A Scalable Algorithm for Clustering Sequential Data *

Valerie Guralnik and George Karypis
University of Minnesota, Department of Computer Science and
Army HPC Research Center, Minneapolis, MN 55455
{guralnik,karypis}@cs.umn.edu

Abstract

In recent years, we have seen an enormous growth in the amount of available commercial and scientific data. Data from domains such as protein sequences, retail transactions, intrusion detection, and web-logs have an inherent sequential nature. Clustering of such data sets is useful for various purposes. For example, clustering of sequences from commercial data sets may help marketer identify different customer groups based upon their purchasing patterns. Grouping protein sequences that share similar structure helps in identifying sequences with similar functionality. Over the years, many methods have been developed for clustering objects according to their similarity. However these methods tend to have a computational complexity that is at least quadratic on the number of sequences. In this paper we present an entirely different approach to sequence clustering that does not require an all-against-all analysis and uses a near-linear complexity K-means based clustering algorithm. Our experiments using data sets derived from sequences of purchasing transactions and protein sequences show that this approach is scalable and leads to reasonably good clusters.

1 Introduction

In recent years, we have seen an enormous growth in the amount of available commercial and scientific data. Data from domains such as protein sequences, retail transactions, intrusion detection, and web-logs have an inherent sequential nature. Clustering of such data sets is useful for various purposes. For example, clustering of sequences from commercial data sets may help marketer identify different customer groups based upon their purchasing patterns. Grouping protein sequences that share similar structure helps in identifying sequences with similar functionality.

Over the years, many methods have been developed for clustering objects according to their similarity. These algorithms can be broadly classified into two categories: partitional and hierarchical. Partitional clustering algorithms, as typified by the K-medoid algorithm [9, 4], obtain clusters of objects by selecting cluster representatives and assigning each object to the cluster with its representative closest to the object. On the other hand, hierarchical clustering algorithms, such as UPGMA or single-link [4], produce a nested sequence of clusters, with a single all-inclusive cluster at the top and single point clusters at the bottom. These clustering algorithms can be easily adapted to cluster sequential data sets, provided that the pairwise similarity between the sequences can be easily computed. However these methods tend to have a computational complexity that is at least quadratic on the number of sequences, as they need to compute the pairwise similarity between all the sequences. Thus, they are only applicable to small data sets. Moreover, computationally efficient schemes such as K-means cannot be directly applied as it is hard to compute sequence centroids.

In this paper we present an entirely different approach to sequence clustering that does not require an all-against-all analysis and uses a near-linear complexity K-means based clustering algorithm. The key idea of our approach is to find a set of features that capture the sequential nature of the various data-sequences, project each data-sequence into a new space whose dimensions are these features, and then use a traditional K-means based clustering algorithm to find the clusters of the data-sequences. Our approach was inspired by research in document clustering that showed that high quality clusters can be obtained when each document is represented using a "bag of words". Clustering the documents based solely on their similarity with respect to these words generates clustering solutions which are equally good to methods that try to take into account phrase, paragraph, and document structure. In light of this example, our algorithm can be thought of as first discovering the "words" (*i.e.*, features) of the sequences, and then clustering the sequences based on the words that they have. Our experiments using data sets derived from sequences of purchasing transactions and protein sequences show that this approach is scalable and leads to reasonably good clusters.

2 Background

Clustering is the task of grouping together the objects into meaningful subclasses. We focus on clustering sequential data in which each object is represented as a sequence of set of items, called *itemsets*. Such sequence is called *data-*

*This work was supported by NSF CCR-9972519, EIA-9986042, ACI-9982274, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program, and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by the Minnesota Supercomputing Institute.

0-7695-1119-8/01 $17.00 © 2001 IEEE

```
a  x  a  b  -  c  s
a  x  -  b  a  c  s
```

Figure 1: Example of string alignment

```
(a b)   (b d)    ()   (c f)   ()
(b f)   (b d g)  (h)  (f k)   (l m)
```

Figure 2: Example of sequence alignment

sequence. For sequential data sets, the problem of clustering becomes one of finding the groups of data-sequences similar to each other.

2.1 Measuring Similarity between Sequences

One of the key steps in all clustering algorithms is the method used to compute the similarity between the objects being clustered. Over the years, a number of different approaches have been developed for computing similarity between two sequences of symbols (*i.e.*, strings) based on sequence alignment [5]. The idea behind these approaches is to align two strings against each other so that to maximize the similarity between the portions of the strings that fall at the same location of the alignment. Figure 1 shows an example of such an alignment between two particular protein sequences. These string-based optimal alignment approaches can be extended to compute the similarity between two sequences of itemsets as follows. Let S_1 and S_2 be two sequences containing m and n itemsets, respectively. Let $S_1(i)$ be the i^{th} itemset of S_1 and $S_2(j)$ be the j^{th} itemset of S_2. Furthermore, let S_1' and S_2' be two sequences of length l obtained after aligning S_1 against S_2, by inserting empty itemsets at either inside, at the beginning, or at the ends of the two sequences, so that every itemset (including empty) in either sequence is opposite a unique itemset in the other sequences. An example of this type of alignment is shown in Figure 2. The score of such alignment A can be defined as

$$score(A) = \sum_{i=1}^{l} sim(S_1'(i), S_2'(i)).$$

The similarity between two itemsets $S_1'(i)$ and $S_2'(i)$ can be measured in various ways. One way is to count the number of items that are common between the two itemsets and scale the count so that the similarity is always a number between 0 and 1, resulting in the following measure:

$$sim(S_1'(i), S_2'(i)) = \frac{|S_1'(i) \bigcap S_2'(i)|}{\frac{|S_1'(i)|+|S_2'(i)|}{2}}.$$

