Taxonomy of spam email Classification by association rules

Mohammed A.Naser **Aathar H.Mohammed**

Department of Computer, College of Sciences for Women, University of Babylon

Abstract

Email has become one of the fastest and most economical forms of communication. However, the increase of email users have resulted in the dramatic increase of spam emails during the past few years. Association rule mining has a wide range of applicability such market basket analysis,and medical diagnosis/ research and so on. This paper explains and study a taxonomy for the spam email classification by association rules in details.

Keywords

Email, spam, taxonomy

الخلاصة

البريد الإلكتروني أَصْبَحَ واحدا من أسرعَ وأكثر أشكال الاتصال الاقتصادية والتجارية. على أية حال،فان زيادة مستعملي البريد الإلكتروني أدّتْ إلى الزيادةِ المضطردة لرسائل الدعاية إلكترونية والغير مرغوب خصوصا خلال السَنَوات الأخيرة. الدوال المترلبطة تستعمل بشكل واسع في العديد من التطبيقات مثل تحليل سلة التسوق والبحث والتشخيص الطبي ..الخ . هذا البحث يستعرض ويدرس بشكل مفصل أنواع التصنيف المختلفة لتلك الرسائل المشار إليها والتي تستخدم تقنية الدوال المترابطة في عملها.

1. Introduction

Electronic mail has become one of the most ubiquitous methods of communication. It is a fast, efficient and an inexpensive method of reaching out to a large number of people at the same time..A large part of the Internet mail traffic comprises of unsolicited bulk email (UBE). or spam as it is popularly,Other than the canned meat distributed by Hormel, spam is a term used to describe Unsolicited Commercial Email (UCE) Some say the keyword is Unsolicited, but we're sure many people have received emails without soliciting them that were not commercial and they sent to multiple accounts. For instance, there's the occasional joke sent in mass from friend to friends and back again, or that all-important virus alert, or the occasional inspiration, etc. But you know these people so it can't be spam, Well, it depends on to whom you speak. There are those who dislike having their mailbox fill up with jokes almost as much as commercial advertisements but because its done by someone you know it's tolerated. So, our definition of spam is any unsolicited email sent in bulk by an unknown entity that is uninvited.

As a result of this, efficient automated methods for analyzing the content of e-mail messages and identifying threats from these messages are becoming imperative. The time spent in this task can be greatly reduced if traditional classification techniques can be adapted to email classification thereby automating the process with sufficient accuracy and efficiency [Steve Davis et al.] . The main contribution of this study to present an algorithms that help in classifying E-mails by association rules.

Association rule mining is one of the most important and well researched techniques of data mining, it was first introduced in [Agrawal,R. et al., 1993] . It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. Association rules are widely used in various areas such as telecommunication networks, market and risk management, inventory control etc. Various association mining techniques and algorithms will be briefly introduced and compared later In this paper, we surveyed the most recent existing association rule mining techniques. The organization of the rest of the paper is as follows: Section2 contain the wrong of spam,section3 the basic concept of association rules ,section 4 contain methods used for generate rules that use later in classification of spam mails and section 5contain conclusion.

2-What's Wrong With Spam?

There are many problems associated with spam such as:

Time Costs

If you are receiving two or three Unsolicited Emails a day you probably think spam isn't all that bad, it's just a minor inconvenience. But if you are receiving 40 to 50 or more than at a day, and you're spending an average of 10 seconds each to decide what you want to do with each message, then you're wasting around 60 hours a year dealing with spam. That's over seven workdays wasted each year! Not to mention the raw frustration and distraction of doing a task that takes you from your productive work.

Server Costs

Then there are the costs to your server of having to manage large amounts of mail entering their system. When too much is sent or arrives at one time it can cause the system to crash, leaving their customers without the ability to send or receive email. One Internet Service Provider that's known for allowing spammers to send bulk mail through its system crashed when several of its users sent large amounts of mail at the same time. It was down for several days and many anti spammers thought that justice had its own way of dealing with spammers and hoped the Provider would start enforcing it's own Terms of Use. No such luck, its back and spammers are sending their junk mail in mass amounts once again.

Consumer Costs

Some consumers have to pay long distance phone charges to connect to the Internet (mostly in countries outside of the US) and some countries charge for every phone call made by their customers. In these cases, the user wastes connection time by downloading and sorting through unwanted email.

Privacy Costs

It's our belief that the biggest problem with spam, other than having to look at it, is that 90% of those sending it do it in a fraudulent way. They buy software that hides their identity, forges email headers, steals others' identities (read about one man's experience with identity theft at Behind Enemy Lines), use bogus cancellation addresses, and stake out a claim to their right to intrude on your privacy[Steve Davis et al.] .

3-Basic Concepts and Basic Association Rules Algorithms.

.

Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two subproblems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. Suppose one of the large itemsets is Lk, Lk = {I1, I2, … , Ik}, association rules with this itemsets are generated in the following way: the first rule is {I1, I2, … , Ik-1}$\Rightarrow${Ik}, by checking the confidence this rule can be determined as interesting or not. Then other rule are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them. Those processes iterated until the antecedent becomes empty. Since the second subproblem is quite straight forward,most of the researches focus on the first subproblem. The first sub-problem can be further divided into two sub-problems: candidate large itemsets generation process and frequent itemsets generation process. We call those itemsets whose support exceed the support threshold as large or frequent item- sets, those itemsets that are expected or have the hope to be large or frequent are called candidate itemsets. In many cases, the algorithms generate an extremely large number of association rules, often in thousands or even millions. Further, the association rules are sometimes very large. It is nearly impossible for the end users to comprehend or validate such large number of complex association rules, thereby limiting the usefulness of the data mining results. Several strategies have been proposed to reduce the number of association rules, such as generating only "interesting" rules, generating only "nonredundant" rules, or generating only those rules satisfying certain other criteria such as coverage, leverage, lift or strength.

Let I=I1, I2, … , Im be a set of m distinct attributes, T be transaction that contains a set of items such that $T \subseteq I$, D be a database with different transaction records Ts. An association rule is an implication in the form of $X \Rightarrow Y$, where $X, Y \subset I$ are sets of items called itemsets, and $X \cap Y = \square$. X is called antecedent while Y is called consequent,the rule means X implies Y. There are two important basic measures for association rules, support(s) and confidence(c). Since the database is large and users concern about only those frequently purchased items, usually thresholds of support and confidence are predefined by users to drop those rules that are not so interesting or useful. The two thresholds are called minimal support and minimal confidence respectively. Support(s) of an association rule is defined as the percentage/fraction of records that contain $X \cup Y$ to the total number of records in the database. Suppose the support of an item is 0.1%, it means only 0.1 percent of the transaction contain purchasing of this item.
Confidence of an association rule is defined as the percentage/fraction of the number of transactions that contain $X \cup Y$ to the total number of records that contain X. Confidence is a measure of strength of the association rules, suppose the confidence of the association rule $X \Rightarrow Y$ is 80%, it means that 80% of the transactions that contain X also contain Y together.

In general, a set of items (such as the antecedent or the consequent of a rule) is called an itemset. The number of items in an itemset is called the length of an itemset. Itemsets of some length k are referred to as k-itemsets. Generally, an association rules mining algorithm contains the following steps:
• The set of candidate k-itemsets is generated by 1-extensions of the large (k -1)- itemsets generated in the previous iteration.
• Supports for the candidate k-itemsets are generated by a pass over the database.
• Itemsets that do not have the minimum support are discarded and the remaining itemsets are called large k-itemsets. This process is repeated until no more large itemsets are found.

4-spam emails classification by association rules

Generally, the main tool for email management is text classification. A classifier is a system that classifies texts into the discrete sets of predefined categories. For the email classification, incoming messages will be classified as spam or legitimate using the rules these result from association rules methods.

4-1 The AIS algorithm.

(Agrawal, Imielinski, Swami) was the first algorithm proposed for mining association rule [Agrawal,R. et al., 1993] . In this algorithm only one item consequent association rules are generated, which means that the consequent of those rules only contain one item, for example we only generate rules like $X \cap Y \Rightarrow Z$ but not those rules as $X \Rightarrow Y \cap Z$.

Figure(1) AIS algorithm

```
1)  $L_1$= { large 1-itemsets};
2)  **For** (k=2; $L_{k-1} \neq ø$; k++) **do begin**
3)      $C_k$=ø;
4)      **Forall** transaction $t \in D$ **do begin**
5)          $L_t$=subset($L_{k-1}$,t); //Large itemsets contained in t
6)          **Forall** large itemsets $l_t \in L_t$ **do begin**
7)              $C_t$= 1-extensions of $l_t$ contained in t;  //Candidates contain in t
8)              **Forall** candidates $c \in C_t$ **do**
9)                  **If** ( $c \in C_k$ ) **then**
                        Add 1 to the count of c in the corresponding entry in $C_k$
                    **Else**
                        Add c to $C_k$ with a count of 1;
10)         **End**
11)     $L_k = \{c \in C_k \mid c.count \geq minsup\}$
12) **End**
13) Answer = $\cup_k L_k$;
```

The main drawback of the AIS algorithm is too many candidate itemsets that finally turned out to be small are generated, which requires more space and wastes much effort that turned out to be useless. At the same time this algorithm requires too many passes over the whole database.

