Hierarchical Sequence Clustering Algorithm for Data Mining

V. Umadevi Chezhian[1], Thanappan Subash[2], M. Ragavan Samy[3]

ABSTRACT - Bioinformatics emerged as a challenging new area of research and brought forth numerous computational problems. Here computers are used to gather, store, analyze and merge biological data. In this paper, the problem of clustering interval-scaled data and sequence data is analyzed in a new approach using Hierarchical Sequence Clustering. In Sequence clustering, it is necessary to find the similarity or distance between each pair of sequences. To find the similarity between sequences the data structure Probabilistic Suffix Tree can be used. An agglomerative algorithm is introduced based on UPGMA (Un weighted Pair wise Group Average Method) cluster analysis, that required O(n^3) of total computing time. Then a new algorithm using the new approach is introduced with O(n^2) computing time. The result of this new algorithm is compared with UPGMA cluster analysis.

Key words: Data Mining, Hierarchical Clustering, Sequence Clustering, Probabilistic Suffix Tree, UPMGA.

I. INTRODUCTION

Cluster analysis has received renewed attention within the last 10 years as a field of study within knowledge discovery and data mining. The massive amounts of data that have become available in a variety of fields have prompted new research in the use of traditional clustering algorithms and the development of new algorithms [1].

The studies of sequential pattern mining have been extended in several different ways [2] considered frequent episodes in sequences, where episodes are essentially acyclic graphs of events whose edges specify the temporal before-and-after relationship but without timing-interval restrictions. Sequence pattern mining for plan failures, the use of regular expressions as a flexible constraint specification tool that enables user-controlled focus to be incorporated into the sequential pattern mining process[3] [4]. The embedding of multidimensional, multilevel information into a transformed sequence database for sequential pattern mining [5] [6] studied issues regarding constraint-based sequential pattern mining. CLUSEQ is a sequence clustering algorithm [7], An incremental sequential pattern mining algorithm, IncSpan [8], SeqIndex efficient sequence indexing by frequent and discriminative analysis of sequential patterns [9], A method for parallel mining of closed sequential patterns [10] and a new efficient incremental clustering algorithm using weighted distance metric.

[1]Lecturer, Department of Computer Science, College of Business and Economics, State of Eritrea. Email: yazh1999@gmail.com
[2]Lecturer, Department of Civil Engineering, Eritrea Institute of Technology, State of Eritrea. Email: thanappansubash@gmail.com
[3]Senior System Engineer, Singapore Refinery Company Pvt. Ltd., Singapore. Email: ragavansamy@gmail.com

A tutorial on suffix tree construction [11] [12] and other space-saving techniques for suffix trees that allow linear-time construction [13], the PST (Probabilistic Suffix Tree) in detail to model the biological sequences [14] [15].

A PST is considered to have an enhanced memory efficient representation than the PSA (Probabilistic Suffix Automata). Since then, it has been used in several domains as an efficient approach for classifying sequences [16] [17].

Proposal of a suffix tree construction method with expected time complexity O (n log n) and it is well suited for the construction of large suffix trees in bioinformatics and other applications, as long as the main memory size is six times as big as the sequence length [18]. Another suffix tree construction algorithm over a wide spectrum of data sources and sizes [19] [20] constructed a PST that represents the probability of each system call given a finite-length sequence of previously observed system calls.

A PST is considered to have an enhanced memory efficient representation than the PSA (Probabilistic Suffix Automata). Since then, it has been used in several domains as an efficient approach for classifying sequences.

In Sequence clustering, it is necessary to find the similarity or distance between each pair of sequences. To find the similarity between sequences the data structure Probabilistic Suffix Tree can be used [21] [22] [23] proposed an algorithm to construct a Suffix Tree for a string of length n, in O(n) time.

II. SEQUENCE CLUSTERING

Let $\Im = \{s_1, s_2, \ldots .. s_n\}$ be the set of all possible symbols.

A sequence is an ordered list of symbols in $\Im$. The number of symbols in a sequence is referred to as the length of the sequence. Given a sequence, a segment is defined as a consecutive position of the sequence. For example, ‘bcd’ is a segment of ‘abcdef’ while ‘abd’ is not. Conventionally, we use the term ‘sequence’ to refer to a whole symbol sequence in the databases while the term ‘segment’ is used to denote a portion of some sequence. A sequence database is a set of sequences. Given a sequence database our objective is to categorize these sequences into clusters according to their sequential similarities.

