayu [2], Olena Daly [1], Hong-Quang Nguyen [3]

[1] *Clayton School of Information Technology, Monash University, Australia*
E-mail: {david.taniar, olena.daly}@monash.edu.au
[2] *Department of Computer Science and Engineering, La Trobe University, Australia,*
E-mail: w.rahayu@latrobe.edu.au
[3] *School of Computer Science and Engineering, International University - VNU-HCMC, Vietnam*
E-mail: nhquang@hcmiu.edu.vn

Received 8 February 2012
Accepted 14 February 2012

Abstract

The negative association between items in databases is as important and interesting as the positive one. But, it has not been studied as much. We consider negative association in a hierarchical setting, in which we are able to generate negative association rules at different hierarchy levels. It allows to impose restrictions when we proceed to the next level and discover only most interesting negative association rules among the vast number of possible negative association rules. In this paper, we propose two algorithms for mining negative association rules by considering that items are organized in a hierarchy, and this hierarchy is reflected on the association rules we produce. In this way, we can mine for both general and specialized rules of negative association between items.

Keywords: Association Rules, Negative Association Rules, Hierarchical Association Rules, Frequent Itemset, Patterns, Data Mining

1. Introduction

One of the most studied techniques of data mining is *association rules*, which have been applied successfully to market basket [1,2,3,4,5,6,7]. Association rules traditionally consider only positive association, like items purchased together, or words appearing together in a document, etc. Another interesting approach of mining association rules is to seek *negative association*. In other words, we could discover which items customers tend *not* to buy together, or which events would *never* happen together. For instance, we could discover that customers buying tea do not usually buy coffee. Negative association rules discover valuable negative correlations between database items. For example, a negative association between competitive brands of the same product may be discovered. Such a negative association would suggest targeted marketing policies for the different groups of customers.

In negative association, we consider the *presence* and *absence* of items in the database records. So, negative association rule mining will produce a vast number of rules in comparison with positive association. In order to reduce the number of rules generated by the negative association rule mining, we need to consider the *hierarchical* nature of the database items [8]. This way will allow differentiating the more general negative association rules from the interesting ones. In traversing up the hierarchy of

items, it will be possible to impose some limitations that enable us to discover negative association rules with higher value of interest.

The *hierarchical* approach provides us with both extensive and general knowledge. Consider Figure 1 as an example. If we were mining rules on a single level, we would only find rules of `Beef` $\Rightarrow$ `Trout` or `Beef` $\Rightarrow \neg$ `Trout` at the bottom level of the hierarchy. We use $\neg$*Expr* to denote the *negation* of the expression *Expr*; that is, $\neg$ `Trout` should read `Not Trout`. Applying hierarchical mining, we, in addition, discover rules of a type `Fish` $\Rightarrow$ `Meat` or `Juice` $\Rightarrow \neg$ `Meat` at a more general level. Hence, we could obtain more informative and general rules, which is crucial in the case of vast negative association rules.

The aim of this paper is to explore negative association rules and to apply the hierarchical approach to distinct the most informative rules. A number of algorithms have been developed and tested. The results will also be presented.

2. Related Work

Negative association rule mining has been addressed in a few research works [9,10,11,12,13]. The negative association rules mining methods could be divided into two categories. In the category I, a special measure for negative association rules is created. In category II, the data or the first generated itemsets are analyzed to further produce only most interesting rules with less computational costs. Table 1 presents the categories of negative association rules mining.

The works [9,13] belong to the category I. A measure to distinguish the most valuable negative rules is employed. The interest measure is applied to negative rules [13]. The *interest* measure was first offered for positive rules. The measure is defined by a formula: $Interest(A \Rightarrow B) = support(AB) - support(A) \times support(B)$. The negative rules are generated from infrequent itemsets. The mining process for negative rules starts within 1-frequent itemsets. Although they studied negative association rule, they did not consider a multiple-level hierarchical settings.

Another way to search negative association rules is to employ statistical theories [9]. Correlations between database items are calculated. If the correlation is positive, a positive association rule will be obtained. If the correlation is negative, a negative association rule will be obtained. The stronger is the correlation, the stronger is the generated rule. In this approach, Chi-squared statistic is employed for mining positive and negative association rules. Each different database transaction in the market basket denotes a cell in a contingency table. The chi-squared statistics is calculated (which is in short a normalized deviation from expectation) for each cell. From the obtained value the strength of the correlation between database items is estimate [14]. The work has the following limitations. The Chi-squared test cannot be applied to some of the database records. The Chi-squared test should be applied only if all cells in the contingency table have an expected value greater than 1. The market baskets, with an expected value smaller than 1, are simply ignored, which may cause the result to be less trustworthy. Besides they do not consider the multiple-level concept in their approach, which is why it is different from our research.

The works of Boulicaut, Bykowski and Jeudy (2000) and Fortes, Balczar and Morales (2001) fall into category II. The data is analysed in the initial step and the obtained information is employed on the next steps of generation process.

Boulicaut, Bykowski and Jeudy (2000) extract negative association rules from the database using only the information about the positive attributes of association rules. Moreover, a set of constraints, namely anti-monotone constraints and monotone constraints has been proposed to mine association rules with negations. When the frequent items are generated, the set of proposed constraints is employed at every step to distinguish the interesting association rules with negations.

Fortes, Balczar and Morales (2001) first discover frequent negative itemsets that contain n items and only 1 negation of an item, for example $\{ABC \sim D, AB \sim CD\}$, where $\sim C, \sim D$ means absence of the item C or D. Then the produced rules with n items and 1 negation are analysed. The obtained information allows disregarding some itemsets on the next step when producing rules with n items and 2