4-2 SETM

The SETM algorithm was proposed in [M. Houtsma et al.,1995] and was motivated by the desire to use SQL to calculate large itemsets [Ramakrishnan Srikant et al.,1996] . In this algorithm each member of the set large itemsets, L_k , is in the form <TID, itemset> where TID is the unique identifier of a transaction. Similarly, each member of the set of candidate itemsets, C_k , is in the form <TID,itemset>. Similar to the AIS algorithm, the SETM algorithm makes multiple passes over the database. In the first pass, it counts the support of individual items and determines which of them are large or frequent in the database. Then, it generates the candidate itemsets by extending large itemsets of the previous pass. In addition, the SETM remembers the TIDs of the generating transactions with the candidate itemsets. The relational merge-join operation can be used to generate candidate itemsets [Ramakrishnan Srikant et al.,1996] . Generating candidate sets, the SETM algorithm saves a copy of the candidate itemsets together with TID of the generating transaction in a sequential manner. Afterwards, the candidate itemsets are sorted on itemsets, and small itemsets are deleted by using an aggregation function. If the database is in sorted order on the basis of TID, large itemsets contained in a transaction in the next pass are obtained by sorting L_k on TID.This way, several passes are made on the database. When no more large itemsets are found, the algorithm terminates.

The main disadvantage of this algorithm is due to the number of candidate sets C_k . Since for each candidate itemset there is a TID associated with it, it requires more space to store a large number of TIDs. Furthermore, when the support of a candidate itemset is counted at the end of the pass, C_k is not in ordered fashion. Therefore, again sorting is needed on itemsets. Then, the candidate itemsets are pruned by discarding the candidate itemsets which do not satisfy the support constraint. Another sort on TID is necessary for the resulting set (L_k). Afterwards, L_k can be used for generating candidate itemsets in the next pass. No buffer management technique was considered in the SETM algorithm [Agrawal,R. et al., 1994] . It is assumed that C_k can fit in the main memory. Furthermore, [Sarawagi Sunita et al. ,1998] mentioned that SETM is not efficient and there are no results reported on running it against a relational DBMS.

4-3 Apriori

Apriori is more efficient during the candidate generation process [Agrawal,R. et al., 1994] . Apriori uses pruning techniques to avoid measuring certain itemsets, while guaranteeing completeness. These are the itemsets that the algorithm can prove will not turn out to be large. However there are two bottlenecks of the Apriori algorithm.

1- is the complex candidate generation process that uses most of the time, space and memory.

2- is the multiple scan of the database. Based on Apriori algorithm,
many new algorithms were designed with some modifications or improvements.

There are many types of apriori such as:-

4-3.1 Apriori (basic Apriori)

Apriori is an algorithm to find all sets of items (itemsets) that have support no less than *minsup*. The support for an itemset is the ratio of the number of transactions that contain the itemset to the total number of transactions. Itemsets that satisfy minimum support constraint are called *frequent itemsets*. Apriori is characterized as a level-wise complete search (breadth first search) algorithm using anti-monotonicity property of itemsets: "If an itemset is not frequent, any of its superset is never frequent," which is also called the *downward closure property*. The algorithm makes multiple passes over the data. In the first pass, the support of individual items is counted and frequent items are determined. In each subsequent pass, a seed set of itemsets found to be frequent in the previous pass is used for generating new potentially frequent itemsets, called *candidate itemsets*, and their actual support is counted during the pass over the data.

At the end of the pass, those satisfying minimum support constraint are collected, that is, frequent itemsets are determined, and they become the seed for the next pass.This process is repeated until no new frequent itemsets are found.By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. The number of items in an itemset is called its *size* and an itemset of size *k* is called a *k*-itemset[Xindong Wu et al. ,2009].

```
F_1= { frequent 1- itemsets);
**For** (k=2; $F_{k-1} \neq ø$ ;k++) **do begin**
    $C_k$ = apriori-gen ($F_{k-1}$); // New candidates
    **Foreach** transaction t ϵ D **do begin**
        $C_t$ = subset( $C_k$ , t); // Candidates contained in t
        **Foreach** candidate c ϵ $C_t$ **do**
            c.count ++;
    **end**
    $F_k$ = { c ϵ $C_k$ | c.count $\geq$ minsup };
**End**
**Answer** = $U_k$ $F_k$ ;
```

