Knowledge Discovery of Closed Frequent Calling Patterns in a Telecommunication Database

Saidat Adebukola Ibrahim, Olusegun Folorunso, and Olutayo Bamisele Ajayi
University of Agriculture, Abeokuta, Nigeria

bookyy2k@yahoo.com folorunsolusegun@yahoo.com
tayoajayi@yahoo.co.uk

Abstract

A telecommunication network produces daily large amounts of calling data which contain hidden and valuable knowledge. This knowledge can be used in determining the calling patterns of customer, finding too thrifty customers, also for locating the best area to concentrate on in order to boost profits. In this paper, we designed an algorithm named CLOTELE which is based on the ideas of the pattern-growth method of mining, for mining closed frequent calling patterns of a telecommunication database from a telecommunication provider. First, by observing the features of the database and then extracting the attributes needed to be mined. Then, we merge the items to form an itemset, the algorithm is now applied to the transformed database. CLOTELE is built on the basis of viewing knowledge discovery as an interactive and iterative process in order to optimize decision making. The quality of the knowledge discovered is evaluated. The experimental results show the knowledge of closed frequent patterns obtained is very useful and easy to interpret by the network operators and administrators.

Keywords*:* CLOTELE, closed frequent pattern, telecommunication database, knowledge.

Introduction

Databases are a dormant potential resource that, tapped, can produce the substantial benefits. Data mining, which is also referred to as knowledge discovery in databases, is the non-trivial extraction of implicit, valid, previously unknown, potentially useful and ultimately understandable patterns from huge amounts of data for use in making crucial decisions (Thearling, 1996).

Given a set of facts (data) F, a language, L and some measure of certainty C, we define a pattern as a statement S in L that describes relationships among a subset F_s of F with a certainty c such that S is simpler in some sense than the enumeration of all the facts in F_s.

A pattern that is interesting (according to a user-imposed interest measure) and certain enough (again according to the user's criteria) is called knowledge. The output of a program that monitors the set of facts in a database and produces patterns in this sense is discovered knowledge. However, in the whole problem of obtaining useful knowledge, the inference of patterns is only a small part. Among the other tasks are the following:

1. data collection and cleaning (what types of data can be used, how errors in the data are handled, what is to be done with missing data, etc.); identification of the necessary background knowledge;

2. choice of pattern discovery methods (what types of knowledge are to be discovered, parameter selection, etc.);

3. discovery of patterns

4. presentation of patterns

5. putting the discovered knowledge into use.

A telecommunication system however, produces daily a large amount of data which contains hidden valuable information about the calling patterns of consumers. A telecommunication network can be viewed as consisting of a number of interconnected components: switches, exchanges, transmission equipment, etc. Each component in its turn contains several sub-components. The number of components depends on the abstraction level used in viewing the system. A network operated by a local telephone company can be considered to contain 10 - 1000 components.

In classical frequent pattern mining algorithms, huge number of frequent patterns are generated, whereas among them exist redundant information. To reduce the redundancies found, one can specify that the frequent patterns must be closed.

Closed frequent pattern mining, however, solve this problem by returning a succinct result with redundancies being removed. The scenario below gives a good understanding of Closed frequent patterns.

Suppose the frequent patterns generated are: *{bread, butter: 10}; {sugar, butter: 10}; {bread, sugar : 10}; {bread, sugar , butter : 10}*.

Closed frequent pattern mining will return one itemset only: *{bread, sugar, butter: 10}*. This itemset, however, represents the complete information about the frequency of its three sub-itemsets.

Problem Definition

Definition: Frequent itemset

Let $I = \{x_1 \ldots x_n\}$ be a set of items. An itemset X is a subset of items, i.e., $X \subseteq I$. For the sake of brevity, an itemset $X = \{x_1, x_2 \ldots, x_m\}$ is also denoted as $x_1, x_2 \ldots, x_m$. A transaction $T = (\text{tid}, X)$ is a 2-tuple, where tid is a transaction identifier (i.e. customer identifier) and X, an itemset. A transaction $T = (\text{tid}, X)$ is said to contain itemset Y if $Y \subseteq X$. A transaction database, TDB is a set of transactions in TDB containing itemset X, the support of an itemset X is the number of times it occurs in a transaction, denoted as supp(X). Given a transaction database TDB and a support threshold, min_sup, an itemset X is a frequent pattern, if supp(X) $\geq$ min_sup.

