Discovery of Frequently Occurring Approximate sub sequences with distance

Ms. V. V. Kamble [*], Mr. S. V. Kamble [**]

[*] COEP Pune, India
[**] Flora Institute of Technology, Pune, India

Abstract- Frequent pattern mining from sequential datasets is an important data mining method. It has various applications like discovery of motifs in DNA sequences, financial industry, the analysis of web log, customer shopping sequences and the investigation of scientific or medical processes etc. Motif mining requires efficient mining of approximate patterns that are contiguous. The main challenge in discovering frequently occurring patterns is to allow for some noise or mismatches in the matching process. Existing algorithms focus on mining subsequences but very few algorithms find approximate pattern mining. In this paper we have presented a new method for finding frequently occurring approximate sequences from sequential datasets. Proposed method uses suffix trees for discovering frequent pattern with fixed length, maximum distance & minimum support.

Index Terms- data mining; sequence mining; frequent patterns; suffix tree;

I. INTRODUCTION

Sequential pattern mining deals with data in large sequential data sets. Sequence mining has gained popularity in marketing in retail industry, biomedical research, DNA sequence patterns, financial industry. It is most common applications are discovery of motifs in DNA sequences, financial industry use data mining to identify interesting share price movements, the analysis of web log for web usage, customer shopping sequences and the investigation of scientific or medical processes and so on. The results of pattern mining can be used for business management marketing, planning and prediction. The difficulty in discovering frequent patterns is to allow for some noise in the matching process. The most important part of pattern discovery is the definition of a pattern and similarity between two patterns which may vary from one application to another.

Agarwal has started work in this area for association rule mining. It analyses customer buying behaviour by finding associations between the different items that customer's buy. Discovering frequent patterns is computationally expensive process and counting the instances of pattern requires a large amount of processing time. Huge amount of literature is available and number of algorithms is proposed for mining the frequent patterns or itemsets. These algorithms differ in their ways of traversing the itemset lattice, dimension and the way in which they handle the database; i.e., how many passes they make over the entire database and how they reduce the size of the processed database in each pass.

In this paper we focus on repeated occurrences of short approximate sequences, i.e. occurrences are not always identical such sub sequences is also called as frequently occurring approximate sequences. There are two types of subsequence, non-contiguous subsequence and contiguous subsequence. If sequence A=abcacbcc and B=abccb then sequence B is non-contiguous subsequence of A by choosing first, second, third, fifth, and sixth element from sequence A. We focus on discovering the contiguous subsequence of fixed length because non-contiguous subsequence mining is not applicable in DNA and protein sequence mining applications. Some algorithms [1] [2] [3] [4] are available to mine contiguous sub sequences. We have presented a method for discovering frequently occurring approximate pattern of fixed length. It uses suffix tree because suffix tree is a data structure that presents the suffixes of a string from sequential dataset that allows for a particularly fast implementation. This approach is applicable in many real life and biomedical applications such as bioinformatics for finding patterns in long noisy DNA sequence and protein motif mining. Section 2 presents related work, section 3 describes problem definition, section 4 describes proposed work and section 5 describes summaries.

II. RELATED WORK

J. Han and M. Kamber[5] stated, data mining is process of extracting knowledge from large database. Data mining is also referred as knowledge discovery from data or KDD. There exist a large number of algorithms for sequential pattern mining and each algorithm is having different features. Early work focused on mining association rules. Later AprioriAll algorithm is derived from the Apriori algorithm. In these types of algorithms, candidate sequences are generated and stored. After these algorithms main goal is to reduce the number of candidate sequences generated so as to minimize input output cost. In another type of algorithm support is counted and it is used to test the frequency. The key strategy here is to eliminate any database or data structure that is maintained all the time for counting the support. Usually, a proposed algorithm also has a proposed data structure, such as SPADE by Zaki 1998 uses vertical databases; PrefixSpan[6] uses projected databases; FLAME[8] uses count suffix tree. current Algorithms in the area can be classified into three main categories, namely, apriori-based, pattern-growth, and early-pruning with a fourth category as a hybrid of the main three [7].