Another way of measuring similarity is to represent itemsets using the vector-space model. In this model, each itemset is considered to be a vector in the item space. In its simplest form, each itemset is represented by the vector $I = (i_1, i_2, \ldots, i_n)$, where i_j is an indicator whether the j^{th} item is in the itemset. Given this representation, the cosine similarity measure is a natural way of computing the similarity, and is defined as

$$sim(S_1'(i), S_2'(i)) = \frac{S_1'(i) \bullet S_2'(i)}{\|S_1'(i)\| \|S_2'(i)\|}.$$

Given any scoring scheme (including the ones introduced above), the *optimal alignment* A^* of two sequences S_1 and S_2 is defined as an alignment that maximizes the total alignment score $score(A^*)$. The score of the optimal alignment can be used as the similarity measure of two sequences. Depending on the application domain, one might want to scale this value so that the similarity between sequences of different lengths are comparable. The following formulas achieve the desired result:

$$sim(S_1, S_2) = \frac{score(A^*)}{\frac{|S_1|+|S_2|}{2}} \text{ or } sim(S_1, S_2) = \frac{score(A^*)}{l}.$$

The similarity of two sequences S_1 and S_2, and the associated optimal alignment, can be computed via dynamic programming [5] with a complexity $O(|S_1| * |S_2|)$.

2.2 Clustering Algorithms

Agglomerative hierarchical clustering and K-means are two techniques that are commonly used for clustering. Hierarchical techniques produce a nested sequence of partitions, with a single all-inclusive cluster at the top and singleton clusters of individual points at the bottom. Each intermediate level can be viewed as combining two clusters from the next lower level (or splitting a cluster from the next higher level). Agglomerative hierarchical algorithms start with all the data points as a separate cluster. Each step of the algorithm involves merging two clusters that are most similar. After each merge, the total number of clusters decreases by one. These steps can be repeated until the desired number of clusters is obtained or the distance between two the closest clusters is above a certain threshold distance.

In contrast to hierarchical techniques, partitional clustering techniques create a one-level (un-nested) partitioning of the data points. Partitional clustering attempts to break a data set into K clusters such that the partition optimizes a given criterion. Centroid-based approaches, as typified by K-means try to assign objects to clusters such that the mean square distance of objects to the centroid of the assigned cluster is minimized. Centroid-based techniques are suitable only for data in metric spaces (e.g. Euclidean space) in which it is possible to compute centroid for a given set of points. Because it is computationally hard to compute centroids in the space of data-sequences, medoid-based approaches are better suited for clustering sequential data sets. Medoid-based methods work with similarity data, *i.e.*, data in arbitrary similarity space. These techniques try to find representative points (medoids) so as to minimize the sum of the distances of points from their closest medoid. It has been

shown, that if the measure used to compute similarity satisfies the *triangle inequality*, then in each cluster of n data-sequences there exists a medoid S_m, such that $M = \sum_{i=1}^{n} score(S_m, S_i)$ is never less than $2 - 2/n$ times the $M_c = \sum_{i=1}^{n} score(S_c, S_i)$, where S_c is the centroid of the cluster [5].

2.3 Limitation of existing approaches

One limitation of using both hierarchical and medoid-based partitional clustering approaches is that when the dynamic programming algorithms are used to compute the similarity, their complexity is $O(n^2m^2 + n^2 \log n)$ and $O(n^2m^2 + ntk)$, respectively; where n is the number of data-sequences, m is the average length of each data-sequence, k is number of clusters and t number of iterations in the medoid-based approach. These high computational requirements make such approaches impractical for most applications that require clustering of moderate and large data sets.

3 Feature-based Clustering

The high computational requirements of both the hierarchical clustering algorithms and K-medoid approaches are due to the fact that (a) they need to compute the pairwise similarity between all the data-sequences and (b) the similarity computations have a complexity that is quadratic to the length of the data-sequences involved. To address these high computational requirements, we explore an alternate approach for clustering sequences that (i) does not use dynamic programming to compute the similarity, and (ii) it uses a K-means algorithm whose complexity is near-linear in the number of sequences.

The key idea of our approach is to find a set of features that capture the sequential nature of the various data-sequences, project each data-sequence into a new space whose dimensions are these features, and then use a traditional vector-space K-means clustering algorithm [14] to find the clusters of the data-sequences in this transformed space.

In the remaining of this section we describe the various algorithms and issues associated with each one of these three steps.