Figure(2) Algorithm 1 Apriori Algorithm

the algorithm to generate association rules is given in figure(3) Algorithm 2.

```
$H_1$ = ø // Initialize
Foreach; frequent k-itemset $f_k$ , k ≥ 2 do begin
    A = (k-1)-itemsets $a_{k-1}$ such that $a_{k-1} \subset f_k$ ;
    Foreach $a_{k-1}$ ϵ A do begin
        *Conf* = support ($f_k$) / support ( $a_{k-1}$);
        If (*conf* ≥ *minconf*) then begin
            Output the rule $a_{k-1}$ => ($f - a_{k-1}$)
                With confidence = *conf* and support = support ($f_k$)
            Add ($f_k - a_{k-1}$ ) to $H_1$;
        End
        Call ap-genrules ($f_k$ , $H_1$);
    End
    Procedure ap-genrules ($f_k$ : frequent *k*-itemset, $H_m$ : set of m-item consequents)
        If ( k > m+1) then begin
            $H_{m+1}$ = apriori-gen( $H_m$ );
            Foreach $h_{m+1} \in H_{m+1}$ do begin
                *Conf* =support ($f_k$ ) / support ($f_k - h_{m+1}$ );
                If ( *conf* ≥ *miconf* ) then
                    Output the rule $f_k - h_{m+1}$ => $h_{m+1}$
                        With confidence = *conf* and support = support ($f_k$);
                Else
                    Delete $h_{m+1}$ from $H_{m+1}$;
            End
            Call ap-genrules ($f_k$ , $H_{m+1}$);
        End
```

Figure (3) Algorithm 2 Association rule generation Algorithm

4-3.2 AprioriTid

AprioriTid is a variation of Apriori. It does not reduce the number of candidates but it does not use the database *D* for counting support after the first pass. It uses a new dataset *Ck* . Each member of the set *Ck* is of the form < *TID,* {*I D*} >, where each *I D* is the identifier of a potentially frequent *k*-itemset present in the transaction with identifier *TID* except *k* = 1. For *k* = 1, *C*1 corresponds to the database *D*, although conceptually each item *i* is replaced by the itemset {*i* }. The member of *Ck* corresponding to a transaction *t* is < *t.TID,* {*c* ∈ *Ck* |*c* contained in *t*} >.The intuition for using *Ck* is that it will be smaller than the database *D* for large values of *k* because some transactions may not contain any candidate *k*-itemset, in which case *Ck* does not have an entry for this transaction, or because very few candidates may be contained in the transaction and each entry may be smaller than the number of items in the corresponding transaction. AprioriTid algorithm is given in Algorithm 3. Here, *c*[*i*] represents the *i* -th item in *k*-itemset *c*.Each *Ck* is stored in a sequential structure.Acandidate *k*-itemset *ck* in *Ck* maintains two additional fields; generator and extensions, in addition to the field, support count. The generator field stores the *ID*s of the two frequent (*k* − 1)-itemsets whose join generated *ck* . The extension field stores the *ID*s of all the (*k* + 1)-candidates that are extensions of *ck* . It is thus expected that Apriori beats AprioriTid in earlier passes (small *k*) and AprioriTid beats Apriori in later passes (large *k*) [Xindong Wu et al. ,2009].

```
$F_1$ = { frequent 1-itemsets};
$\bar{G}_1$ = database D;
for ( k = 2; $F_{k-1} \neq ø$; k++ ) do begin
  $C_k$ = apriori-gen($F_{k-1}$ ); // New candidates
  $\bar{G}_k = ø$;
  foreach entry t $\epsilon$ $\bar{G}_{k-1}$ do begin
     // determine candidate itemsets in $C_k$ contained
     // in the transaction with identifier t.TID
     $C_t$ = {c $\epsilon$ $C_k$ | (c - c[k]) $\epsilon$ t.set-of-itemsets ^
           (c- c[k-1]) $\epsilon$ t.set-of-itemsets};
    foreach candidates c $\epsilon$ $C_t$ do
        c.count++;
    if ($C_t \neq ø$) then $\bar{G}_k$ += < t.TID,$C_t$ >;
  end
  $F_k$ = {c $\epsilon$ $C_k$ | c.count $\geq$ minsup};
 end
 Answer = $U_k$ $F_k$;
```

Figure(4) Algorithm 3 Apriori tid algorithm

4-3.3 Apriori-Hybrid

This algorithm is based on the idea that it is not necessary to use the same algorithm in all passes over data. As mentioned in [Agrawal,R. et al., 1994] , Apriori has better performance in earlier passes, and Apiori-TID outperforms Apriori in later passes. Based on the experimental observations, the Apriori-Hybrid technique was developed which uses Apriori in the initial passes and switches to Apriori-TID when it expects that the set C_k at the end of the pass will fit in memory. Therefore, an estimation of C_k at the end of each pass is necessary. Also, there is a cost involvement of switching from Apriori to Apriori-TID. The performance of this technique was also evaluated by conducting experiments for large datasets. It was observed that Apriori- Hybrid performs better than Apriori except in the case when the switching occurs at the very end of the passes [Ramakrishnan Srikant et al.,1996] .