Definition: Closed frequent itemset

A frequent itemset, X is now said to be a closed itemset if there exists no X′ such that X′ is a proper superset of X and every transaction containing X also contains X′. A closed itemset is frequent if it passes the given support threshold.

Knowledge about closed frequent patterns is interesting and useful when the right algorithm is used. Another approach of mining closed frequent patterns which adopts the methodology of pattern growth methods and avoids the candidate generation and test approach is hereby proposed. A pattern growth uses the Apriori property, however, instead of generating candidates-sets, it recur-

sively partitions the database into sub-databases according to the frequent patterns found and searches for local frequent patterns to assemble local ones (Han, Pei, & Yin (2000).

In this paper, we design an algorithm named CLOTELE which is based on the idea of the pattern growth method, with completeness property. We use this algorithm to find effectively closed frequent patterns from a telecommunication database which are small subsets of frequent itemsets, but they represent exactly the same knowledge in a more succinct way. From the set of closed itemsets, it is in fact straight-forward to derive both the identities and supports of all frequent itemsets. Mining the closed itemsets is semantically equivalent to mine all frequent itemsets, but with the great advantage that closed itemsets are fewer than frequent ones (Lucchess, Orlando & Perego, 2004).

We are only interested in the database generated by a PABX switch of a telecommunication provider in Nigeria. Table 1 shows an example of selected real data where *RID* is the unique identifier for each call, *LineID* identifies the phone line that was randomly picked by the PABX to make the call. *CodeUsed* is the phone code used by the staff, it uniquely identifies each staff. *AreaCalled* is the area that is called. *Category* states whether the call is international, trunk or local. *CallersName* is the name of the person that makes the call. *TrunkGroup* is the trunk group that the phone line belongs to. *Direction* identifies the direction of the call whether incoming or outgoing call.

Table 1: An example of selected calling data

RID	*CallDate*	*LineID*	*Direction*	*CodeUsed*	*AreaCalled*	*CallersName*	*TrunkGroup*	...
1	01/01/04	71501	I	90311	MTN	Ifeanyi Odoemele	707	…
2	01/01/04	70703	O	25177	Lagos	Bisola Ogunsanya	716	…
3	01/01/04	70704	O	90311	Lagos	Bisola Ogunsanya	707	…
4	01/01/04	70703	O	90311	Maiduguri	Tosan Kukoyi	718	…
…	...	…	…	…	…	…	…	…

Related Works

A-Close (Pasquier , Bastide, Taouil, & Lakhal. (1999) was the first algorithm for closed itemset mining based on the Apriori heuristic. A-close is a variation of Apriori, it adopts the Apriori framework, but looks for frequent closed itemsets and prunes the frequent itemsets that are not closed. The major cost of the A-Close is from two aspects: (1) it has to generate a lot of candidates and scan the transaction database again and again to count candidates; and (2) in the last scan to compute closures, there could be a large number of surviving frequent itemsets. For each transaction, the intersection with each surviving frequent itemsets is done. This makes the closure computation quite costly.

The authors of FP-growth proposed CLOSET (Pei, Han, & Mao, 2000) for mining closed frequent patterns. This algorithm inherits from FP-growth, the compact FP-Tree data structure and the exploration technique based on recursive conditional projections of the FP-Tree. Frequent single items are detected after a first scan of the dataset, and with another scan, the pruned transactions are inserted in the FP-Tree stored in the main memory. Despite the efficiency of this FP-

growth, if the database is huge, the FP-tree will be large and the space requirement for recursion is a challenge (Pei et al., 2001).

CHARM for finding closed frequent itemsets proposed by Zaki and Hsiao performs a bottom-up depth first browsing of a prefix tree of frequent itemsets built incrementally. As soon as a frequent itemset is generated, its tid-list is compared with those of the other itemsets having the same parent. When two tid-lists are equal, or one includes the other, the associated nodes are merged since the itemsets surely belong to the same equivalence class. Itemset tid-lists are stored in each node by using the diff-set technique (Zaki & Hsiao, 2002).