The problem of subsequence mining was introduced in [9]. Later on several other algorithms have proposed as improvement

on [9] such algorithms are SPADE [10] and BIDE [1]. Statistical sampling based method [11] uses compatibility matrix to find patterns in presence of noise. Most of the algorithms like CloSpan[2] are presented to mine exact motifs, such algorithms does not allow noise in matching process. FLAME [8] algorithm is discovering motif and it allow noise in matching process, so it is efficient for approximate substrings. FAS-Miner [12] algorithm can discover patterns of longer lengths and higher supports. It uses suffix array to store suffixes and sorted in lexicographic order. [13] Proposed an algorithm for finding frequent approximate sequential patterns. It uses hamming distance model and break-down-and-build-up methodology.

III. Problem Definition

In this paper we have presented a method to efficiently mine all frequently occurring approximate substring. Many challenges arise in sequential mining such as projection and prefix extension techniques need more space. Another challenge is allow some noise in matching process because every substring needs to compare with other substring for checking mismatches. We want to find the instances of the pattern in the presence of noise but we do not want that it matches unrelated subsequences which may have large number of mismatches. The input sequence database is composed of symbols from a finite discrete alphabet set. In the case of time series database, it can be converted into a sequence database. Time series database can be converted into a sequence database. Let Σ denote the finite discrete alphabet set. An input sequence S is ordered list of items or events. For example, S is a DNA Sequence over {A, C, G, T}. The i-th item of a sequence S can be denoted as S[i]. Consider two substings p and q of S having same length n. The hamming distance d(p,q) of two strings p and q is the number of mismatching characters [12].

$$d(p, q) = |I|$$
$$I = \{ i \mid pi \neq qi, 1 \leq i \leq n\}$$

In this model, two strings are considered approximately the same if and only if they are of equal length and their distance is less than or equal to a user specified distance. Here we are considering the (L, d, k) model for discovering patterns, In this model L denotes the length of the frequent pattern string, d denotes the maximum number of mismatches i.e. hamming distance between pattern string and instance of that string and k denotes the minimum support that is minimum number of instances of pattern in input sequential database. Such model is commonly used for finding DNA motif in computational biology. Proposed approach outputs the model strings that had sufficient support that is the support is greater than or equal to predefined k.

IV. Proposed Method

In this section, we present a new method for discovering frequent approximate sequences that can be used for pattern mining. Throughout this section, we will assume that the input sequence i.e. dataset is composed of symbols from a discrete alphabet set and outputs the model strings that had sufficient support along with its all instances. We have proposed method for model (L, d, k) and tried to improve the performance of existing algorithm presented in [8]. In model (L, d, k), L is the length of the pattern, d is the maximum distance within which two strings are considered similar and k is the minimum support or frequency required for a valid patterns. In proposed method we are considering only those strings of length L that actually occur in the dataset and compute the support for each of them by scanning the dataset. This approach will not discover patterns as the model string might not actually occur in the dataset even once.

Input:

1. Input sequence is composed of symbols from a discrete alphabet sets.

Set of items/alphabets I= {a1, a2, ..., am}
Input Sequential database D = { x | ∀x ϵ I}
E.g. I = {A, B, C, D}
D = ACABBDACCAB

2. Values of L, d and k.

Where
L: Length of substring
d: Distance/number of mismatches allowed
k: Minimum Support

Output:

Set of (L, d, k) model Strings.
Op = {s1, s2, ..., sn}
Where
Si = {si1, si2, . . .,sim} is set of instances for si.
Op = {si| d(si, sij) ≤ d, |sij|=L and m ≥ k}

System:

```
FreqPattern(mTree, root, L, d, k)
1.      process();
2.      getmodelTree();
3.      For i=0 to i<mTree.size()
4.              computeSupport();
5.      End For

process()
Construct count suffix tree on input dataset.

getmodelTree()
Construct suffix tree of depth L on strings of length L that occurs in the count suffix tree.

computeSupport()
1.  oldMatches=model.parent.matches;
2.  if(model.parent.id=1)
3.      newmatches=expandMatches()
4.  else
5.      for k=0 to k<oldMatches.size()
6.          newMatches.addAll(expandMatches())
7.      End for
8.  setMatches(newMatches)
9.  for i=0 to modelTree.size()
10.     x=modelTree.get(i)
11.     if(distance()<=d)
12.         model.matches.add(x)
13.         model.modelSupport +=x.support;
```

14. End for
15. End

Fig. 1. Pseudocode for proposed method.