3.1 Finding the Feature Space

An essential part of the proposed approach is finding the set of features that will form the basis of the transformed space. In particular, these features must satisfy the following properties:

1. The features should capture the sequential relations between the different itemsets that are present in the data-sequences. This is particularly important, since the proposed clustering algorithm will cluster the data-sequences based solely on their similarity with respect to these features.
2. The features should be present in a nontrivial number of data-sequences. This is because rare features do not improve the overall clustering, as they are useful only in defining affinity between a small set of data-sequences.
3. The feature space should be complete, in the sense that all such interesting features should be contained in the transformed space.

Our algorithm achieves these goals by using as features all the sequential patterns whose length is between l_{min} and l_{max} and satisfy a given minimum support constraint. A sequential pattern is a list itemsets with the support above a user-specified threshold, where the support of the pattern is the percentage of data-sequences that contain it. Given a sequential pattern $< s_1, s_2, \ldots, s_n >$, where s_i is an itemset, the length of the pattern is the number of items in all itemsets s_i of the pattern. The gap between itemsets i_1 and i_2 of the data-sequence supporting a particular pattern is defined $occurrence(i_1) - occurrence(i_2)$, where definition of the occurrence is domain specific. Depending on the application domain, one might impose minimum/maximum gap constraints on sequential patterns. These frequent sequential patterns, can be computed efficient using a variety of sequential pattern discovery algorithms [1, 13, 15, 8, 6].

3.2 Projecting in to the Feature Space

The critical step in our approach is that of representing each data-sequence in the newly discovered space of sequential features. If N is the dimensionality of the feature space, a straightforward way of achieving this is to represent each data-sequence as an N-dimensional vector of zeros and ones, with ones corresponding to all the features that are supported by that particular data-sequence.

Unfortunately, this representation can potentially lead to poor clustering results. This is because, the different features that are supported by a particular sequence may be highly dependent which can substantially distort the similarity measure that is used in the transformed space. For instance, if a particular sequential pattern w of length l, with $l > l_{min}$ is supported by a particular sequence, then all of its sub-patterns of length greater than l_{min} will also be supported as well. As a result, when we compare two sequences that both have w, their similarity will be distorted by the different sub-patterns of w that they also share. Similar problem occurs when two sequential patterns partially overlap as well. For example, consider the following scenario in context of protein clustering. Let's assume that we have the database of amino-acid sequences shown in Figure 3 together with all sequential patterns of consecutive amino-acids of lengths 3 and 4 that have support of 50%. Let's concentrate on the first two sequences S_1 and S_2 and the two discovered patterns AQVH and HKKS. Saying that both proteins subscribe to both patterns will mean that there are two similarity regions of length 4 between them, while if we computed the alignment of those proteins we would find that there is only one region of length 4 where both proteins align (either AQVH

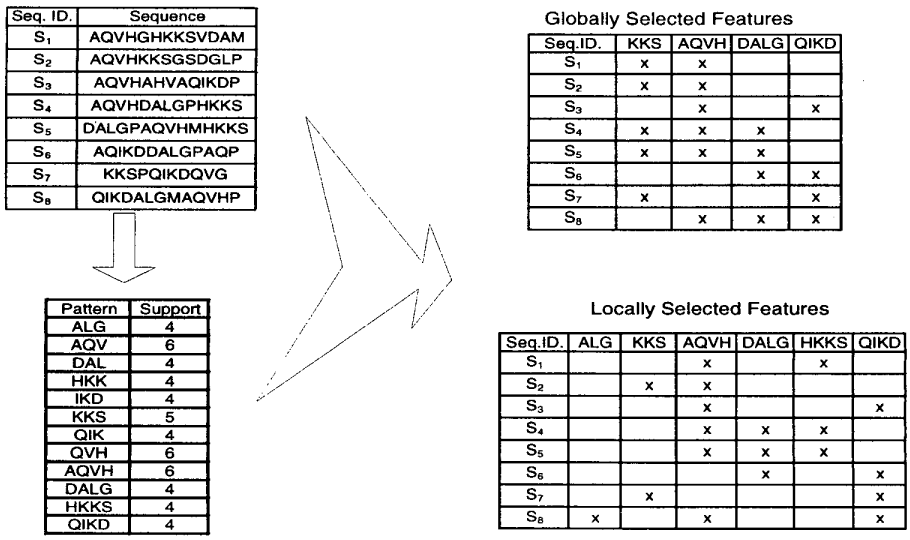


Figure 3: Feature Selection Example

or HKKS). Therefore, it is important to represent each sequence in a way such that the dimensions that they are using are as independent of each other as possible. We implemented two different approaches to address this problem, that are described in the rest of this section.

3.2.1 Global Approach

One way of addressing the above problem is to prune the feature space by selecting only a set of independent features prior to projection. In particular, we say that two sequential patterns are *dependent* if and only if (i) either one is the prefix of the other or one is a sub-pattern of the other, and (ii) the intersection of their respective supporting sets is non-trivial.

These conditions essentially call two patterns that draw support from the same region of the sequence to be dependent. Coming back to the example from Figure 3, let's assume that the intersection of two patterns supporting set is non-trivial if its cardinality is at least two thirds of smallest support of the pattern. Under this condition one possible set of independent patterns is KKS, AQVH, DALG, QIKD, as shown in Figure 3.