Proposal of a suffix tree construction method with expected time complexity O (n log n) and it is well suited for the construction of large suffix trees in bioinformatics and other applications [18], as long as the main memory size is six times as big as the sequence length. Another suffix tree construction algorithm over a wide spectrum of data sources and sizes constructed a PST that represents the probability of each system call given a finite-length sequence of previously observed system calls[19][20].

III. UPGMA ALGORITHM

Given a set of n objects and each having number of characteristics (attributes)

```
1. Find Euclidean distances d(i,j) for all 1<= i, j<=n
   and i!=j
2. Form the dissimilarity matrix D.
3. Repeat step 4 to step 7 n-1 times.
4. Find the minimum value in the lower triangular
   of D Matrix and let the corresponding row and
   column be t1 and t2 respectively.
5. a=count(objects(t1))
   b= count (objects (t2))
   r=a + b
6. Rearrange the elements as:
   For each object i other than t1 and t2
   d(i,t2) = a/r * d(t2,i)+b/r*d(t1,i)
   d(t2,i)=d(i,t2)
   d(i,t1)=-1
   d(t1,i)=-1
7. concatenate object(t2),object(t1)
```

In the 4th step, consider only the lower triangular matrix for finding the minimum, and the corresponding row value is stored in t1 and the corresponding column value is stored in t2. In the next step, count the number of objects in t1 and t2. In the first time it will be 1 for t1 and t2. Thus a=1, b=1, and r=2. In the 6th step, the value of row and column corresponding to t1 is set to -1, and the values of row and column corresponding to t2 is equal to a/r*d(t2,i) + b/r*d(t1,i). In the next step, the names of objects t2 and t1 are concatenated and the resultant value is stored in t2. In the next iteration suppose the minimum is found in row, with row value t2, the new t1 is t2 and the number of objects in t1 is t2 and the number of objects in t1 is now 2, i.e. a=2. After n-1 iteration all become a single cluster.

IV. COMPUTATIONAL STUDY

For constructing dissimilarity matrix D, it requires O (n^3) computing time. Instead of computing dissimilarity matrix D, we can also find similarity matrix. In this case, find the maximum instead of minimum in the 4th step. To find the minimum in step 4, the algorithm considers only the lower triangular matrix. So it needs O (nc_2). Otherwise it requires O (n^2) of time. For rearranging the matrix elements in step 6 require O (n). Since the steps 4 to 7 are repeated utmost (n-1) times, the total computation time is utmost O (n^3). The following table (1) shows the computing time in seconds when 'n' varies.

Table 1: Computing time in seconds when 'n' varies.

N	10	50	100	200	300	500	1000
Computing Time (s)	0.549	0.549	0.604	0.659	0.824	0.879	1.373

V. NEW APPROACH BASED ON PST

The new algorithm is introduced to partition the object into group of clusters by using a new approach. The idea is: the algorithm starts with the choosing of two distantly related objects as a seed point. Let there be two clusters. Then it scans each object and for every object checks the distance between this object with the seeds. The object should be held into one of the clusters whose distance is minimum and it should be less than the calculated threshold [<= the average of all distances], and if otherwise marked that object as uncluster. The threshold is computed at the creation of each cluster [average distance metrics for all unclustered objects]. Then scan all unclustered objects and choose one object as a next seed which has the maximum distance for all existing clusters. This process should be continued until there is no more object to be clustered.

In a particular iteration, after selecting the new seed point, if no objects are grouped into that cluster, then that seed point can be detected as an outlier. That means this new seed point is very distantly related to all other objects. So the proposed algorithm is also used to detect the outliers.

VI. ALGORITHM

Input: set of n objects and its distance with other objects (distance matrix)

Output: Cluster of objects

Find Threshold T: Sum of distance value between each pair of unclustered objects (initially all objects are unclustered) in the distance matrix divided by count. [Average of distant metrics]

Cluster Formation: Let X be the set of all objects.

Step 1: Choose two objects from X which has the maximum distance in the distance matrix. These two objects are the seed points of two initial clusters.Delete these two objects from X.

Step 2: Uc = X //Uc= set of un clustered objects

For each object in X.

Search among the existing cluster seed points, with which if the distance is minimum and if it is also less than T, then place the object in the corresponding cluster [and remove this object from Uc], otherwise mark this object as uncluster [do not remove this object from Uc].

Step 3: For each object in Uc

Choose an object as a seed point of next cluster which is having maximum distance with more number of existing cluster seed points.

Remove that object from X.