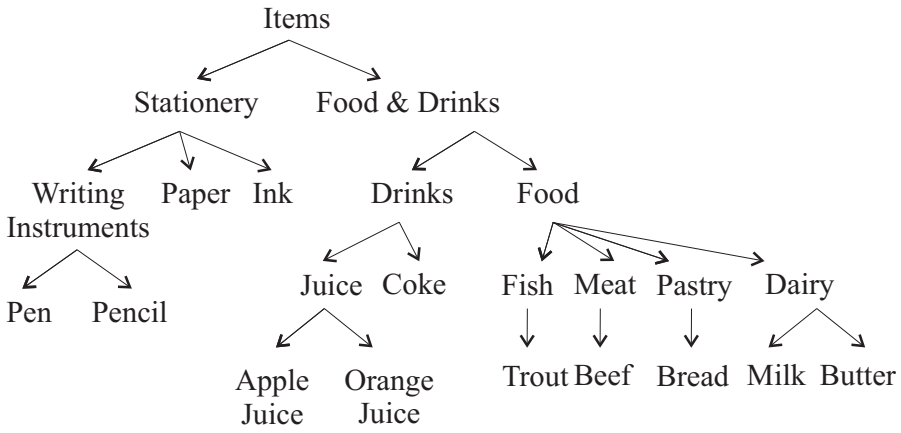


Figure 1: Hierarchy of Items

Table 1: Negative association mining categories

Authors	**Category I**	**Category II**
Brin, Motwani and Silverstain (1997)	Special measure	
Wu, Zhang, C. and Zhang, S. (2002)	Special measure	
Boulicaut, Bykowski and Jeudy (2000)		Active constraints checking
Fortes, Balcazar and Morales (2001)		Generated rules analysis
Savasere, Omiecinski and Navathe (1998)	Interest measure	Data analysis (domain knowledge)

negations ($A \sim BC \sim D$). n takes on values from 1 to overall number of different items in the database. The negative rules mining stops when the maximum number of negations m in rules has been reached, m provided by the user, or when all negative rules have been generated, if m has not been provided. Additionally, they did not study the hierarchical nature of items.

In Savasere, Omiecinski and Navathe (1998), a combination of both a novel interest measure and domain knowledge analysis is employed. The work may belong to both categories. The items that belong to the same category are expected to have similar associations with other database items. For example, if `Coke` is frequently bought with `Chips`, then `Pepsi` is expected to be frequently bought with `Chips`. In case `Pepsi` is rarely bought with `Chips`, the negative association has been discovered. The authors make sure they do not have to scan all possible negative itemsets and try to distinguish the interesting ones. The search for negative association is based on the strong positive association of the items in the same category. Once a strong association rule has been discovered, the remaining items in the category will be verified to form the strong negative association. This approach certainly ensures that all generated rules will be highly interesting. However, only a small fraction of interesting negative association rules can be extracted in this way. Only when two items in the same category form both positive and negative association with the same itemset, can the negative association rule be detected. That is the disadvantage of the project and different approaches should be created to target the remaining negative association rules. The remaining rules may be as interesting as or more interesting than the generated ones. Only after evaluating both types of rules, can conclusions be drawn about the value of the generated and remaining negative association rules.

Another paper [15] studied multiple-level approach in association rules. They consider only positive association on different levels.

There has been some work on negative association in domains with very limited number of database attributes. A good example is a weather prediction. The database only contains 10-15 factors like fog, rain, sunshine, etc, on a particular day. In this kind of application, we can easily add the negative values to the database and treat them as additional database attributes. Our work is different as the database of market basket is vast (e.g. possibly thousands of different items) so requires a distinctive approach. Our approach can be applied to different domains where the number of items/attributes is not limited to a few dozens, such as sensor networks [16,17], and disease exploratory [18].

We may say that our approach is different from what has been done before. We consider both positive and negative association rules employing multiple-level hierarchical concept. The hierarchical approach is especially important for mining negative association rules. Negative association rules are vast and numerous comparatively to positive ones. The hierarchical approach allows reduce the number of negative frequent itemsets and distinct more valuable ones.

3. Proposed Algorithms

In this section, we propose two algorithms for mining hierarchical negative association rules. The first algorithm is called the "Basic Algorithm", and the second is the "Hierarchy Top-Down Algorithm". Before the two algorithms are described, we need to explain the following definitions.

3.1. Definitions

In order to understand negative association rules and its hierarchical setting, we need to formalize the following definitions.

Definition 1: *Negative Association Rule* $A \Rightarrow B$ is a kind of association rules that considers both presence and absence of items in the database record.

Itemsets A and B may contain both items and their negations. For example, we have frequent 1-itemsets a and b. For negative association we consider a, b, $\neg a$, $\neg b$ and itemsets (a,b), $(a,\neg b), (\neg a,b), (\neg a,\neg b)$. The data mining includes both positive and negative association rules search.

Definition 2: An itemset I is $frequent$ at level l, if the support of I is no less then the corresponding

minimum support value $minsup[l]$. The confidence of rule $A \Rightarrow B$ is high at level l, if its confidence is no less than its corresponding minimum confidence level $minconf[l]$.

Different values of minimum support and confidence at different levels of hierarchy are easy to explain. If the rule $\{Meat \Rightarrow \neg Dairy\ Products\}$ holds at support level 40% then for the rule $\{Meat \Rightarrow \neg Milk\}$, the minimum support value should be decreased to, for instance, 20%, as `Dairy Products` is the ancestor of `Milk` and its support is accumulated from all its descendants.

Definition 3: An *ancestor* of an item is a node in the database hierarchy tree, which is located higher than the item in the hierarchy and there is a route connecting the item and the ancestor. An item may have many ancestors.

Definition 4: Rule $A \Rightarrow B$ is strong if each ancestor of every item in A and B is frequent at its corresponding level, and the itemset AB is large at the current level and the confidence of $A \Rightarrow B$ is high at the current level.

Definition 5: Given the items hierarchy tree a *category* may be formed by any of its sub-trees.

Examples of a category may be Food, Stationary, and Footwear.

There are a vast number of negative association rules in comparison with positive association. Some criteria have been applied to narrow down the search space of negative rules. We seek for negative association only within frequent items (itemsets). In this way meaningless rules will be pruned away. Let us assign the minimum support value to 10% and say an item (e.g. `Nails`) has only 1% support in the database. We obtain rules of a type $Milk \Rightarrow \neg Nails, Hammer \Rightarrow \neg Nails$. The main reason the rules have a high support is that the item Nails is rare. The generated rules may create an illusion the foregoing items are never brought together.

The rules will be searched for within a category, for instance, within Food, or Stationary, or Clothes etc. Rules of a type $Milk \Rightarrow \neg Nails$ may be hard to comprehend and are not of a great value for a marketing policy. At the same time, rules of a type $\{Fish \Rightarrow \neg Meat\}$ could advise that some of the customers are vegetarians, or rules of a type $\{GreenTea \Rightarrow \neg Coffee\}$ could advise about the preference for green tea and no coffee. The two above-mentioned rules found together in a group of customers may suggest the group of customers tend to incline to health food. More health food products targeting the group may be offered in a marketing campaign.