Using the definition of independence, we can then use a greedy algorithm to select a maximal set of independent features and restrict the space to only this set of features. Even though this approach ensures that the set of features that we select are independent, it has a number of potentially serious drawbacks. First, computation of the pairwise intersection of the supporting sets for each sequential pattern is computationally expensive. Second, the resulting space will either be over-pruned or under-pruned. Thus in our example, patterns AQVH and HKKS are found dependent (the number of proteins that support both of them is 4). As a result all the sequences supporting both of these patterns subscribe to only AQVH. However, almost all of the sequences that support both patterns have two regions of similarity of length 4. Hence, we are presented with an over-pruned space. Ideally we would like for S_1 to subscribe to both patterns, and for S_2 to subscribe to only one of them. On the other hand, patterns DALG and QIKD are found independent (the number of proteins that support both of them is 2). As a result the sequences S_6 and S_8 have two regions of similarity of length 4 QIKD and DALG which is not correct. As we can see, over-pruning of the space contradicts the required property of completeness. Under-pruning doesn't solve the problem of having redundant features.

3.2.2 Local Approach

In order to correct the problem of the global approach, we developed a method for selecting a set of independent features that is done locally, on a per data-sequence basis. In this approach, for each data-sequence we first find the set of features that it supports, and from this set we select a maximal set of independent features. In this context, two features are considered to be *independent*, if they are supported by non-overlapping segments of the underlying data-sequence. The advantage of this approach is that it allows us to subscribe each data-sequence to as many independent features as possible (regardless of the features selected by other data-sequences), and at the same time, the process of feature selection is very fast.

One potential problem with this approach is that sequences that share a large number of sequential patterns, may actually end up having low similarity, because the independent sets they selected, had little overlap. Two address problem we select the locally independent features using the same greedy strategy, so that we increase the likelihood that if two data-sequences share a number of sequential patterns, then a considerable number of them will be selected by both of them—ensuring that if two data-sequences are similar in the original space, will also be

similar in the transformed space. This can be done in a number of ways. One way to select a feature out of set of dependent patterns is to select a more frequent pattern, or pattern that has more items. An example of locally selected features is presented in Figure 3, in which the selection strategy gave preference to the longer pattern.

3.3 Clustering in the Feature Space

Once the data-sequences have been projected into the feature space, we use an efficient vector-space clustering algorithm based on K-means [14] to find k clusters. In this algorithm, each data-sequence is represented by a vector in the feature-space, and the similarity between two data-sequences is computed using the cosine similarity function, commonly used in the context of information retrieval [10]. Moreover, in some domains it is important to account for frequently occurring low complexity sequential patterns. To do this, we scale each of the features following the *inverse-document-frequency* methodology, again inspired by research in information retrieval. In this approach, if a particular feature appears in m out of n data-sequences, its weight is multiplied by $\log(n/m)$. The effect of this scaling is that infrequently occurring features are given higher weight that features that occur in almost every data-sequence.

4 Experimental Evaluation

We experimentally evaluated our approach using datasets arising in two domains: retail and bioinformatics. All experiments were run on a Linux machine with 4GB of memory utilizing 550 MHz Pentium III CPU.

4.1 Evaluation of Cluster Quality

For clustering, two measures of cluster "goodness" or quality are used. One type of measure allows us to compare different sets of clusters without reference to external knowledge and is called an *internal quality* measure. One internal measure is weighted average similarity, which is based on the pairwise similarity of sequences in each cluster. The weighted average similarity is calculated as follows. Let CS be a clustering solution. For each cluster C_j, we first compute its average similarity

$$AS_j = \frac{\sum_{S \in C_j, S' \in C_j} sim(S, S')}{n_j(n_j - 1)},$$

where n_j is number of sequences in cluster C_j. The weighted average similarity for a set of clusters is calculated as the sum of the average similarities for each cluster weighed by the size of each cluster:

$$WAS_{cs} = \sum_{j=1}^{m} n_j * AS_j,$$

where n_j is the size of cluster C_j, and m is the number of clusters.

The other type of measures lets us evaluate how well the clustering is working by comparing the groups produced by the clustering techniques to known classes. This type of measure is called an *external quality* measure. One external measure is the *entropy* [12], that is calculated as follows. Let CS be a clustering solution. For each cluster C_j, we first compute the distribution of the data-sequences that it contains for each class i, *i.e.*, p_{ij} is equal to the probability a randomly drawn data-sequence from cluster C_j to be of class i. Then using this class distribution, the entropy of each cluster C_j is calculated using the formula

$$E_j = -\sum_i p_{ij} log(p_{ij}).$$

The total entropy for a set of clusters is calculated as the sum of the entropies for each cluster weighted by the size of each cluster:

$$E_{cs} = \sum_{j=1}^{m} \frac{n_j * E_j}{n},$$

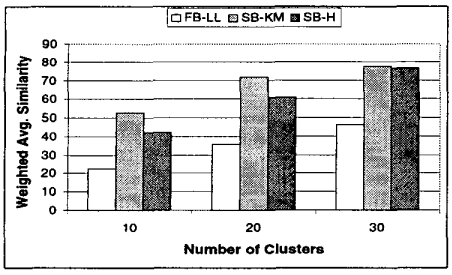
where n_j is the size of cluster j, m is the number of clusters, and n is the total number of data-sequences in that data set. Note, that the entropy value 0 indicates a perfect clustering solution. The higher the entropy value the worse the clustering solution is.