Frequently occurring pattern of model (L, d, k) a string of length L that occurs minimum k times in the input dataset, with each occurrence being within a Hamming distance of d from the model string. Hamming distance between strings S and P is shown in fig. 1 from [12].

S = AABCACDCABCD
P = AACCABDCABCD

Fig. 2. Hamming Distance between string S and P are 2

In this example Fig. 2 hamming distance between string S and P is 2 (area which is not highlighted). We first construct a count suffix tree on the actual dataset. Then construct a model suffix tree on those strings of length L that actually occurs in the dataset i.e. model tree is constructed for count suffix tree. The pseudo code of proposed method is given in Fig. 1. The method process() construct the count suffix tree on input dataset, which helps to quickly compute the support. Method getmodelTree() construct the model suffix tree from count suffix tree, which is of depth L on set of strings of length L. The pruning strategy is similar as explained in FLAME[8]. All the strings of length L staring with the same symbol makes one partition. This partition corresponds to all the nodes in the model suffix tree under the subtree corresponding to same node. These partitions continue on for L levels, and at the last level, we have only one model string for each partition. At each node in the model suffix tree, the routine computeSupport() is called, which computes the list of matches and the new support. At any node, if algorithm finds that support is lower than k, it prunes away that that subtree in the model suffix tree, and if it finds a model of length L with the required support, it outputs the result in the form shown in Fig. 3 below.

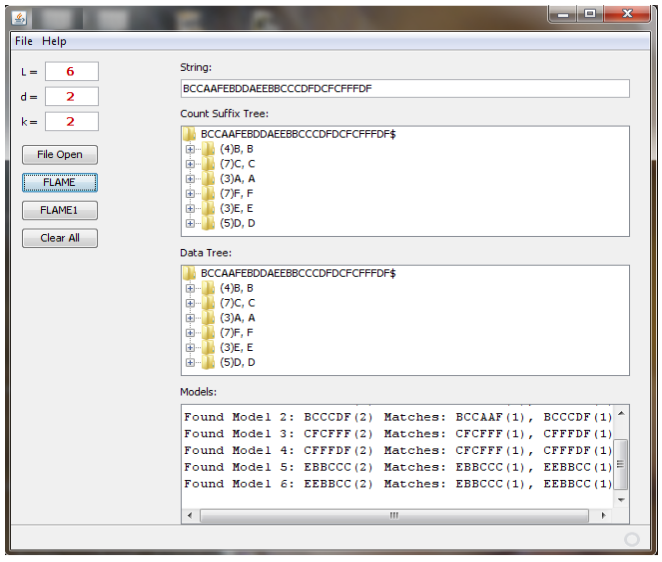

Fig. 3. Graphical User Interface.

V. Evaluation

In this section, we compare proposed method with FLAME algorithm. We present results from various experiments to test the effectiveness and performance of proposed method.

Comparison with FLAME algorithm

We compare FLAME and proposed method by performing a typical (L,d,k) motif mining task on DNA datasets. We vary the length of sequences from 200 to 1200 symbols. Both algorithms try to find (L,d,20) motifs for d=1,2 and L varying from 6 to 8. For the task of finding motifs with L varying from 6 to 8, and d=1 (denoted as (6-8,1,20) in Figure 4), the FLAME algorithm works well for small database sizes. However, as the database size increases, we see that its performance begins to decrease. For the (6-8,2,20) task, FLAME takes larger time to complete sequence lengths beyond 600. Proposed method takes relatively very less time as compare to FLAME algorithm.