Hatonen, Klemettinen, Mannila, Ronkainen and Toivonen (1996) designed a system for discovering and browsing knowledge from telecommunication network alarm databases called TASA (Telecommunication Network Alarm Sequence Analyzer). The system uses a framework for locating frequently occurring episodes from sequential data.

Daszczuk et al. (2000) experimented with a data mining project carried out by the Warsaw University of Technology in the network planning and Maintenance Department of a Polish cellular telecom provider on real data from the technology area. The goal of the project was to find the customer profile of subscribers that make calls shorter than five seconds. This was proposed to the marketing department.

Wu, Peng, and Chen (2001) proposed an algorithm for mining sequential alarm patterns from the alarm data of a GSM system. The knowledge extracted is useful for alarm prediction and alarm control.

Claudio et al. (2004) in their own paper proved a mathematical theory for checking out duplicates in the patterns generated. This idea is used in the designed algorithm to eliminate duplicates in the closed patterns generated.

Types of Knowledge to be Discovered

From a sequence of the calling data, different types of knowledge can be discovered. If the goal would be just to obtain good predictive performance, a neural network could be useful. There is impressive evidence of the wide applicability of neural networks. However, in the current application, one important goal is the comprehensibility of the discovered knowledge: the telecommunication provider wishes to know which area to concentrate on; how long is a particular staff spending on making calls, etc. This rules out the simple-minded use of the neural networks.

Procedure of Mining Closed Frequent Calling Patterns

A PABX switch is a resource used by telecommunication service providers to provide communication service. Big firms also use it in order to allow a smooth telecommunication link. With the rapid advancement of digital technology, telecommunication switches have evolved from hard-wired, mechanical devices to flexible, dynamic, software-configurable devices. In effect, they have become specialized computers. The procedure of mining closed frequent calling patterns involves the following tasks:

1. **Data Cleaning:** The goal of data cleaning is to identify the available data sources and extract the data that is needed for preliminary analysis in preparation for future mining. Clearly, the process of data cleaning depends on the patterns to be mined. In order to extract useful closed frequent calling patterns, data cleaning and data preprocessing are needed (Han & Kamber, 2001). The operations of data cleaning include selecting those attributes needed for data mining and removing those redundant information or outliers.

2. Discovery of Patterns: After the process of data cleaning, we design a data mining algorithm to discover the valuable closed frequent calling patterns from the telecommunication database.

Algorithm Design and Implementation

This section presents the complete description of the algorithm.

Algorithm: Mining closed frequent patterns integrating the advantages of the pattern growth methods.

Input: Transaction database, TDB which consists of a set of items.

Output: Closed frequent itemsets.

Method: The algorithm can be divided into several steps as below:

Step 1 - Find frequent items

In this step, the transaction database is scanned once. During this scan, the count for each item is collected. In the meanwhile, their counts are now compared with the minimum support threshold to generate the frequent items, since only the frequent items play a role in the frequent pattern mining. Those items that do not meet up with the minimum support are considered infrequent and hence, discarded.

Note that the support defined in this paper is the absolute occurrence frequency.

For each itemset of a transaction, let *Freq(X)* be the frequent-item projection of itemset X which includes only the frequent items found in each transaction. The order should be left as they are in the original database; the ordering is a little time waste, since the objective of the study is to reduce the processing time, though, same results will be generated if it is in any order. If multiple transactions share an identical frequent item set, they can be merged into one with the number of occurrences registered as count. It is easy to check whether two sets are identical if the frequent items in all of the transactions are sorted according to an original fixed order.

An Illustrative Example for Mining Closed Frequent Calling Patterns

Example 1 using Table 1, let the first two columns be the transaction database, TDB and setting the minimum support to 2. The Frequent-item projection is shown in the third column.

Note: The Transaction ID and Items used is just for ease of explanation, it could represent, for example Callersname and Codeused respectively. Each of the code used must have been coded in order to minimize storage space e.g. a, b, c…

Table 1: The Transaction database (to be used as the running example)

Transaction ID	Items	Freq(X)
10	a, c, d, e, f	a, c, d, e, f
20	a, b, e	a, e
30	c, e, f	c, e, f
40	a, c, d, f	a, c, d, f
50	c, e, f	c, e, f

Step 1 - By scanning the database once, the complete set of frequent items (a:3, c:4, d:2, e:4, f:4) are found and output which forms the *F_List.* Note that item *b* has been pruned because its support is 1 and does not meet up with the *min_sup* = 2.