4.2 Retail Data Set

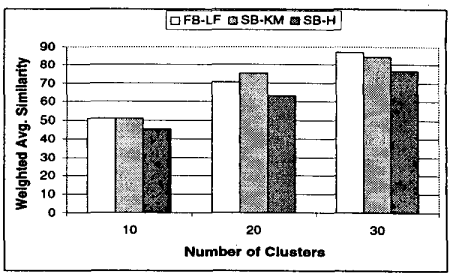
The retail data set contained a history of store-branded credit-card purchases of 7451 customers of a major department store, such that each customer made 3 or more purchases. The total number of distinct products purchased was 222348. For this data set we found 2435 frequent sequential patterns of length 2 or more with minimum support equal to 0.1%. The maximum length of the pattern that was discovered was 9.

To subscribe data-sequences to discovered patterns we used both global and local methods with different feature selection approaches, namely selecting a longer pattern or a more frequent pattern, resulting in four test sets FB-GL (global selection of longer patterns), FB-GF (global selection of more frequent patterns), FB-LL (local selection of longer patterns) and FB-LF (local selection of more frequent patterns). After the independent patterns were selected, the FB-GL approach kept 255 patterns and subscribed 2061 data-sequences, the FB-GF approach selected 241 patterns and subscribed 2552 sequences, the FB-LL approach kept 707 frequent patterns and subscribed 3164 data-sequences, and the FB-LF kept 546 patterns and subscribed 3230 data-sequences. Note that schemes that give preference to the more frequent patterns resulted in spaces with fewer dimensions as frequent patterns are inherently more dependent. The sequences that didn't support any of the frequent patterns were not used for clustering.

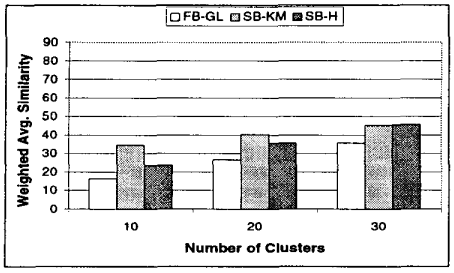
The resulting clustering solutions were compared against solutions produced by similarity-based approaches



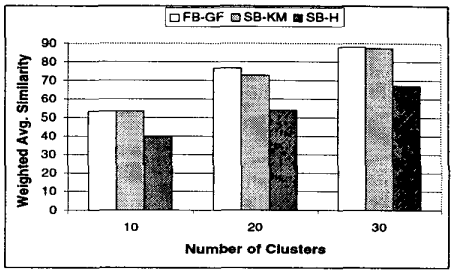
FB-LL vs. SB-KM and SB-H



FB-LF vs. SB-KM and SB-H



FB-GL vs. SB-KM and SB-H



FB-GF vs. SB-KM and SB-H

Figure 4: Comparison of Feature Based Clustering vs. Similarity Based Clustering

– hierarchical algorithm which optimizes group average similarity [4] (SB-H) and K-medoid (SB-KM). To ensure that the comparison were performed in an unbias way, only the data-sequences that could be projected on the feature space were clustered. This resulted in four different sets of experiments, one for each of the feature selection strategies. In the absence of class information, we used the weighted average similarity of the clusters in the sequence space, as a measure of quality of the clustering solution.

Figure 4 shows the weighted average similarity of 10, 20 and 30-way clustering solutions generated by the different algorithms. Note that high values of weighted average similarity represent better clustering solutions. From this figure it can be seen that both global and local approaches, which selected longer patterns, performed poorly. In analyzing the reason for this behavior, we discovered that the data-sequences in this data set are short and therefore supported only a small number of sequential patterns. As a result, by preferring longer sequential patterns the majority of data-sequences only subscribed to a small number of dimensions (usually one or two). Thus, if one sequence contained a long pattern and another contained its sub-pattern, those sequences mostly likely ended up in different clusters due to the fact that they contained different features. This resulted in un-similar data-sequences getting clustered in the same group. After examining frequent dimensions of the resulting clusters, we found for example that customers who bought home collection items were put in the same cluster as customers who bought hair-care products.

To overcome this problem, we ran the experiments FB-LF and FB-GF in which more frequent patterns were selected. In both cases the feature based approach outperformed the hierarchical algorithm, and showed comparable performance to K-medoid. Comparing the global selection methods against those that select features that are locally independent in each data-sequence, we can see that the latter approach performs considerably better. Note, that as it was described in Section 3.2 the resulting global schemes became over-pruned. This is evident by cardinality of the transformed space in the global selection scheme which is about 3 times smaller. As a result global schemes were not able to cluster as many data-sequences as local ones.