In Table 2, we present an example of support values of itemsets $\{Meat, Fish\}$, $\{Meat, \neg Fish\}$, $\{Fish, \neg Meat\}, RR\{\neg Fish, \neg Meat\}$, where the number of database records equals 10, minimum support is 4 (records) and minimum confidence is 50%. The rule $\{Fish \Rightarrow \neg Meat\}$ has support 5 and confidence $5/7 = 71\%$.

3.2. The Basic Algorithm

In order to describe our Basic algorithm, consider a sample database hierarchy (taxonomy) presented in Figure 1 earlier. Normally, the database contains the data about the purchases of the specific items, represented by the bottom level of the hierarchy (`Pen, Pencil, AppleJuice, OrangeJuice, Trout, Beef, Bread, Milk, Butter`). Table 2 presents an extract of five transactions from a database with a total of nine items.

The hierarchies (taxonomies) are stored in a separate file. Database transactions change dynamically as new purchases are made by customers. The hierarchy, once defined by a domain expert, stays constant and stored in a read-only file. The hierarchy file is accessed when it is necessary to trace an item's hierarchical relations. One way of storing the hierarchical information is by creating a table where each row contains both an item and all its ancestors (refer to Table 3).

Our goal is to discover association rules not only between the specific items, but also those association rules (both positive and negative) on the higher taxonomy levels. A straightforward approach for calculating the support for items at all levels is described as follows. While scanning the database to update the support of a specific item (the bottom hierarchy level item), we would have to traverse the hierarchy tree every time this item has occurred in the database transaction to update the counters of the item's ancestors.

Table 2: An Example of Negative Association (Meat, Fish)

	Meat	**¬ Meat**	Σ()
Fish	2	5	7
¬Fish	2	1	3
Σ()	4	6	10

Table 3: A sample database

Transaction ID	**Items**
1	*AppleJuice*, *Pen*
2	*OrangeJuice*, *Beef*, *Pencil*, *Pen*
3	*Beef*, *Bread*, *Trout*, *AppleJuice*
4	*Milk*, *Butter*, *Pen*, *Pencil*
5	*Butter*, *Trout*, *Beef*, *OrangeJuice*, *AppleJuice*

For instance, while scanning the database in Table 2, the item `AppleJuice` has been detected in the first transaction (transaction #1). The counter of the `AppleJuice` has been automatically updated. Now, to update the counters of the `AppleJuice`'s ancestors, the hierarchy tree in the Figure 1 has to be traversed to the very top level. Figure 2 shows all ancestors of the item `AppleJuice` (in bold).

Extracting the ancestors of the item `AppleJuice` from the taxonomy file (refer to Table 3), we obtain the following ancestors: `Juice, Drinks, Food&Drinks, Items`. The purchase of the specific item `AppleJuice` means that the counters of all those general items are to be increased. The item `AppleJuice` belongs to the category `Juice`, category `Drinks`, category `Food&Drinks`, and category `Items`. All these categories have to be updated in order to reflect the recent change of the `AppleJuice`'s counter.

Instead of identifying all ancestors of an encountered database item during database scan from additional data sources (refer to Table 2), there is a more efficient way of traversing the items' hierarchical relationships. The hierarchical information can be embedded directly into the database. Then no additional data source is required. In order to do so, ancestors of each transaction item can be inserted into the same transaction. While reading the transaction, the counters of the item and all its ancestors will be automatically updated.

The technique would improve the efficiency of the mining algorithm. A database scan is required before the mining process in order to alter each database transaction with the items' ancestors. The additional database scan in the pre-processing stage allows for the discovery rules at different hierarchy levels with no information being required from additional data sources.

In Table 4 an example of a database record is presented. In order to mine for rules at different levels of the hierarchy, the database record will be modified, which is presented in Table 5.

The Basic algorithm generates all negative association rules at different levels simultaneously. In order to do this, the original database will be altered. All hierarchy ancestors of the transaction items will be added to the same transaction. One of the approaches to calculate hierarchical multiple-level rules is to add to the record all ancestors of all items, so that we calculate support of items on different levels at the same time. The Basic Algorithm is presented in Figure 3.

3.3. The Hierarchy Top-Down Mining Algorithm

Another approach for mining negative association rules is to start from the top of the hierarchy and deepen the search only if the parents have the nec-

Table 4: Hierarchy representation in a table

Item	Item's ancestors
AppleJuice	*Juice, Drinks, Food&Drinks, Items*
Pen	*WritingInstruments, Stationery, Items*

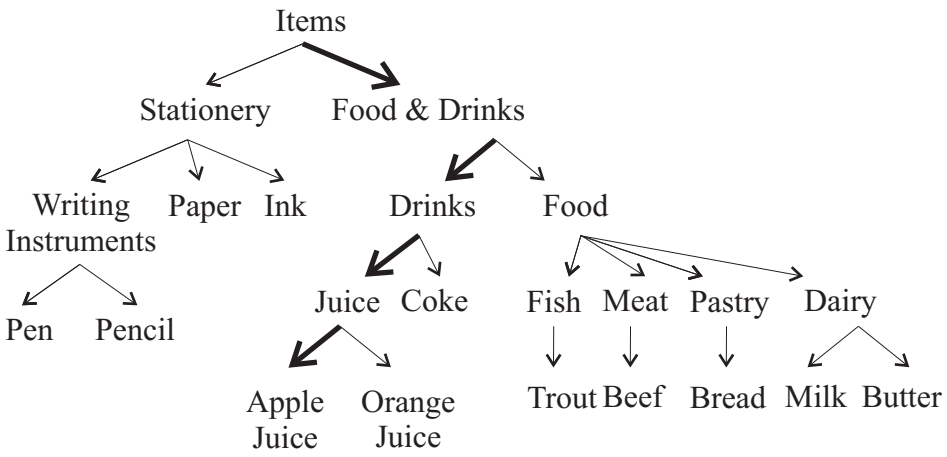


Figure 2: `AppleJuice`'s ancestors

Table 5: A database transaction

Transaction ID	Items
1	*AppleJuice*, *Pen*

Table 6: The modified database transaction

Transaction ID	Items
1	*AppleJuice, Juice, Drinks, Food&Drinks, Items, Pen, WritingInstruments, Stationery, Items*