Following the order of the *F_List*, the complete set of closed frequent patterns can be partitioned into 5 subsets as follows:

(1) those containing item *a* (2) those containing item *c* but no item *a* (3) those containing item *d* but no item *a* nor *c* (4) those containing item *e* but not item *a* nor *c* nor *d* (5) those containing only item *f.*

Step 2 - Construct the L-struct for the frequent-item projection in Table 1

Definition: L-struct is a data structure defined below:

(1) It consists of a Lookup table, *L* (user-defined) such that every occurrence of a frequent item is stored in an entry with three fields: an *item-id*, the *support count* and a *node-link*. There is also a structure called *TransLink* which constitutes one for each of the frequent-item projection with two fields: an *item-id* and a *node-link*, and the other one identifying the first by its Trans-id, (denoted as T*j*, where T represents each structure and *j* denotes each transaction). (see Figure 1)

(2) When the frequent-item projections are loaded into memory, those with the same first item (in the order of *F_List*) are linked together as a queue, and the entries in the Lookup table as the head of the queue

Step 3 - Mining the L-struct to generate closed frequent patterns

Based on this definition, the remaining of the mining can be performed on the *L-struct* only without referencing any information in the original database. After that the sets of frequent patterns will be generated, each of these mined one by one to generate the closed frequent patterns as follows:

- After scanning the TDB once and having collected the frequent item projections of each transaction.
- Create a table-like structure labeled “L- struct” as described in definition 6.1. Its support count registers the occurrence of each frequent item in the transaction database.
- Following the subsets of the frequent items, to mine the *i*-projected database (where *i* represents each frequent item), create a new structure that registers only the postfix items of each of the subsets, L_i-Lookup table. The support count of these items records the occurrence of the corresponding item with *i* in the each frequent item projection. Then do the following:

  Let the sorted items in the L_i Lookup table be *[i | I],* where *I* denotes each item in the postfix item-list. Call insert_struct(*[i | I]*), The function insert_struct(*[i | I]*) is performed as follows:

  Check the occurrence of *I* in the L*i* Lookup table, if *I* occurs with *i* in the T_j table, increment its count by 1. In this case, the set of locally frequent items i.e. the items appearing at least the defined *min_sup* times in the *i*-projected database is found. Then each *I* will now be combined with *i* to form iI_1, iI_2, …, iI_n, where n is the number of frequent patterns generated from L*i*. After getting each iI_k *(where k = 1, n)*, then build up links for L*i.*

  Note: the *i*-projected database consists of all the frequent-item projection containing item *i.*

- Going by the definition of closed frequent patterns in section 2, the set of frequent patterns got (including *i* itself) from the above algorithm are checked to generate the closed frequent patterns whose procedure goes thus:
  - ✓ First check for which of these itemsets is closed, if there is, then this is output while Lookup table is created for the remaining ones so far they are frequent. This Lookup table i.e. L_{iIk} (where $k = 1, n$) will be drawn using the L_i Lookup table. The same process for finding frequent patterns is applied and then check for their corresponding closed itemsets. This continues until all the possible combinations have been found.
  - ✓ After the closed frequent itemsets containing item *i* are found, the *i*-projected database i.e. *i*-queue is no longer needed in the remaining of mining. To traverse the *i*-queue once more, each frequent item projection in the queue is appended to the queue of the next item in the projection following *i* in the *F_List.*

Illustration

Following the highlighted steps, the CLOTELE algorithm is illustrated using Table 1 in Example 1. On getting the frequent-item projection, Figure 1 shows the L-struct.