Since both Apriori and AprioriTid use the same candidate generation procedure and therefore count the same itemsets, it is possible to make a combined use of these two algorithms in sequence. AprioriHybrid uses Apriori in the initial passes and switches to AprioriTid when it expects that the set *Ck* at the end of the pass will fit in memory

4-3.4 AprioriAll

The algorithm is given in Algorithm 4. In each pass the frequent sequences from the previous pass are used to generate the candidate sequences and then their support is measured by making a pass over the database. At the end of the pass, the support of the candidates is used to determine the frequent sequences.

The apriori-gen-2 function takes as argument *Fk*−1, the set of all frequent (*k* − 1)-sequences. First, join operation is performed as

insert into *Ck*
select *p*.fitemset1, *p*.fitemset2, . . . , *p*.fitemset*k*−1, *q*.fitemset*k*−1
from *Fk*−1 *p, Fk*−1*q*
where *p*.fitemset1 = *q*.fitemset1, . . . , *p*.fitemset*k*−2 = *q*.fitemset*k*−2,

```
F_1 = {frequent 1-sequences} ; // Result of fitemset phase
for ( k = 2; $F_{k-1} \neq \emptyset$ ; k++ ) do begin
   $C_k$ = apriori-gen-2 ($F_{k-1}$ ); // New candidate sequences
   foreach transaction sequences  $t \in D_T$  do begin
      $C_t$ = subseq ($C_k$ , $t$); // Candidates sequences contained in t
      foreach candidate $c \in C_t$ do
         c:count++;
   end
   $F_k = \{ c \in C_k \mid c{:}count \geq minsup \}$
end
Answer = maximal sequences in $U_k F_k$;
```

Figure(5) Algorithm 4 Apriori all algorithm

then, all the sequences $c \in Ck$ for which some $(k-1)$-subsequence is not in $Fk-1$ are deleted. The subseq function is similar to the subset function in Apriori. As in Apriori, the candidate sequences *Ck* are stored in a hash-tree to quickly find all candidates contained in a transaction sequence. Note that the transformed transaction sequence is a list of sets of fitemsets and all the fitemsets in a set have the same transaction-time, and no more than one transaction with the same transaction-time is allowed for the same sequence-id. This constraint has to be imposed in the subseq function [Xindong Wu et al. ,2009].

4-4 Dynamic Itemset Counting

DIC (Dynamic Itemset Counting) tries to generate and count the itemsets earlier,thus reducing the number of database scans. The database is viewed as intervals of transactions,and the intervals are scanned sequentially. While scanning the first interval, the 1-itemsets are generated and counted. At the end of the first interval, the 2-itemsets which are potentially large are generated. While scanning the second interval, all 1-itemsets and 2-itemsets generated are counted. At the end of the second interval, the 3-itemsets that are potentially large are generated, and are counted during scanning the third interval together with the 1-itemsets and 2-itemsets. In general, at the end of the *k*th interval, the (k+1)-itemsets which are potentially large are generated and counted together with the previous itemsets in the later intervals. When reaching the end of the database, it rewinds the database to the beginning and counts the itemsets which are not fully counted. The actual number of database scans depends on the interval size. If the interval is small enough, all itemsets will be generated in the first scan and fully counted in the second scan. It also favors a homogeneous distribution as does the PARTITION. [Sergey Brin, et al.,1997]

4-5 CARMA

CARMA (Continuous Association Rule Mining Algorithm) brings the computation of large itemsets online. Being online, CARMA shows the current association rules to the user and allows the user to change the parameters, minimum support and minimum confidence, at any transaction during the first scan of the database. It needs at most 2 database scans. Similar to DIC, CARMA generates the itemsets in the first scan and finishes counting all the itemsets in the second scan. Different from DIC, CARMA generates the itemsets on the fly from the transactions. After reading each transaction, it first increments the counts of the itemsets which are subsets of the transaction. Then it generates new itemsets from the transaction, if all immediate subsets of the itemsets are currently potentially large with respect to the current minimum support and the part of the database that is read. For more accurate prediction of whether an itemset is potentially large, it calculates an upper bound for the count of the itemset, which is the sum of its current count and an estimate of the number of occurrences before the itemset is generated. The estimate of the number of occurrences (called maximum misses) is computed when the itemset is first generated[Christian Hidber et al., 1999].