Algorithm: *Basic* Hierarchical Negative Association Rule Mining

```
k:=1;   // k is the pass number
For all transactions t in D
{
    Add all ancestors of all items in t to t, and remove any duplicates
    Count 1-itemsets occurrence
}
N1:= {frequent 1-itemsets ∪ their negations};
k:=2; // k is the pass number
while (Nk-1 ≠ 0)
{
    Ck' := New candidates of size k generated from Nk-1
    For all transactions t in D
    {
        Increment the count of candidates in Ck'
            if (the positive items of the candidate are contained in t
                and negative items are not contained in t)
    }
    Nk:= All candidates in Ck' with min support
    k:=k+1;
}
```

Figure 3: The Proposed Basic Algorithm

essary minimum support. That is, for example, only if the rule $\{Meat \Rightarrow \neg DairyProducts\}$ is strong we will check if the rules $\{Beef \Rightarrow \neg Milk\}, \{Lamb \Rightarrow \neg Yogurt\}$ are strong. This will narrow down our search significantly as we do not have to check all rules to be strong. Note that the procedure of counting the support of the itemsets changes comparatively to the classical Apriori algorithm. Normally we seek the presence of items when we count the support of an itemset. When we mine for negative association, we are also interested in the absence. In other words when we count support of an itemset AB we look for the items A and B to be present in the transaction. In the case of negative association when we measure the support of $(A\neg B)$ we require A to be present in the current transaction and B to be absent in order to increase the itemset's $(A\neg B)$ support.

We adopt the algorithm proposed by Han and Fu (1995 [15]) for mining positive hierarchical multiple-level rules. We modify it for mining negative hierarchical multiple-level rules. The negations of items have been taken into consideration and the support-counting procedures have been modified. We also apply restrictions to make the generated rules more informative and user-appreciated.

The rules are searched for within a category. For instance, the data mining process is performed in the category *Food&Drinks*. We encode each item in the database with 3 digits as we consider 3 levels of hierarchy. For instance 111 would mean `Food-Meat-Beef`, 112 would mean `Food-Beef-Lamb`. In Table 6, a small encoded database with 9 transactions is presented.

Our frequent 1-itemsets contain both items and their negations. The minimum support is different on different levels of hierarchy. Figure 3.3 presents an example of Hierarchy Top-Down Algorithm execution.

We start mining from the top level. In the encoded table the first digit relies to the top level. It means that items that have their first digit 1 are in `Food` category, digit 2 are in `Drinks` category. First

Table 7: Encoded Transaction Table

TID	Items
T1	{111, 121, 211, 221}
T1	{111, 211, 222, 323}
T3	{112, 122, 411}
T4	{111, 121}
T5	{111, 122, 413}
T6	{111, 323, 524}
T7	{211, 323, 411, 524, 713}
T8	{211}
T9	{111}

we scan the database and count the number of items that start with the same digit, for instance 1. These items will be large 1-itemsets on the level 1 (top level). The minimum support at the first level is 4. After scanning the database we obtain items $\{1**\}$ and $\{2**\}$ with their support 7 and 4.

We filter out the rest of the items and get filtered table $T[2]$. We add negations of the frequent 1-itemsets if they have enough support and obtain $N[1,1]$. We use $N1[1,1]$ to obtain 2-candidates at level 1. In our example we have two 2-candidate. After we check the candidates' support, we discover large 2-itemset at level 1, which is $\{1**, \neg 2**\}$. When we scan the database to update the count of a positive item we seek the presence of the item in the database. When we update counts of items' negations we seek absence of the items in the database.

Now we can proceed mining on the next level 2. Only the items having enough support on the level 1 may be considered on the level 2. We take large 1-itemsets at level 1, which are $\{1**\}$, $\{2**\}$, $\{\neg 2**\}$ and consider their second digit, which describes the next level of the hierarchy. The minimum support on the level 2 is 3 so after scanning the database we obtain the large 1-itemsets on the level 2: $\{11*\}$, $\{12*\}$, $\{21*\}$, $\{\neg 22*\}$, $\{\neg 22*\}$. We use them to discover large 2-itemsets at the level 2. After we have discovered large 2-itemsets we use them for discovering large 3-itemsets at the level 2. After we have discovered large 3-itemsets we use them for discovering large 4-itemsets at the level 2. There is only one large 4-itemset at the level 2 so it is the end of mining at level 2.

The input of the level 3 is output of the level 2. We take the large 1-itemsets at the level 2 and consider their third digit to discover the large 1-itemsets at the level 3. We derive large 2-itemsets from large 1-itemsets. There is only one 2-itemset so this is the end of the mining at the level 3.

We have considered all three desirable levels and the output of the algorithm is the large itemsets at different levels of the hierarchy. The algorithm is presented at Figure 3.3.

Note: $N[L,1]$ contains 1-itemsets of level L and their negations. The procedures of generating candidates and counting their support in the filtered transaction table have been modified to take into consideration the absence of items. $NN[L]$ is the union of frequent itemsets of level L for all k.

4. Performance Evaluation

Two proposed algorithms, Basic and Hierarchy Top-Down algorithms, were implemented and tested. The test database was downloaded from a web site of Department of Computer Science, University of Regina, Canada (http://www.cs.uregina.ca). The data is a synthetic data created using random numbers conception. The file contains 10,000 records, 8 items, and 3 levels of hierarchy. The data file contains transactions database in an $m \times n$ matrix. Each transaction appears in a row. Columns are separated by a space and represent items. A 1 indicates that item is present in the transaction and a 0 indicates it is not. In the testing, we focus on a subcategory of products (e.g. food and drink category), and we

Level-1 minsup=4
N[1,1]

Itemset	Support
{1**}	7
{2**}	4
¬ {2**}	5

Level-1 large 2-itemsets:
N[1,2]

Itemset	S
{1**, ¬ 2**}	5

Filtered transaction table T[2]

TID	Items
T1	{111, 121, 211, 221}
T1	{111, 211, 221}
T3	{112, 122}
T4	{111, 121}
T5	{111, 122}
T6	{111}
T7	{211}
T8	{211}
T9	{111}

Level-2 minsup=3
Level-2 large 1-itemsets:
N[2,1]

Itemset	Support
{11*}	7
{12*}	4
{21*}	4
¬ {21*}	5
¬ {22*}	7

Level-2 large 2-itemsets:
N[2,2]

Itemset	S
{11*, 12*}	4
{11*, ¬ 21*}	5
{11*, ¬ 22*}	5
{12*, ¬ 21*}	3
{12*, ¬ 22*}	3
{¬ 21*, ¬ 22*}	5