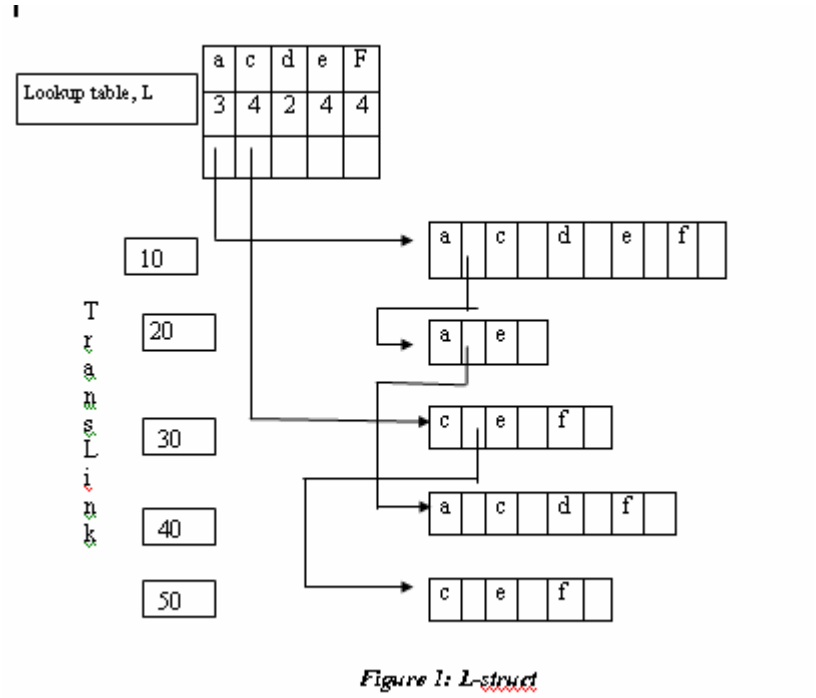


Figure 1: L-struct

To mine the *a*-projected database, create *a*-Lookup table, L_a as shown in Figure 2. In L_a, every frequent item except for *a* itself, has an entry with the same three fields as *L*. The support count in L_a records the support of the corresponding item in the *a*-projected database. For example item *c* appears twice with *a* in the *a*-projected database, thus the support count in entry *c* of L_a is 2

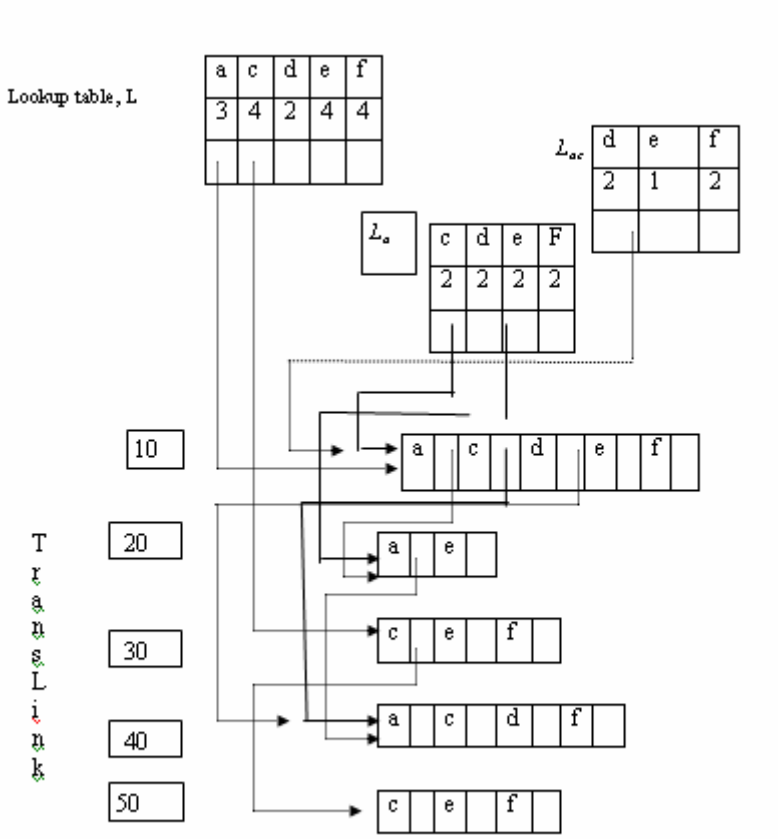


Figure 2: Lookup table, L_a

By traversing the *a*-queue once, the set of locally frequent items, i.e., items appearing at least 2 times, in the *a*-projected database is found, which is {c:2, d:2, e:2, f:2}. This scan outputs frequent patterns {ac:2, ad:2, ae:2, af:2}. A link is now built for L_a table as shown in figure 2. To check for closed itemsets, item *a* is contained in 3 transaction-IDs i.e. [10, 20, 40], whereas *ac*, *ad*, *ae*, *af* are not contained in same transactions as *a*, thus {*a*:3} is closed. It remains to check whether the frequent patterns are closed.