4-6 FP-Tree

Frequent Pattern mining (FP-Tree), is another milestone in the development of association rule mining, which breaks the main bottlenecks of the Apriori. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. FP-tree is an extended prefix-tree structure storing crucial, quantitative information about frequent patterns. Only frequent length-1 items will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones.

```
$F$ [I] =ø;
Foreach $i \in I$ that is in $D$ in frequency increasing order do begin
    $F$ [I] =$F$ [I] U {$I$ U {i}};
    $D^i$ = ø;
    $H$ = ø;
    Foreach $j \in I$ in $D$ such that $j < i$ do begin
        // ( $j$ is more frequent than $i$ )
        Select $j$ for which support ( $I$ U { $i$ ,$j$ }) $\geq$ $minsup$;
        $H = H$ U { $j$};
    End
    Foreach ($Tid$ , $X$) $\in D$ with $i \in X$ do
        $D^i = D^i$ U {($Tid$, { $X \setminus$ {$i$}} $\cap$ $H$)};
    Construct conditional FP-tree from $D^i$;
    Call $F$[$I$ U {$i$}](conditional FP-tree);
    $F$[$I$] = $F$[$I$] U $F$ [ $I$ U {$i$}](conditional FP-tree);
End
```

Figure (6) Algorithm 5 Fp_growth algorithm

FP-Tree scales much better than Apriori because as the support threshold goes down, the number as well as the length of frequent itemsets increase dramatically. The candidate sets that Apriori must handle become extremely large, and the pattern matching with a lot of candidates by searching through the transactions becomes very expensive. The frequent patterns generation process includes two sub processes: constructing the FT-Tree, and generating frequent patterns from the FP-Tree. The mining result is the same with Apriori series algorithms.

The efficiency of FP-Tree algorithm account for three reasons.

1- the FP-Tree is a compressed representation of the original database because only those frequent items are used to construct the tree, other irrelevant information are pruned.

2- this algorithm only scans the database twice.

3- FP-Tree uses a divide and conquer method that considerably reduced the size of the subsequent conditional FP-Tree.In [Han, J. , et al., 2000] there are examples to illustrate all the detail of this mining process. Every algorithm has his limitations, for FP-Tree it is difficult to be used in an interactive mining system. During the interactive mining process, users may change he threshold of support according to the rules. However for FP-Tree the changing of support may lead to repetition of the whole mining process. Another limitation is that FP-Tree is that it is not suitable for incremental mining. Since as time goes on databases keep changing, new datasets may be inserted into the database, those insertions may also lead to a repetition of the whole process if we employ FP-Tree algorithm.

4-7 TreeProjection

is another efficient algorithm recently proposed in [Agrawal,R. et al., 2000. The general idea of TreeProjection is that it constructs a lexicographical tree and projects a large database into a set of reduced, item-based sub-databases based on the frequent patterns mined so far. The number of nodes in its lexicographic tree is exactly that of the frequent itemsets.

The efficiency of TreeProjection can be explained by two main factors:

(1) the transaction projection limits the support counting in a relatively small space;

(2) the lexicographical tree facilitates the management and counting of candidates and provides the flexibility of picking efficient strategy during the tree generation and transaction projection phrases.

4-8 PRICES

Wang and Tjortjis [Wang, C., et al. ,2004] presented an efficient algorithm for mining association rules. Advantages of this approach reduces large itemset generation time, known to be the most time-consuming step, by scanning the database only once and using logical operations in the process.

4-9 Matrix Algorithm.

This algorithm for efficient generating large frequent candidate sets is proposed by [Yuan, Y., et al. 2005], generates a matrix which entries 1 or 0 by passing over the cruel database only once, and then the frequent candidate sets are obtained from the resulting matrix. Finally association rules are mined from the frequent candidate sets. Experiments results confirm that the proposed algorithm is more effective than Apriori Algorithm.

4-10 Sampling Error Estimation (SEE)

[Chuang ,K., et al., 2005] explore another progressive sampling algorithm which aims to identify an appropriate sample size for mining association rules.

Algorithm *SEE* will generate a curve of *sampling errors* versus the sample size immediately after one database scan, and then suggest the corresponding sample size at the convergence point of this curve as the appropriate sample size for association rules. To measure *sampling errors*, frequencies (or said *support count*) of each item in the entire database and in each sample will be requiredin *SEE*.

To efficiently acquire such information, *SEE* is devised as a two phases algorithm:

(1) the *database scan phase*, in which the database is scanned once
and simultaneously the *frequency* of each item in the entire database and in
each sample is stored.

*(3)*The *convergence detection phase*, in which *sampling errors* of each sample size are
*(4)*calculated, and then the appropriate sample size is identified from the curve of *sampling errors* versus the sample size. The pseudo code of SEE is outlined below:

```
Algorithm SEE: SEE(D,L, $\mathbb{R}$,minSup)
  SEE[$n$][$m$] : store sampling errors of the $m^{th}$ sample of size $s_n$ .
//The database scan phase
    01.  While has next transaction $t_d$
    02.      For every item $i_k$ in  $t_d$
    03.          $IList[i_k] \rightarrow freq_D$ ++;
    04.      For $n$ = 1 to P
    05.          For $m$ = 1 to $L$
    06.              If ( $rv[n][m]$.next < $\frac{8n}{|D|}$ )
    07.                  For every item $i_k$ in $t_d$
    08.                      $I\,List\,[i_k] \rightarrow freq_S[n][m]$ ++;

// The convergence detection phase
    01.  For $n$ =1 to $P$
    02.      For $m$ = 1 to $L$
    03.          e = 0; count_item =0;
    04.          For each item $i_k$ in $I\,List$ {
    05.              If $i_k$ belongs to case (1) / (2) / (3)
    06.                  e+= $\left( \frac{IList[ik] \rightarrow freqS[n][m]}{8n} - \frac{IList[ik] \rightarrow freqD}{|D|} \right)^2$
    07.                  Count_item ++;
    08.          $SEE[n][m] = \sqrt{\frac{e}{count\_item}}$ ;
    09.      $A\_SEE(s_n) = \frac{\sum_{j=1}^{L} SEE[n][j]}{L}$ ;
    10.      If ( $s_n$, $A\_SEE(s_n)$) is the convergence point
    11.          Report $s_n$ as the appropriate sample size; program terminated;
```