Even though the feature-based approach didn't show significant improvement over similarity based algorithms, the proposed approach has number of advantages. First, by projecting only the data-sequences that support frequent patterns onto the feature space, our approach eliminates data-sequences that are outliers. This is because the sequences that do not contain frequent patterns are not similar to a lot of other sequences in the data set and thus are not relevant for clustering. Second, examining the dimensions which occur frequently in each cluster helps us to gain insight about its characteristics and thus interpret the clustering solution. The medoids of the K-medoid approach can serve as representatives of the clusters. However, since it is unknown what regions of the medoid sequence occur frequently in the cluster and what regions are unique to this particular medoid, it will be hard to use this sequence to describe the cluster. Examples of clusters found by our approach are group of customers who buy home collection products and group of people who buy clothes for teenagers.

4.3 Data Sets of Proteins

As another way to evaluate the performance of the proposed clustering algorithm we generated three different data sets, DS1, DS2, and DS3, containing protein se-

No. of Seqs	Clust. Sim.	Functional Classes F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
232	0.69	232	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
229	0.66	0	228	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
207	0.45	0	0	207	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
220	0.62	0	0	0	220	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
211	0.5	0	0	0	0	211	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
196	0.43	0	0	0	0	0	196	0	0	0	0	0	0	0	0	0	0	0	0	0	0
191	0.48	0	0	0	0	0	0	191	0	0	0	0	0	0	0	0	0	0	0	0	0
185	0.34	0	0	0	0	0	0	0	185	0	0	0	0	0	0	0	0	0	0	0	0
200	0.25	2	0	0	0	0	17	0	2	178	1	0	0	0	0	0	0	0	0	0	0
171	0.38	0	0	0	0	0	0	0	0	0	163	8	0	0	0	0	0	0	0	0	0
154	0.51	0	0	0	0	0	0	0	0	0	0	153	0	0	0	1	0	0	0	0	0
180	0.31	0	3	0	0	1	0	1	0	0	37	138	0	0	0	0	0	0	0	0	0
183	0.5	0	0	0	0	0	1	0	0	0	43	0	104	0	0	35	0	0	0	0	0
122	0.41	0	0	0	0	9	0	0	0	0	0	0	0	111	0	0	2	0	0	0	0
181	0.27	0	1	0	0	0	0	0	0	0	0	0	0	96	0	0	77	0	0	2	5
83	0.48	0	0	0	0	0	0	0	0	0	0	0	0	0	83	0	0	0	0	0	0
56	0.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	0	0	0	0
128	0.38	0	0	0	0	2	0	1	0	0	0	0	0	0	77	48	0	0	0	0	0
770	0.14	0	3	2	1	15	5	27	14	13	12	26	0	14	68	56	56	96	96	142	124
876	0.13	0	1	23	16	12	16	17	33	47	18	22	1	12	84	25	51	146	143	106	103

Feature-based clustering solution

No. of Seqs	Clust. Sim.	Functional Classes F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
83	0.248	53	3	0	2	2	1	5	3	4	1	2	0	0	1	0	2	0	0	3	1
521	0.212	101	230	15	0	2	6	0	21	14	68	0	0	3	1	3	12	5	20	3	17
143	0.254	0	0	110	0	0	10	0	2	2	0	0	1	0	3	0	5	1	0	7	2
124	0.218	10	0	90	0	1	0	1	1	5	0	0	0	0	3	0	5	3	0	1	4
262	0.301	1	1	0	232	3	1	1	10	1	3	1	1	0	1	0	2	0	4	0	0
294	0.24	0	0	10	0	223	6	0	3	10	19	6	0	0	0	1	4	0	1	1	10
515	0.191	42	0	2	0	2	80	222	6	18	25	5	1	27	6	3	36	8	22	1	7
188	0.192	0	0	0	0	0	0	1	116	4	0	1	0	0	1	1	5	25	16	18	0
52	0.241	0	0	0	0	0	2	0	48	0	0	0	0	0	0	0	0	2	0	0	0
175	0.212	0	0	0	0	0	21	0	0	150	0	0	0	0	1	0	0	0	1	2	0
151	0.21	0	0	0	0	0	0	0	0	0	96	0	0	1	2	4	7	3	8	5	25
416	0.2	0	0	0	0	0	8	9	2	5	6	6	215	7	3	8	0	35	19	51	23	19
302	0.223	9	0	0	0	0	0	31	0	0	0	0	0	230	1	5	0	11	2	10	3	0
314	0.191	18	2	4	3	9	13	3	5	6	15	14	3	198	5	4	5	0	2	2	3
326	0.192	0	0	0	0	0	0	0	1	1	1	0	0	0	172	10	22	18	32	36	33
41	0.302	0	0	1	0	0	2	0	1	0	0	0	0	0	0	37	0	0	0	0	0
254	0.191	0	0	0	0	0	33	1	1	4	3	1	0	0	7	96	22	15	22	25	24
263	0.188	0	0	0	0	0	1	0	5	4	0	0	0	0	4	86	15	23	28	59	38
203	0.183	0	0	0	0	0	19	1	1	9	0	1	0	0	10	0	26	97	11	21	7
148	0.182	0	0	0	0	0	0	0	6	0	0	0	0	0	5	4	19	21	11	40	42