Similarly, the process continues for the *ac*-projected database by examining the *c*-queue in L_a, which creates an *ac*-Lookup table. This outputs items *d* and *f* with same supports:2 and thus considered to be frequent, and item {*e*:1} which is infrequent. The frequent patterns {*acd*:2, *acf*:2} are generated, which are now checked for closed patterns. Since *acd* is contained in same transaction as *cd*, and *acd* being a larger set, it is then said to be closed. Likewise, the itemset *acf* is also closed in *ac*. This outputs only {*acd*:2, *acf*:2} as closed itemsets.

Each of these will now be checked to see if there can be any of its combination that can be closed. The L_{ac} table will be used by dynamically changing its link for *acd* and *acf*.

For *acd*-projected database, there are {e:1, f:2}, only *f* is frequent and thus {*acdf:2}* is output as frequent pattern. Since there is no larger set of it, then it is considered to be closed.

For *acf*-projected database, *d* has already been considered and there is no link for item *e*, thus *acf:*2 is output as closed itemset. The same process continues for *ad*-projected database using the L_a table, then the recursion backtracks to find patterns containing *a* and *d* but no *c*, (item *c* has been considered in the mining of *ac*), this outputs {e:1, f:2}, only item *f* is frequent with *ad.* Thus,

pattern {*adf*: 2} is frequent. To check whether it is closed or not, it occurs in same transaction as *ad* and there is no larger set of it, so it is considered to be closed.

For *ae*, from L_a table, only item *f* will be considered whose support is 1 and thus not frequent. Hence, itemset {*ae*:2} is closed (note: items *c* and *d* have been considered).

There will be no link for *af* in the L_a table, since all other items have been considered, thus considered closed.

The overall output from the *a*-projected database includes {a:3, adf:2, ae:2, acdf:2, acf:2, af:2}. The last scan is to check for any duplication in the closed patterns generated: On cross-checking, it is discovered that itemsets: {*acdf*: 2}, {*af*:2}, {*adf*:2}, {*acf*:2}, {ae:2} all occur in same transactions where {*acdf*: 2} is the superset of all, thus picked as the closed frequent itemsets with {a:3} which occurs in 3 transactions. Summarily, the set of closed frequent patterns after removing duplication in the *a*-projected database is the set {*a*: 3, *acdf*: 2}.

After the closed frequent patterns containing item *a* has been found, the *a*-projected database i.e. *a*-queue is no longer needed in the remaining of the mining.

By mining the *c*-projected database recursively, we can find the set of closed frequent patterns containing item *c* but no *a*. Notice item *a* will not be included in the *c*-projected database since all the closed patterns having *a* have already been found.

Similarly, the mining goes on. It is easy to see that the above mining process finds the complete set of closed frequent patterns without duplication.

Notice also that the depth-first search for mining the first set of closed frequent patterns at any depth can be done in one database scan by constructing the look-up tables at all levels simultaneously.