Step 4: Repeat step 2 and step 3 until no more objects in Uc.

VII. TIME COMPLEXITY

Finding Threshold T takes O(n^2) at the maximum (if we consider only either upper or lower triangular matrix it will take only nc_2 time).

For cluster formation, step 1 again takes O(n^2) [or nc2] time in order to find the maximum distant value for initial seed point selection. Step 2 takes O(n x 2) for the first time and O(n x 3) for the second and so on, and finally it will take O(n x k), where n is the number of objects and k is the number of clusters that we will get. Step 3 takes O(p), where p is the number of unclustered objects in that time. Thus the total computing time is O(n^2)+O(nk)+O(p). In general the proposed algorithm takes O(n^2) at the maximum.

VIII. COMPARISON ANALYSIS

Unclustered objects are B and F.
Here F has the highest distance for both the seeds. So choose the next seed as F.

Table 2: F -The Highest Distance

E	D	F
d(A,E)	d(A,D)=5	d(A,F)=7
A	-	-
d(B,E)=7	d(B,D)=8	d(B,F)=1
-	-	B
d(C,E)=10	d(C,D)=5	d(C,F)=2
-	-	C
d(G,E)=4	d(G,D)=5	d(G,F)=2
-	-	G

Now the three clusters are:

{A,E} {D} {F,B,C,G}. -------------- (A)

Repeat this process until no more objects are marked as unclustered.

UPGMA clustering algorithm works as follows:-

Table 3: The Minimum Value -1

	A	B	C	D	E	F	G
A	0	2	3	5	4	7	6
B	2	0	9	8	7	1	4
C	3	9	0	5	10	2	3
D	5	8	5	0	12	11	5
E	4	7	10	12	0	9	4
F	7	**1**	2	11	9	0	2
G	6	4	3	5	4	2	0

Here minimum value is 1, so that corresponding objects B and F are closely related and hence that objects are merged into a cluster. Then the distance metrics are changed as per the formula.

Table 4: The Minimum Value - 3

	A	{B,F}	C	D	E	G
A	0	4.5	3	5	4	6
{B,F}	4.5	0	5.5	9.5	8	3
C	3	5.5	0	5	10	3
D	5	9.5	5	0	12	5
E	4	8	10	12	0	4
G	6	3	**3**	5	4	0

Now the minimum value is 3, so choose the corresponding objects C and G and merge these two objects. Then the distance metrics are changed as per the formula.

Table 5: The Minimum Value - 4

	A	{B,F}	C	D	E
A	0	4.5	4.5	5	4
{B,F}	4.5	0	4.25	9.5	8`
{C,G}	4.5	4.25	0	5	7
D	5	9.5	5	0	5
E	**4**	8	7	5	0

Now minimum value is 4, and that corresponding objects are A and E, so merge these two clusters. Then the distance metrics are changed as per the formula.

Table 6: The Minimum Value – 4.25

	{A,E}	{B,F}	{C,G}	D
{A,E}	0	6.25	5.75	5
{B,F}	6.25	0	4.25	9.5
{C,G}	5.75	4.25	0	5
D	5	9.5	5	0

Here the minimum value is 4.25. Now {C,G} and {B,F} clusters are merged.

Now it will become

{A,E} {B, C, G, F} {D} ----------- (B)

The UPGMA clustering algorithm proceeds in this way until all the objects belong to one cluster. When it is stopped here, the result (B) is the same as (A) which we got using the new algorithm.

IX. CONCLUSION

The new algorithm was introduced to get the partitioned clustering in hierarchical approach. Initially, two objects were chosen (based on their distance metrics or similarity) as seed points and two clusters were obtained. At each successive iterations, the correct object was chosen as a new seed point to increase the cluster quality so that, the number of clusters increased by one. The algorithm terminates if no more objects are to be clustered. Also, the result obtained from the proposed algorithm is proved with the most popular clustering algorithm called UPGMA. The UPGMA method required O (n^3) of computing time but the proposed algorithm required only O (n^2) of computing time. Also, the newly developed algorithm can be used to detect the outliers.

REFERENCE

[1]. Jain, A. K., Murthy, M. N. and Flynn, P. J. 1999. Data clustering-a review. *ACM computing Surveys*, 31: 264 – 323.

[2]. Mannila, H, Toivonen. H and Verkamo, A. I. 1997. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289.

[3]. Zaki, M. J. 1998. Efficient enumeration of frequent sequences. In. *Proc. 7th Int. Conf. Information and Knowledge Management (CIKM'98)*, pages 68–75, Washington, DC.