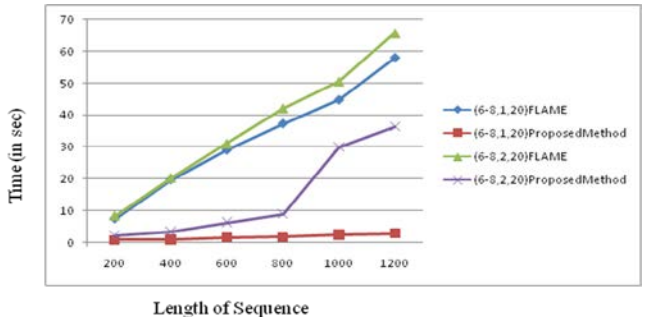

Fig. 4 FLAME vs Proposed Method for varying database sizes

Proposed method is a simple algorithm that finds all patterns that appears more frequently than expected. In proposed method, model tree is constructed from data tree i.e. considering only those strings of length L that actually occur in the dataset. FLAME algorithm builds model tree on all possible model string so it discovers large number of patterns than proposed method. We demonstrate this behavior using a synthetic dataset containing 12800 symbols long. We run FLAME and proposed method on a variety of (L,d,k) models. The results of this experiment are presented in Fig. 5.

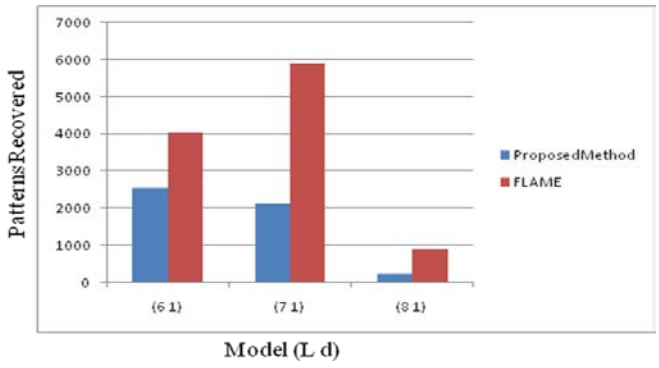

Fig. 5 FLAME vs Proposed Method for number of patterns recovered.

Test Report:

This approximate subsequence mining problem is of particular importance in computational biology, where the challenge is to detect short sequences, usually of length 6-15, that occur frequently in a given set of DNA or protein sequences. In this white box testing, we check system works for lengths 6-15.

Mining Time Series Data: We study the performance of proposed method for different parameters of the (L, d, k) model. In this experiment we use (L,d,k) model to mine the stock price dataset. We have decided the ranges and then assigned a symbol to each range and encoded the numerical series into a symbolic sequence. The dataset totaled about 9331 symbols. We present the time taken by proposed method to find several (L,d,k) models.

We run proposed method for L=5,8,11 and 14, while varying the distance d and support 21. The results of this experiment are shown in Fig. 6. We observe from the figure that as the distance is increased the time taken to execute the search increases. Proposed method is able to find models of length 14 within 15118 ms.

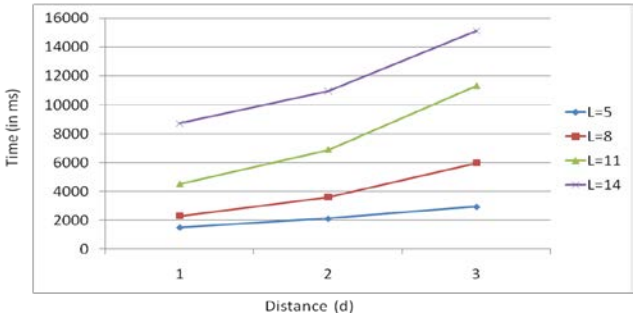

Fig. 6 Proposed Method: Distance vs time taken on stock price data.

VI. Conclusions

Proposed method will discover frequent approximate patterns of model (L, d, k). (L,d,k) model is a string of length L that occurs k times in the dataset, with each occurrence being with a Hamming distance of d from the model string. The dataset and the values of L, d, and k are specified as input.