K-medoid clustering solution

Figure 5: Clustering solution for DS1

No. of Seqs	Clust. Sim.	Functional Classes F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
266	0.45	266	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
243	0.69	0	243	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
236	0.47	0	0	236	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
229	0.6	0	0	0	229	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
225	0.82	0	0	0	0	225	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
227	0.41	0	0	0	0	0	227	0	0	0	0	0	0	0	0	0	0	0	0	0	0
207	0.48	0	0	0	0	0	0	207	0	0	0	0	0	0	0	0	0	0	0	0	0
246	0.57	0	0	0	0	0	0	0	197	0	0	0	0	0	49	0	0	0	0	0	0
93	0.49	0	0	0	0	0	0	0	0	59	7	2	0	1	0	0	0	0	24	0	0
190	0.49	0	0	0	0	0	0	0	0	0	190	0	0	0	0	0	0	0	0	0	0
184	0.56	0	0	0	0	0	0	0	0	0	0	184	0	0	0	0	0	0	0	0	0
175	0.3	0	0	0	0	0	0	0	0	0	0	0	174	0	0	0	0	0	0	1	0
193	0.64	0	0	0	0	0	0	0	0	0	0	0	0	193	0	0	0	0	0	0	0
156	0.58	0	0	0	0	0	0	0	0	0	0	0	0	0	156	0	0	0	0	0	0
169	0.24	0	0	0	6	0	2	1	0	0	9	0	4	2	111	27	5	2	0	0	0
85	0.65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85	0	0	0	0	0
115	0.32	0	26	0	0	0	0	0	0	0	0	0	0	0	82	2	4	0	1	0	
132	0.31	0	0	0	0	24	15	0	0	1	0	0	0	2	0	18	62	10	0	0	0
646	0.14	0	0	3	24	1	3	15	0	15	16	33	22	33	19	41	94	55	139	74	59
1271	0.11	2	3	31	7	3	26	40	0	50	59	50	52	77	56	54	91	162	119	177	212

Feature-based clustering solution

No. of Seqs	Clust. Sim.	Functional Classes F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
205	0.29	194	0	0	0	0	0	0	0	0	0	2	0	0	0	2	6	0	1	0	0
267	0.29	0	206	0	0	0	0	0	8	0	0	0	3	0	0	0	4	7	0	39	0
267	0.255	2	2	237	0	0	0	0	0	1	1	0	0	14	2	2	4	2	0	0	0
261	0.233	0	0	0	142	0	2	7	1	0	4	40	0	1	4	5	12	16	19	2	6
400	0.194	1	5	13	102	2	0	1	1	76	19	11	3	0	6	10	3	19	12	109	7
269	0.352	0	0	0	0	232	0	0	0	0	2	2	0	0	6	0	7	12	5	0	3
175	0.22	4	0	0	0	0	118	0	0	39	1	2	0	1	2	1	1	2	1	2	1
177	0.214	11	6	0	0	0	128	0	0	25	0	0	3	0	0	0	2	0	1	0	1
487	0.19	9	38	1	0	0	1	206	9	82	6	28	2	1	3	3	10	5	5	22	56
201	0.215	6	5	3	21	9	14	5	107	4	3	1	4	0	3	2	4	4	6	0	0
216	0.211	30	3	15	0	8	6	4	104	9	9	1	2	2	1	10	4	4	1	3	0
546	0.192	0	0	0	0	0	0	10	0	5	196	45	5	2	54	18	34	32	10	41	94
458	0.217	0	0	0	0	1	2	4	3	2	5	11	219	0	115	6	13	41	10	16	10
405	0.199	1	4	1	1	0	1	1	26	15	1	31	3	244	2	9	17	1	0	14	33
320	0.181	4	0	0	0	0	1	4	0	0	22	25	5	5	41	13	30	89	17	25	39
147	0.194	0	0	0	0	0	0	0	4	0	0	8	0	0	20	68	38	0	6	0	3
147	0.21	6	3	0	0	1	0	4	0	5	0	11	0	0	10	62	2	19	0	18	6
111	0.204	0	0	0	0	0	0	0	0	0	1	8	1	0	0	49	40	1	9	0	2
81	0.189	0	0	0	0	0	0	6	1	0	0	18	25	0	2	1	14	6	7	0	1
148	0.194	0	0	0	0	0	0	3	0	0	0	10	0	0	0	6	6	4	110	0	9

K-medoid clustering solution

Figure 6: Clustering solution for DS2

quences from the SWISS-PROT [2] public protein sequence database. Each one of the data sets contains proteins from 20 different protein families. DS1 contains 4,775 sequences, DS2 contains 5,288, and DS3 contains 43,569 sequences. For each of the data sets, we found frequent patterns of consecutive amino-acids, of length 3 through length 6. The minimum support used for each data set was equal to 25% of the size of the smallest class. In all of our experiments, we used the local scheme for selecting independent dimensions during projection, and these dimensions were selected by giving preference to the longest patterns. We also experimented with the global selection scheme, but the quality of the resulting solutions was quite poor. For this reason we do not report these experiments in this paper.

We evaluated the quality of the resulting clustering solution using the entropy measure.