Level-2 large 3-itemsets:
N[2,3]

Itemset	S
{11*, 12*, ¬ 21*}	3
{11*, 12*, ¬ 22*}	3
{11*, ¬ 21*, ¬ 22*}	5
{12*, ¬ 21*, ¬ 22*}	3

Level-2 large 4-itemsets:
N[2,3]

Itemset	S
{11*, 12*, ¬ 21*, ¬ 22*}	3

Level-3 minsup=3
N[3,1]

Itemset	S
{111}	4
{211}	4
¬ {211}	4
¬ {221}	7

large 2-itemsets:
N[3,2]:

Itemset	S
{111, ¬ 211}	5
{111, ¬ 221}	5
{ ¬ 211, ¬ 221}	5

large 3-itemsets:
N[3,2]:

Itemset	S
{111, ¬ 211, ¬ 221	5

Figure 4: Support of itemsets on different levels of hierarchy

Algorithm: *Hierarchy Top-Down* Negative Association Rule Mining
Input: Encoded transaction table (e.g. for category Food&Drinks),
Minsup[L] for each level L.
Output: Multiple-level positive and negative frequent itemsets
Procedure:

```
For(L:=1;N[L,1]≠0 and  L<max_level;L++)
{
    If (L=1)
    {
        N[L,1]:= large_1_itemsets
            And_Negations(T[1],L);
        T[2]:= filter_table (T[1], N[1,1]);
    }
    else
        N[L,1]:= large_1_itemsets(T[2],L);
    for (k:=2; N[L,k-1]≠0;k++)
    {
        Ck:=get_candidate_set(N[L,k-1]);
        For all transactions t in T[2] do
        {
            Ct:= get_subsets(Ck,t);
            For each candidate c do
                if (positive items of c are in t & negative items of c are not in t)
                    c.support++;
        }
        N[L,k]:= {c in Ck|c.support>=minsup[L]}
    }
    NN[L]:=Uk N[L,k];
    //frequent itemsets of level L (for all k)
}
```

Figure 5: Hierarchy Top-Down algorithm

consider three levels of hierarchy (see Figure 1). The minimum support value was varied. It is in the range of $[1\%, 50\%]$.

4.1. Performance Evaluation of the Basic Algorithm

The Basic algorithm has four phases:

1. Create candidates,
2. Create candidate hash tree,
3. Read transactions from the database and update the counter of candidate itemsets in the hash tree, and
4. Create large itemsets.

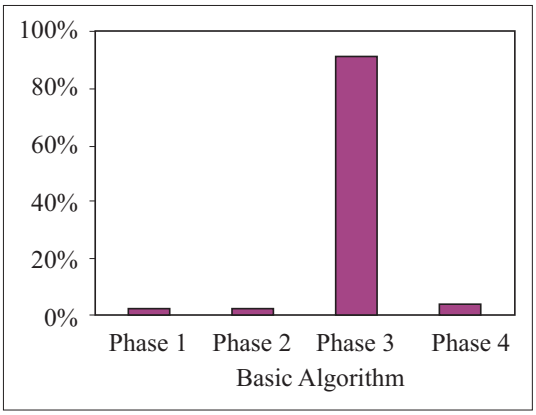
Figure 6(a) shows the proportion of time that each phase of the Basic algorithm takes to the overall execution time. It is clear that scanning of database takes slightly more than 90% of the execution time. It is a common fact for any database applications. The database scanning is commonly the most expensive operation.

In Figure 6(b), we explore in details what happens during the phases of mining hierarchical negative association rules in order to investigate the reasons of the time consumption. The execution time of $1^{st}, 2_{nd}$ and 4^{th} phases is significantly less than of 3^{rd} phase. So phases 1, 2 and 4 have been joined, we call it phase *A*, whereas phase *B* corresponds to phase 3. Figure 6(b) presents how phases *A* and *B* of the Basic algorithm behave with different *k*, where *k* is the current length of the itemsets (i.e. 1-itemset, 2-itemset, 3-itemset). In the example, the bar chart minimum support value equals to 30% and the largest *k* equals 5. The same kind of bar chart was observed for different support values and accordingly different values of *k*. Execution time in phase *A* is very stable. Execution time in phase *B* is the longest when the length *k* of the itemsets is medium relatively to the longest generated length of the itemset. As the frequent itemsets generation starts from 1-itemsets (itemset containing 1 item) to 2-itemsets, 3-itemsets, we generally observe a gradual increase in the number of itemsets. When the length *k* is medium the number of *k*-itemsets reaches its maximum and then gradually decreases. As opposed to the classical Apriori algorithm where the growth of the number of itemsets can be considered relatively slow, in the negative association rules the growth of the number of itemsets is extremely fast.

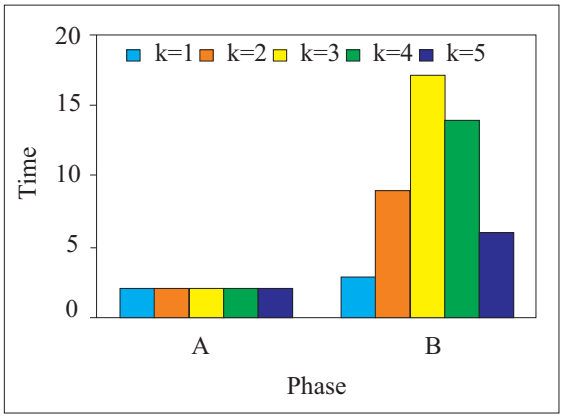
Figure 6(c) shows a graph of dependency between the minimum support value and number of generated frequent itemsets. The graph shows that when the value of minimum support (in %) increases, the number of frequent itemsets satisfying this minimum support decreases. On the contrary, when the value of minimum support decreases the number of frequent itemsets gradually decreases and there is a significant jump of frequent itemsets number when minimum support is low. The behaviour of the graph is easy to illustrate with an example. When the minimum support value equals, say 50%, only a few itemsets will occur in the half of the database records. So the number of frequent itemsets is very small. When the minimum support is 20%, we see there are quite a few itemsets that occur in one fifth of the database. When the minimum support value is in its lowest range, it is obvious from the graph that there occurs more and more itemsets satisfying the low requirements.