Figure (7) Algorithm 6 SEE Algorithm

4-11 FDM.

[Cheung, D. ,et al., 1996] presented an algorithm FDM is a parallelization of Apriori to (shared nothing machines, each with its own partition of the database. At every level and on each machine, the database scan is performed independently on the local partition. Then a distributed pruning technique is employed.

4-12 Schuster and Wolff algorithm

Schuster and Wolff [Schuster, A et al., 2001] described another Apriori based D-ARM algorithm - DDM. As in FDM, candidates in DDM are generated level wise and are then counted by each node in its local database. The nodes then perform a distributed decision protocol in order to find out which of the candidates are frequent and which are not.

4-13 FPM (Fast Parallel Mining)

Another efficient parallel algorithm for mining association rules on a shared-nothing parallel system has been proposed by [Cheung ,D., et al., 1998] . It adopts the count distribution approach and has incorporated two powerful candidate pruning techniques, i.e., distributed pruning and global pruning. It has a simple communication scheme which performs only one round of message exchange in each iteration.

4-14 Data Allocation Algorithm (DAA)

New algorithm, is presented in [Manning ,A. et al. that uses Principal Component Analysis to improve the data distribution prior to FPM.

[Parthasarathy, S., et al.,2001] have written an excellent recent survey on parallel association rule mining with shared memory architecture covering most trends, challenges and approaches adopted for parallel data mining. All approaches spelled out and compared in this extensive survey are apriori-based.

```
1. Initialize $C_1 = \{\{i\} : i \in I\}$, $k = 1$, $Passed = ø$
2. While $C_k \neq ø$
    a) Do
        • Choose an itemset $X_i \in C_k$ which was
          Not yet chosen and for which either
          $H(X_i) < MinSup \leq P(X_i, DB^j)$ or
          $P(X_i, DB^j) < MinSup \leq H(X_i)$ and
          Broadcast $< i, Support(X_i, DB^j) >$.
        • If no such itemset exists broadcast $< pass >$.
    b) Until $|Passed| = n$
    c) $L_k = \{X_i \in C_k : H(X_i) \geq MinSup\}$
    d) Broadcast the support counts for every
       $X_i \in L_k$ which was never chosen.
    e) $C_{k+1} = Apriori\_Gen(L_k)$
    f) $k = k + 1$
3. $Gen\_Rules(L_1, L_2, \ldots, L_k)$

For node j out of n
4. Initialize $C_1 = \{\{i\} : i \in I\}$, $k = 1$, $Passed = ø$
5. While $C_k \neq ø$
    g) Do
        • Choose an itemset $X_i \in C_k$ which was
          Not yet chosen and for which either
          $H(X_i) < MinSup \leq P(X_i, DB^j)$ or
          $P(X_i, DB^j) < MinSup \leq H(X_i)$ and
          Broadcast $< i, Support(X_i, DB^j) >$.
        • If no such itemset exists broadcast $< pass >$.
    h) Until $|Passed| = n$
    i) $L_k = \{X_i \in C_k : H(X_i) \geq MinSup\}$
    j) Broadcast the support counts for every
       $X_i \in L_k$ which was never chosen.
    k) $C_{k+1} = Apriori\_Gen(L_k)$
    l) $k = k + 1$
6. $Gen\_Rules(L_1, L_2, \ldots, L_k)$

When node j receives a message M from node p:
1. If $M = <pass>$ insert p into Passed.
2. Else if $|Passed| = n$ the $M$ is the support counts
   of itemsets $p$ has not yet sent. Update accordingly.
3. Else $M = < i, Support < X_i, DB^p >>$
   • If $p \in Passed$ the remove $p$ from $Passed$.
   • Recalculate $H(X_i)$ and $P(X_i, DB^p)$
```

Figure(8) Algorithm 7 DDM algorithm

5-Conclusions

In this paper we present a taxonomy for spam emails classification algorithms by association rules with a survey for algorithms. Also, this paper include the basic principles of spam,and the problem of it, . In addition, In this paper, we have presented a wide range of the techniques that have been used or proposed for use to classify SPAM emails. The taxonomy presented here is clearly preliminary in nature, and non-exhaustive. A clear task for the future is to expand it with information about additional SPAM filters and techniques, and to address any refinements that become apparent during that process.