System first construct two suffix trees: a count suffix tree on the actual dataset and a suffix tree on the set of all possible model strings present in count suffix tree. The model suffix tree is typically the set of all string of length L from count suffix tree. Count suffix tree is merely a suffix tree in which every node contains the number of leaves in the subtree rooted at that node.
The difficulty in discovering frequent patterns is to allow for some noise in the matching process. It's most common applications are discovery of subsequences in DNA sequences, financial industry use data mining to identify interesting share price movements, the analysis of web log for web usage, customer shopping sequences and the investigation of scientific or medical processes and so on. The results of pattern mining can be used by business management for marketing, planning and prediction.

A series of experiments on real and synthetic datasets, we demonstrate that proposed method can be used in several real pattern mining tasks. Proposed method is significantly faster than FLAME. We also presented experiments which show that proposed method discover patterns actually occur in the dataset but FLAME discover patterns that might not actually occur in the dataset even once.

VII. Future Enhancement

Proposed method discovers patterns of (L,d,k) model where L is length of pattern, d is maximum distance and k is minimum distance. Sometimes the exact length of the pattern is not known but has a rough idea of the range in which it may lie. One often ends up trying several (L,d,k) values such as (6-15, 2, 20), (6-16,1,15) etc. Thus future scope in this dissertation is to add functionality for discovering such patterns.

References

[1] V. Kamble, Emmanuel M and A. Phakatkar, " Novel Approach for Discovery of Frequently Occurring Approximate Sequences," ICCA Conference, 2012.

[2] J. Wang and J. Han, "BIDE: Efficient Mining of Frequent Closed Sequences," in ICDE, 2004, pp. 79–90.

[3] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining Closed Sequential Patterns in Large Datasets," in SDM, 2003.

[4] M. J. Zaki, "Sequence Mining in Categorical Domains: Incorporating Constrains," in CIKM, 2000, pp. 442–429.

[5] J. Pei, J. Han, and W. Wang, "Mining Sequential Patterns With Constraints in Large Databases," in CIKM, 2002, pp. 18–25.

[6] J. Han, and M. Kamber, Data Mining Concepts and Techniques. Morgan Kanufmann, 2000.

[7] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth," in ICDE, 2001, pp. 215–224.

[8] R. Nizar, Mabroukeh and C.I. Ezeife, "A taxonomy of sequential pattern mining algorithms," ACM Comput. Surv. 43, 1, Article 3, 2010.

[9] A. Floratou, S. Tata and J. M. Patel, "Efficient and Accurate Discovery of Patterns in Sequence Data Sets," IEEE Trans. Knowledge and Data Engineering, vol. 23, pp. 1154-1168, Aug. 2011.

[10] R. Agrawal and R. Srikant. Mining sequential pattems. In proc. 1995 Int. Conf Data Engineering (ICDE'95), pages 3-14, Taipei, Taiwan, Mar. 1995.

[11] M. J. Zaki, "SPADE: An Efficient Algorithm for Mining Frequent Sequences," Machine Learning, vol. 42, no. 1/2, pp. 31–60, 2001.

[12] J. Yang, W. Wang, P. S. Yu, and J. Han, "Mining Long Sequential Patterns in a Noisy Environment," in SIGMOD, 2002, pp. 406–417.

[13] X. Ji and J. Bailey, "An efficient technique for mining approximately frequent substring patterns," Seventh IEEE International Conference on Data Mining, pp. 325-330, 2007.

[14] F. Zhu, X. Yan, J. Han and P. Yu, "Efficient Discovery of Frequent Approximate Sequential Patterns," Seventh IEEE International Conference on Data Mining ICDM, pp. 751-756, Oct. 2007.

[15] V. Makinen and G. Navarro, "Compressed Compact Suffix Arrays," in Proc. Of Combinatorial Pattern Matching (CPM), vol. 3109 of Lecture Notes in Computer Science, 2004.

Authors

First Author – Ms. V. V. Kamble, COEP Pune, India
Second Author – Mr. S. V. Kamble, Flora Institute of Technology, Pune, India