4.2. Performance Evaluation of the Hierarchy Top-Down Algorithm

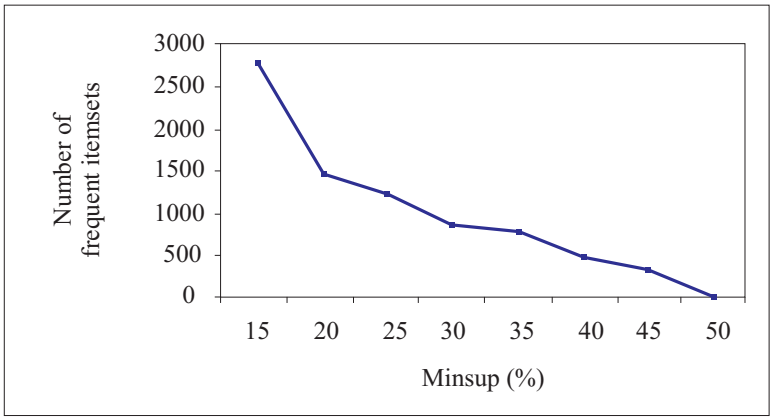
In the experimentations, we explored the *Food&Drink* category data and three levels of hierarchy (refer to Figure 7). The top level is marked as I, second as II and third as III. From Figure 7(a), we can see how the number of frequent itemsets at different levels changes with the increase of minimum support. The support value is different for different levels of hierarchy. It is greater at the top level and is decreased at the lower levels. The top level of hierarchy represents general items like `Food, Drinks`. The lower levels represent specific items like `Beef, Milk`. For example, the general category Meat consists of `Beef` and `Pork` at the lower level. If the minimum support at the level of `Meat` is 10% and Meat has a support of 10%, then when we proceed to the lower level of `Beef` and `Pork`, we must decrease the minimum support value as none of `Beef`



(a) Phases in the *Basic* Algorithm



(b) Phases with different *k*



(c) Minsup/number of frequent itemsets dependence for the *Basic* algorithm

Figure 6: Performance Evaluation of the Basic Algorithm

or `Pork` may have 10% support (unless one of them has 0% which is rare in the real databases). As we proceed from Level I to Level II, from Level II to Level III, we decrease the minimum support value. Figure 7(a) shows the three numbers that represent the values of minimum support at three levels.

When we analyse Figure 7(a) carefully, we can see that the bars representing Levels I and II are quite stable. We also observe that the bars of level III, which is the lowest level, changes significantly comparatively to the two top levels. The first value of support on Level III equals 1, which is quite low, and it explains the jump of the graph of the Level III. At 1% of minimum support, a significant number of frequent itemsets in the database satisfy the requirement. The graph in Figure 7(b) represents the distribution of itemsets of all levels according to their size (1-itemset, 2-itemset, 3-itemset). The majority of itemsets in this example are 8-itemsets because of the lowest value of minimum support for Level III. When the *minsup* equaled to 1% at Level III, the majority of the long frequent itemsets were generated. So the jump of itemsets number for $k = 8$ is explained by the relaxation of the *minsup* value to 1% at Level III.

4.3. Execution Times of Basic and Hierarchy Top-Down Algorithms

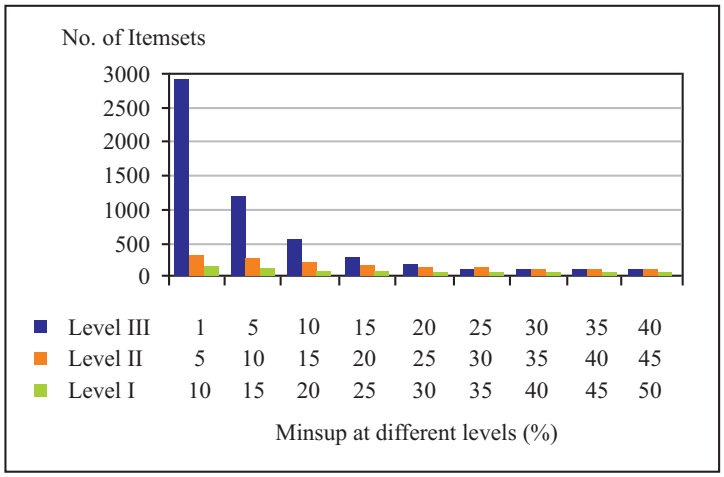
In this section, we compare the execution times of the Basic and Hierarchy Top-Down algorithms. Figure 8 presents a graph describing the behavior of the algorithms with the increasing of *minsup* value. For the Hierarchy Top-down algorithm, we take the average value of *minsup* for the three levels.

Figure 8 shows that execution time of Hierarchy Top-Down algorithm is significantly shorter than of the Basic algorithm. We start mining for the rules from the top of the hierarchy so we are able to prune invaluable itemsets already on the top level. Only the itemsets that have enough support at the top level are considered at the next lower levels. Besides, in the Basic algorithm, we were adding the ancestors of each item to the same transaction, which enlarged the length of the data base records and made the frequent itemsets generation slower. In the Hierarchy Top-Down algorithm, we encode the information about the item's parents in the item itself. Therefore we do not need additional items in the database for hierarchy tracing.

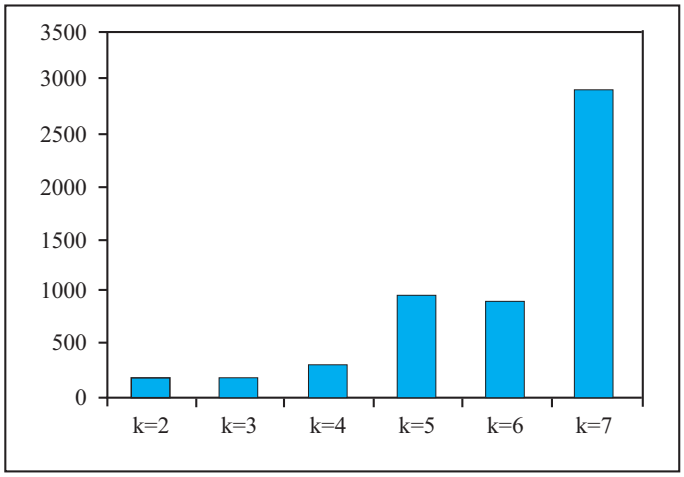
In the Table 8, we show the percentage of how much faster the Hierarchy Top-Down algorithm is in comparison with the Basic algorithm for each value of minimum support. We discover that in higher values of minimum support, the difference is less, as the computational requirements are moderate. In the lowest values of minimum support the jump between execution times of the algorithms is huge, as the number of generated itemsets and combinations is significant and the advantage of the Hierarchy Top-Down algorithm is doubtless. We may conclude that the Hierarchy Top-Down algorithm is much more efficient for mining negative multiple-level association rules in the databases. The algorithm adopts a better technique for database mining, generates quality result in more sufficient time, and requires less resources.