References

Agrawal,R. et al., 1993 Agrawal, R., Imielinski, T., and Swami, A. N. 1993. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 207-216.

Agrawal,R. et al., 1994 Agrawal, R. and Srikant, R., 1994. Fast algorithms for mining association rules. In Proc. 20th Int. Conf. Very Large Data Bases, 487-499.

Agrawal,R. et al., 2000 Agarwal, R. Aggarwal, C. and Prasad V. 2000. A tree projection algorithm for generation of frequent itemsets. In J. Parallel and Distributed Computing.

Cheung, D. ,et al., 1996 Cheung, D., Han, J., Ng, V., Fu, A. and Fu, Y. 1996, A fast distributed algorithm for mining association rules, in `Proc. of 1996 Int'l. Conf. on Parallel and Distributed Information Systems', Miami Beach, Florida, pp. 31 - 44.

Cheung ,D., et al., 1998 Cheung, D., Xiao, Y. Aug 1998. Effect of data skewness in parallel mining of association rules, Lecture Notes in Computer Science, Volume 1394, Pages 48 – 60

Christian Hidber et al., 1999 Christian Hidber, June 1-3. 1999. Online Association Rule Mining, SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data,Philadephia, Pennsylvania, pp.145-156.

Chuang ,K., et al., 2005 Chuang, K., Chen, M., Yang, W., Jun 2005. Progressive Sampling for Association Rules Based on Sampling Error Estimation, Lecture Notes in Computer Science, Volume 3518, Pages 505 – 515

Han, J. , et al., 2000 Han, J. and Pei, J. 2000. Mining frequent patterns by pattern-growth: methodology and implications. ACM SIGKDD Explorations Newsletter 2, 2, 14-20.

Manning ,A. et al. Manning, A., Keane, J., Data Allocation Algorithm for Parallel Association Rule Discovery, Lecture Notes in Computer Science, Volume 2035, Page 413-420.

M. Houtsma et al.,1995M. Houtsma and A. Swami, .March 1995. Set-Oriented Mining for Association Rules in Relational Databases, *Proceedings of the 11th IEEE International Conference on Data Engineering*, pp. 25-34, Taipei, Taiwan.

Parthasarathy, S., et al.,2001 Parthasarathy, S., Zaki, M. J., Ogihara, M., February 2001. Parallel data mining for association rules on shared-memory systems. Knowledge and Information Systems: An International Journal, 3(1):1–29

Rakesh Agrawal et al.,1994 Rakesh Agrawal and Ramakrishnan Srikant, 1994. Fast Algorithms for Mining Association Rules in Large Databases, *Proceedings of the Twentieth International Conference on Very Large Databases*, pp. 487-499, Santiago, Chile.

Ramakrishnan Srikant et al.,1996 Ramakrishnan Srikant, *1996.* Fast Algorithms for Mining Association Rules and Sequential Patterns, *Ph.D Dissertation, University of Wisconsin*, Madison.

Sarawagi Sunita et al. ,1998 Sarawagi Sunita, Thomas Shiby, and Agrawal Rakesh, June 2-4, 1998. Integrating Mining with Relational Database Systems: Alternatives and Implications, Proceedings ACM SIGMOD International Conference on Management of Data, SIGMOD 1998 , Seattle, Washington, USA

Schuster, A et al., 2001 Schuster, A. and Wolff, R. (2001), Communication-efficient distributed mining of association rules, in `Proc. of the 2001 ACM SIGMOD Int'l. Conference on Management of Data', Santa Barbara, California, pp. 473-484.

Sergey Brin, et al.,1997 Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur.1997.Dynamic Itemset Counting and Implication Rules for Market Basket Data, *Proceedings of the ACM SIGMOD Conference*, pp. 255-264.

Steve Davis et al. Steve Davis & Gloria Craney., “How Do I Stop Spam?”

Yuan, Y., et al. 2005 Yuan, Y., Huang, T., Sep 2005. A Matrix Algorithm for Mining Association Rules, Lecture Notes in Computer Science, Volume 3644, Pages 370 – 379

Wang, C., et al. ,2004 Wang, C., Tjortjis, C., Jan 2004. PRICES: An Efficient Algorithm for Mining Association Rules, Lecture Notes in Computer Science, Volume 3177, Pages 352 – 358

Xindong Wu et al. ,2009 Xindong Wu, vipin kumar,2009.Top ten algorithms in data mining.