Algorithms for *Construct, Mine-Fi, Clo-Mine*

Procedure Construct – L_struct
// for constructing the L_struct
Let supp = support-count
For all transactions in TDB
    If item *x* ϵ itemset < min_sup
        Delete *x* from itemset
        Else
            Insert item *x* into Frequent-item Projection
    EndIf
    For all item ϵ Frequent-item Projection
        If item ϵ Lookup table, *L*
          increment supp
        else
          insert item into Lookup table with *supp* = 1
        EndIf
    EndFor
    For itemset ϵ Frequent-item Projection
        insert itemset into TransLink
    EndFor
    If item x_1 in TransLink ϵ item in Lookup, *L*
        connect node-link of previous occurrence to the item
        elseif itemset in TransLink(i) itemset in TransLink (j)
    EndIf

```
EndFor
Procedure MineFI – L-struct
// for generating the frequent items
Let X_p := item-list of postfix of x
For all x-projected database ϵ Lookup table, L_i
  For all X_p in L_i
   Add X_p into Lookup table, L_i+1
   initialise supp = 0
   If item (x + X_p) in TransLink
     Add 1 to supp
   EndIf
  EndFor
Let supp = support count
  For all support-count(x + X_p) ≥ min_sup
    Add (x + X_p) into set of Frequent itemsets
     connect node-link of frequent item in L_i to x + X_p in TransLink
      CloMine(y,z,s)
  EndFor
Procedure CloMine(y, z, s)
// for generating closed frequent itemsets
y := the projected item being mined
z := concatenation of item being mined and its postfix
tid := transaction identifier
s := supp
Add y to set of frequent itemset, F
For all frequent itemsets, F
  If s(F_i) = s(F_j) [where i = 1, n and j = 2, n]
        If tid(F_i)  = tid(F_j)
                Add F_i and F_j to closed itemset
        Else
          For all itemset k ϵ F_i and k ϵ F_j
                If size(F_i) > size(F_j)
           Delete F_i from list
                EndIf
          EndFor
        EndIf
        Else
        Add F_i to closed itemset
   EndIf
EndFor
// to check for any combination of closed patterns generated that could be closed
Construct;
End Procedure
```

Experimental Results

In this section we evaluate the correctness of the closed frequent calling patterns discovered by algorithm CLOTELE. Without loss of generality, we selected a 7 months old telecommunication database of size 11.8MB, tested on different minimum support in order to show the flexibility of the algorithm. All the tests were performed on a 733MHz Pentium III PC, with 128MB RAM and

20GB HDD running Microsoft Windows XP. CLOTELE is written in Microsoft Visual Basic.NET.

The database is represented as *T60I35D660K*, where T represents the number of Caller's names (i.e. 60); I represents the numbers of areas called (i.e. 35); and D represents the number of records in the database (i.e. 660,000).

Figure 3 shows the flexibility of our algorithm, with the growth of pattern length, the support threshold is decreasing. As the support threshold goes down there is a change in the number of patterns generated. Figure 3 reports the difference between the numbers of patterns generated in situations of applying various support constraints. From figure 3, it is easy to see that if the support constraints decrease, the number of closed calling patterns increases dramatically.

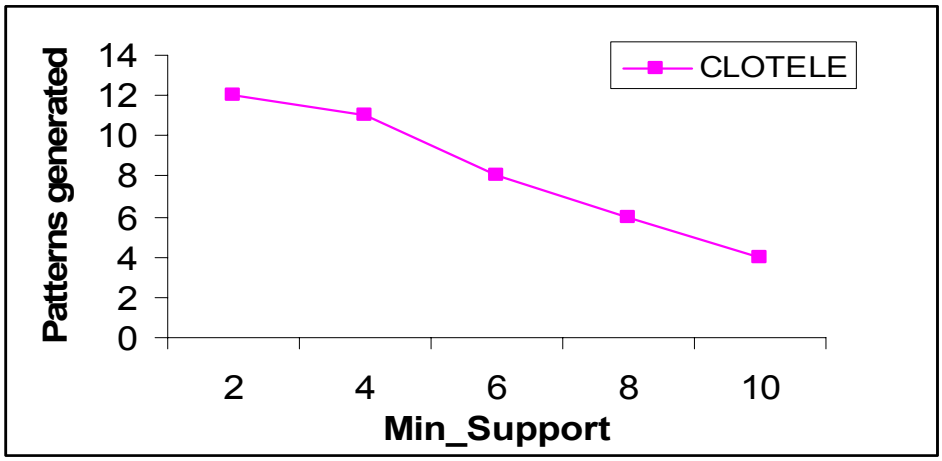


Figure 3: The Minimum-Support with the number of closed patterns generated

Below are some examples of the mining results obtained on implementation of the algorithm on *T60I35D660K*. The result obtained is in form of

<Abuja, Econet, lagos, MTN - 12> : this implies that 12 of the staff make call to each of the areas in the order listed.

<Econet - 61> : this implies that 61 of the staff make calls to Econet.

<Benin, Econet, Lagos, MTN - 6>: this implies that 6 of the staff make calls to the areas in the order listed.