5. Conclusion

We have presented a new approach in the mining of association rules, particularly mining hierarchical multiple-level negative association rules. Hierarchy and negative mining is an important combination of mining techniques as it allows us to prune vast negative rules. Besides, general rules may be preferable especially in negative association mining. For instance, if we generate a rule $\{Beef \Rightarrow \neg Juice\}$, we may not be interested in what kind of juice (because "$\neg Juice$" is negation of any kind of juice). As opposite, in the positive association rule $\{Beef \Rightarrow Juice\}$, it is required to discover a specific kind of juice. The hierarchical approach allows us to generate rules more general, which is crucial for negative association. We are able to cease any further explorations, as we know no additional knowledge may be extracted on the next levels. We have proposed and implemented two algorithms to mine for hierarchical multiple-level negative association rules, which are called the "Basic" and "Hierarchy Top-Down". After we analysed the results, we may conclude that the Hierarchy Top-Down algorithm is more efficient and fast.



(a) Minsup/number of freq itemset



(b) Number of frequent *k*-itemsets

Figure 7: Hierarchy Top-Down algorithm

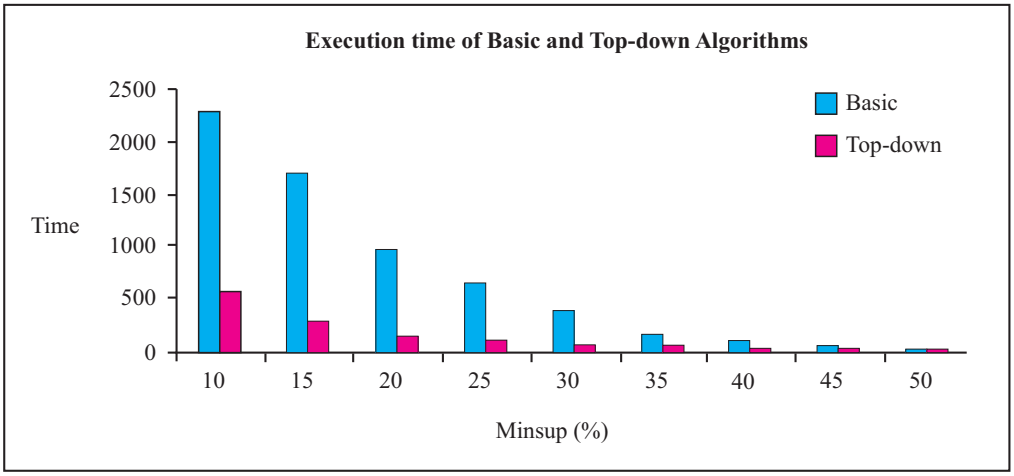


Figure 8: Basic and Top-Down Algorithms

Table 8: Basic vs Top-Down Algorithm

Minsup (%)	**Top-Down vs. Basic**
50	38%
45	47%
40	58%
35	69%
30	73%
25	79%
20	79%
15	80%
10	81%

Hierarchical negative association rules may be extended to cover other types of rules and patterns, including sporadic and exception rules [19,20]. New metrics, which have been used successfully in association rules and other pattern mining, such as composite datasets [21], item weight [22], weak ratio [23], binary data coefficient [24], similarity metrics [25], ontology [26], and continuous attribute discretization [27] can be applied to hierarchical negative association rules.

Our proposed research may also be applied to various domain applications, such as mobile systems [28,29,30,31,32,33,34], RFID [35,36], real-time systems [37], business process [38], web systems [39,40,41], and fuzzy data [42].

References

1. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile* (J. B. Bocca, M. Jarke, and C. Zaniolo, eds.), pp. 487–499, Morgan Kaufmann, 1994.
2. R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 207–216, 1993.
3. R. Srikant and R. Agrawal, "Mining generalized association rules," in *Proceedings of the 21st International Conference on Very Large Data Bases VLDB*, pp. 407–419, 1995.
4. M. Z. Ashrafi, D. Taniar, and K. A. Smith, "A data mining architecture for distributed environments," in *Proceedings of the 2nd International Workshop on Innovative Internet Computing Systems*, pp. 27–38, 2002.
5. M. Z. Ashrafi, D. Taniar, and K. A. Smith, "Redundant association rules reduction techniques," *International Journal of Business Intelligence and Data Mining*, vol. 2, no. 1, pp. 29–63, 2007.
6. H. C. Tjioe and D. Taniar, "Mining association rules in data warehouses," *International Journal of Data Warehousing and Mining*, vol. 1, no. 3, pp. 28–62, 2005.
7. A. Savasere, E. Omiecinski, and S. B. Navathe, "An efficient algorithm for mining association rules in large databases," in *Proceedings of the 21st International Conference on Very Large Data Bases VLDB*, pp. 432–444, 1995.
8. E. Pardede, J. W. Rahayu, and D. Taniar, "Object-relational complex structures for xml storage," *Information & Software Technology*, vol. 48, no. 6, pp. 370–384, 2006.
9. S. Brin, R. Motwani, and C. Silverstein, "Beyond market baskets: Generalizing association rules to correlations," in *Proceedings ACM SIGMOD International Conference on Management of Data*, pp. 265–276, 1997.
10. A. Savasere, E. Omiecinski, and S. B. Navathe, "Mining for strong negative associations in a large database of customer transactions," in *Proceedings of the Fourteenth International Conference on Data Engineering ICDE*, pp. 494–502, 1998.
11. J.-F. Boulicaut, A. Bykowski, and B. Jeudy, "Towards the tractable discovery of association rules with negations," in *Proceedings of the Fourth International Conference on Flexible Query Answering Systems FQAS*, pp. 425–434, 2000.
12. I. F. Ruiz, J. L. Balcázar, and R. M. Bueno, "Bounding negative information in frequent sets algorithms," in *Proceedings of the 4th International Conference on Discovery Science 2001, Washington, DC, USA, November 25-28, 2001*, pp. 50–58, 2001.
13. X. Wu, C. Zhang, and S. Zhang, "Mining both positive and negative association rules," in *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 658–665, 2002.
14. L. Tan and D. Taniar, "Adaptive estimated maximum-entropy distribution model," *Information Sciences*, vol. 177, no. 15, pp. 3110–3128, 2007.
15. J. Han and Y. Fu, "Discovery of multiple-level association rules from large databases," in *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*, pp. 420–431, 1995.
16. Y. Li, H. Chen, R. Xie, and J. Z. Wang, "Bgn: A novel scatternet formation algorithm for bluetooth-based sensor networks," *Mobile Information Systems*, vol. 7, no. 2, pp. 93–106, 2011.
17. E. Shakshuki, X. Xing, and T. R. Sheltami, "Fault reconnaissance agent for sensor networks," *Mobile Information Systems*, vol. 6, no. 3, pp. 229–247, 2010.
18. Z. Xing and J. Pei, "Exploring disease association from the nhanes data: Data mining, pattern summarization, and visual analytics," *International Journal of Data Warehousing and Mining*, vol. 6, no. 3, pp. 11–27, 2010.
19. O. Daly and D. Taniar, "Exception rules mining based on negative association rules," in *Proceedings of the International Conference on Computational Science and Its Applications (ICCSA 2004), Part IV, Lecture Notes in Computer Science*, pp. 543–552, 2004.
20. D. Taniar, J. W. Rahayu, V. C. S. Lee, and O. Daly, "Exception rules in association rule mining," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 735–750, 2008.