[4]. Garofalakis, M., Rastogi, R. and Shim, K. 1999. SPIRIT: Sequential pattern mining with regular expression constraints. In. *Proc. Int. Conf. Very Large Data Bases (VLDB'99)*. pages 223 – 234, Edinburgh, UK..

[5]. Pinto, H., Han, J., Pei, J., Wang, K., Chen, Q. and Dayal, U. 2001. Multi-dimensional sequential pattern mining. In *Proc. 2001 Int. Conf. Information and Knowledge Management (CIKM'01)*, pages 81–88, Atlanta, GA.

[6]. Pei, J., Han, J. and Wang, W. 2002. Constraint-based sequential pattern mining in large databases. In *Proc. 2002 Int. Conf. Information and Knowledge Management (CIKM'02)*, pages 18–25, McLean, VA.

[7]. Yang, J. and Wang, W. 2003. CLUSEQ: Efficient and effective sequence clustering. In *Proc. 2003 Int. Conf. Data Engineering (ICDE'03)*, pages 101–112, Bangalore, India.

[8]. Cheng, H., Yan, X. and Han, J. 2004. IncSpan: Incremental mining of sequential patterns in large database. In *Proc. 2004 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'04)*, pages 527–532, Seattle, WA.

[9]. Cheng, H., Yan, X. and Han, J. 2005. Seqindex: Indexing sequences by sequential pattern analysis. In *Proc. 2005 SIAM Int. Conf. Data Mining (SDM'05)*, pages 601–605, Newport Beach, CA.

[10]. Cong, S., Han, J. and Padua, D. 2005. Parallel mining of closed sequential patterns. In. *Proc. 2005 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'05)*, pages 562–567, Chicago, IL.

[11]. Nelson, M. R. 1996. Fast String Searching with Suffix Trees. *Dr. Dobb's Journal, Algorithm Alley*.

[12]. Gusfield, D. 1997. Algorithms on Strings, Trees, and Sequences-Computer Science and Computational Biology. *Cambridge University Press.*

[13]. Kurtz, S .1999. "Reducing the Space Requirement of Suffix Trees," *Software–Practice and Experience*, 29(13**)**: 1149–1171.

[14]. Yona, G and Bejerano. G 1999. Modeling protein families using probabilistic suffix trees. In. *Proceedings of the third annual international conference on Computational molecular biology*. Pages 15 – 24, ACM press, New York, NY, USA.

[15]. Bejerano, G. and Yona, G. 2001. Variations on probabilistic suffix trees: Statistical modeling and prediction of protein families. *Bioinformatics*, 17(1): 23–43.

[16]. Yang, J. and Wang, W. 2002. Towards Automatic Clustering of Protein Sequences. In. *Proceedings of the IEEE Computer Society Conference on Bioinformatics*. pages. 175–186. IEEE Computer Society, Washington DC, USA.

[17]. Sun, Z. and Deogun, J. S. 2004. Local Prediction Approach for Protein Classification using Probabilistic Suffix Trees. In. *Proceedings of the second conference on Asia-Pacific bioinformatics*. Pages 357 – 362. Australian Computer Society Inc., Darlinghurst.

[18]. Schürmann, K. B. and Stoye, J. 2006. Suffix Tree Construction and Storage with Limited Main Memory. Report 2003-06, Universitat Bielefeld 33501, Bielefeld, Germany.

[19]. Tata, S., Hankin, R. A. and Patel, J. M. 2004. Practical Suffix Tree Construction. In. *Proc. Int. Conf. Very Large Data Bases (VLDB'04)*. Pages 36 – 47, Toronto,Canada.

[20]. Mazeroff, G., Cerqueira,V. D., Michael Gregor, J. and Thomason, M. G. 2003. Probabilistic Trees and Automata for Application Behavior Modeling. In. *Proc. 43rd Asian South east Conf*., pages 435 440, Savannah, GA.

[21]. Ukkonen, E. 1995. On-line Construction of Suffix Trees. *Algorithmica*, 14: 249 – 260.

[22]. Mc Creight, E. 1976. A Space-economical Suffix Tree Construction Algorithm. *Journal of the ACM*, 23: 262–272.

[23]. Farach, M. 1997. Optimal suffix tree construction with large alphabets. In. *Proceedings of the 38th Annual Symposium on the Foundations of Computer Science (FOCS 97)*, pages 137-143