It is worth mentioning that the closed frequent patterns generated are beneficial to the management in knowing the proper steps to take in making crucial decision

Conclusion

In this paper, we devised a solution procedure for mining closed frequent calling patterns from the calling data of a telecommunication database from a telecommunication switch provided by a telecommunication provider. First, by observing the features of the database, the data cleaning procedure involved is collecting the attributes needed to be mined. After the data cleaning procedure, the items found in the attributes are merged into a set of sequences. By integrating the advantages of previous pattern growth algorithm, we designed CLOTELE to discover the useful closed frequent calling patterns. Algorithm CLOTELE is implemented on *T60I35D660K*. The result obtained is useful for the telecommunication network operators for making crucial decision.

Acknowledgement

We would like to express our thanks to Charles and Kay of Paperless Solution for introducing us to the telecommunication provider.

References

Daszczuk, W., Gawrysiak, P., Gerszberg, T., Kryszkiewicz, M., Miescicki, J., Mieczyslaw, Okoniewski, M. et al. (2000). Data mining for technical operation of telecommunication companies: A case study. Retrieved from www.ii.pw.edu.pl/english/research/RAPORT03

Han, J. & Kamber, M. (2001). *Data mining: Concepts and techniques*. Academic Press

Han, J., Pei, J. & Yin, Y. (2000). Mining frequent patterns without candidate generation. *Proceedings 2000 ACM-SIGMOD International Conference on Management of Data (SIGMOD'00).*

Hatonen, K., Klemettinen, M., Mannila, H., Ronkainen, P. & Toivonen, H. (1996). Knowledge discovery from telecommunication network alarm databases. Retrieved from www.isbn.nu/toc/0818672404

Lucchess, C., Orlando, S. & Perego, R. (2004). Mining frequent closed itemset without duplicates generation. *Proceedings of 2004 ACM-SIGMOD International Conference on Management of Data*.

Pasquier, N., Bastide, Y., Taouil, R., & Lakhal, L.. (1999). Discovering frequent closed itemsets for association rules. *Proceedings 7th International Conference Database Theory (ICDT'99).*

Pei, J., Han, J., & Mao, R. (2000). CLOSET: An efficient algorithm for mining frequent closed itemsets. *Proceedings 2000 ACM-SIGMOD International Workshop Data Mining and Knowledge Discovery (DMKD'00).*

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. C. (2001). PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. *Proceedings 2001 International Conference Data Engineering (ICDE'01).*

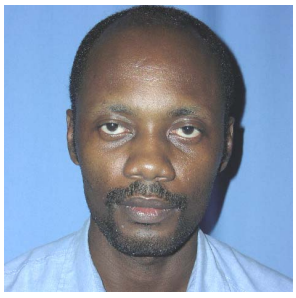
Thearling, K. (1996). An introduction to data mining. Retrieved from http://www.thearling.com/text/dmwhite/dmwhite.htm

Wu, P-H., Peng, W-C., & Chen, M-S. (2001). Mining sequential alarm patterns in a telecommunication database. Retrieved from www.ee.ntu.edu.tw/~mschen/paperps/vldb2001

Zaki M. & Hsiao C. (2002). CHARM: An efficient algorithm for closed itemset mining. *In SDM'02* April.

Biographies

IBRAHIM, Saidat Adebukola (nee OKUNLAYA) was born on 6th April, 1974. She married. She had her education in University of Agriculture, Abeokuta with M.Sc Computer Science (Data Mining) in year 2004, B.Sc Mathematical Sciences (Computer Science option) in year 2000. She was the best graduating student in her set.



AJAYI, Olutayo Bamidele.is an information technology specialist/system programmer. Born on 4th May, 1967 and married with two (2) children. He had his degrees in University of Ibadan, Ibadan (1996 – 1997) with Post Graduate Diploma in Petroleum Engineering, University of Agriculture, Abeokuta (1987 – 1991, 2002) with B.Sc Mathematical Sciences (Computer Science option), M.Sc Computer Science (Management Information Science) and P.hD Computer Science (in view). He had worked for 13 years in service with various International Conferences and local ones too. He had published 3 journals and 2 others in press. I am a challenge driven lateral thinker with a relaxed style to achieving set goals, pro-active with excellent communication skills. A great team player with excellent interpersonal skills, coupled with willingness to adapt and a flexible approach to life. Always eager to learn new things and able to work effectively at all levels within organisations.