21. M. S. Khan, M. K. Muyeba, F. Coenen, D. Reid, and H. Tawfik, "Finding associations in composite data sets: The cfarm algorithm," *International Journal of Data Warehousing and Mining*, vol. 7, no. 3, pp. 1–29, 2011.
22. Y. S. Koh, R. Pears, and G. Dobbie, "Automatic item weight generation for pattern mining and its application," *International Journal of Data Warehousing and Mining*, vol. 7, no. 3, pp. 30–49, 2011.
23. B. Jiang, X. Hu, Q. Wei, J. Song, C. Han, and M. Liang, "Weak ratio rules: A generalized boolean association rules," *International Journal of Data Warehousing and Mining*, vol. 7, no. 3, pp. 50–87, 2011.
24. D. M. Lewis and V. P. Janeja, "An empirical evaluation of similarity coefficients for binary valued data," *International Journal of Data Warehousing and Mining*, vol. 7, no. 2, pp. 44–66, 2011.
25. P. Kumar, R. S. Bapi, and P. R. Krishna, "A new similarity metric for sequential data," *International Journal of Data Warehousing and Mining*, vol. 6, no. 4, pp. 16–32, 2010.
26. J. Kim, P. Kim, and H. Chung, "Ontology construction using online ontologies based on selection, mapping and merging," *International Journal of Web and Grid Services*, vol. 7, no. 2, pp. 170–189, 2011.
27. K. Engle and A. Gangopadhyay, "An efficient method for discretizing continuous attributes," *International Journal of Data Warehousing and Mining*, vol. 6, no. 2, pp. 1–21, 2010.
28. S. Jan, M. Li, G. Al-Sultany, H. S. Al-Raweshidy, and I. A. Shah, "Semantic file annotation and retrieval on mobile devices," *Mobile Information Systems*, vol. 7, no. 2, pp. 107–122, 2011.
29. F. Morvan and A. Hameurlain, "A mobile relational algebra," *Mobile Information Systems*, vol. 7, no. 1, pp. 1–20, 2011.
30. D. Taniar and J. Goh, "On mining movement pattern from mobile users," *International Journal of Distributed Sensor Networks*, vol. 3, no. 1, pp. 69–86, 2007.
31. J. Goh and D. Taniar, "Mining frequency pattern from mobile users," in *Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems KES'2004, Lecture Notes in Computer Science, Part III, Volume 3215, Springer*, pp. 795–801, 2004.
32. J. Y. Goh and D. Taniar, "Mobile data mining by location dependencies," in *Proceedings of 5th International Conference on Intelligent Data Engineering and Automated Learning - IDEAL'2004, Lecture Notes in Computer Science 3177 Springer*, pp. 225–231, 2004.
33. R. Trasarti, F. Giannotti, M. Nanni, D. Pedreschi, and C. Renso, "A query language for mobility data mining," *International Journal of Data Warehousing and Mining*, vol. 7, no. 1, pp. 24–45, 2011.
34. T. Delot, S. Ilarri, N. Cenerario, and T. Hien, "Event sharing in vehicular networks using geographic vectors and maps," *Mobile Information Systems*, vol. 7, no. 1, pp. 21–44, 2011.
35. C.-L. Chen, "Design of a secure rfid authentication scheme preceding market transactions," *Mobile Information Systems*, vol. 7, no. 3, pp. 201–216, 2011.
36. H.-H. Hsu and C.-C. Chen, "Rfid-based human behavior modeling and anomaly detection for elderly care," *Mobile Information Systems*, vol. 6, no. 4, pp. 341–354, 2010.
37. F. Xhafa, C. Paniagua, L. Barolli, and S. Caballé, "A parallel grid-based implementation for real-time processing of event log data of collaborative applications," *International Journal of Web and Grid Services*, vol. 6, no. 2, pp. 124–140, 2010.
38. J. F. M. Bernal and M. Morisio, "Towards a dynamic rule-based business process," *International Journal of Web and Grid Services*, vol. 6, no. 4, pp. 385–399, 2010.
39. T. Takashita, Y. Abe, T. Itokawa, T. Kitasuka, and M. Aritsugi, "Design and implementation of a system for finding appropriate tags to photos in flickr from web browsing behaviour," *International Journal of Web and Grid Services*, vol. 7, no. 1, pp. 75–90, 2011.
40. J. Oh, O.-R. Jeong, E. Lee, and W. Kim, "A framework for collective intelligence from internet q&a documents," *International Journal of Web and Grid Services*, vol. 7, no. 2, pp. 134–146, 2011.
41. Z. Ma, D. L. Silver, and E. Shakshuki, "User profile management: reference model and web services implementation," *International Journal of Web and Grid Services*, vol. 6, no. 1, pp. 1–34, 2010.
42. T. Kwok, K. A. Smith, S. Lozano, and D. Taniar, "Parallel fuzzy c-means clustering for large data sets," in *Proceedings of the 8th International Euro-Par Conference on Euro-Par, Lecture Notes in Computer Science 2400 Springer*, pp. 365–374, 2002.