Figure 5, 6, and 7 show the 20-way clustering solution produced by our algorithm on the DS1, DS2, and DS3 data sets, respectively. For DS1, a total of 13,331 frequent patterns of length 3–6 were discovered, out of which 11,780 were kept after independent patterns were selected locally. In the case of DS2, the initial and final number of patterns

No. of Seqs	Clust. Sim.	Functional Classes F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20
2466	0.33	2463	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2169	0.15	11	1578	0	0	15	5	193	2	0	1	3	1	267	0	4	2	0	1	80	6
3581	0.67	0	0	3581	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
582	0.67	0	0	581	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1291	0.52	0	0	0	1291	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1522	0.53	0	0	0	0	1522	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1573	0.37	0	0	1	0	0	1240	0	0	0	0	0	0	332	0	0	0	0	0	0	0
1159	0.35	0	0	1	0	0	0	1157	0	0	0	0	0	0	0	0	0	0	0	0	0
1773	0.43	0	0	0	0	0	0	0	1773	0	0	0	0	0	0	0	0	0	0	0	0
1712	0.39	0	0	0	0	4	0	0	0	1708	0	0	0	0	0	0	0	0	0	0	0
1219	0.2	0	0	0	0	0	0	1	72	0	1123	1	1	0	0	0	1	7	10	1	8
718	0.49	0	0	0	0	0	0	0	0	0	0	714	0	0	1	1	0	1	0	1	0
1005	0.25	0	0	0	0	0	0	0	3	0	91	882	0	0	0	2	26	0	0	0	1
1708	0.22	0	0	0	0	0	19	4	1	0	2	0	1182	479	1	0	0	0	0	20	0
802	0.54	0	0	0	0	0	0	1	0	0	0	0	0	799	0	0	2	0	0	0	0
1050	0.29	0	0	0	0	320	0	2	0	40	0	0	2	506	0	0	0	0	178	0	2
1129	0.22	177	0	0	0	0	0	20	0	243	1	0	2	1	676	0	1	0	0	1	7
2916	0.13	8	4	0	0	11	1	46	184	56	7	229	21	30	2	1595	283	8	400	25	6
534	0.68	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	533	0	0	0	0
14660	0.12	84	721	21	125	702	66	68	66	159	3161	650	1055	211	958	665	676	1905	956	1080	1331

Figure 7: Feature-based clustering solution for DS3

were 19,129 and 14,139, respectively, and in the case of DS3 they were 22,672 and 21,223. Also, each sequence subscribed to an average of 71, 76, and 81 features for DS1, DS2, and DS3, respectively. The first two columns of each table show the number of proteins assigned to each cluster, and the average pairwise protein similarity between the proteins in each cluster, respectively. For each of the clusters, the remaining 20 columns of each row show the class distribution of the proteins that were assigned to that particular cluster.

Data Set	Feature-Based K-means	K-medoid
DS1	1.43	2.12
DS2	1.51	2.19

Table 1: Comparison of Entropy Measure

Looking at the various clustering solutions we can see that the proposed algorithm was able to produce clusters that primarily contained proteins from either one or two protein families. Furthermore, 14 functional classes are clearly distinguishable in both DS1 and DS2, and 13 are distinguishable in DS3. The members of the remaining functional classes were also mostly kept together, however they were clustered together with members of other functional classes. The overall quality of the clustering solution produced by our algorithm, as measured by entropy, was 1.43, 1.51, and 1.67, for DS1, DS2, and DS3, respectively.

A common characteristic of the clustering solutions for all three data sets was the fact that one or two of the clusters tend to be somewhat larger than the rest, and were both *loose* (as measured by the average pairwise similarity) and contained proteins from different families. In analyzing the reason for this behavior, we discovered that the proteins that were in these clusters contained patterns that were of length either 3 or 4, indicating that the proteins in them did not share some of the longer conserved patterns that the rest of the proteins did. One way of addressing this limitation of our approach is to use amino-acid substitution matrices or amino-acid similarity matrices to define equivalent classes of patterns [3, 11, 7].

The entropy measure of the clustering solution generated by our approach was compared against the entropy measure of clustering solution generated by K-medoid algorithm. Only two data sets DS1 and DS2 were used in this comparison, due to the need to compute all-against-all similarity matrix for each of the data sets. The computation of the matrix for each DS1 and DS2 took over three days. Because data set DS3 contained roughly ten times more data-sequences than either DS1 and DS2, the computation of the similarity matrix for this data set would have taken a prohibitively large amount of time.

Table 1 shows the comparison of entropy results for both data sets. From this table it can be observed that our algorithm outperformed the K-medoid. Figure 5 and 6 compares the 20-way clustering solutions produced by our approach and K-medoid algorithm on DS1 and DS2 respectively . A common characteristic of those clustering solutions is that the groups of proteins that could not be correctly clustered by our approach also did not cluster well by K-medoid. In addition, the functional classes which were clearly distinguishable in the feature based clustering solution were not clustered as well by K-medoid approach.

5 Conclusion

In this paper we presented a new approach to sequence clustering that uses a near-linear complexity K-means based clustering algorithm. Our approach is based on projecting the data-sequences onto space of frequent sequential patterns and using K-means based clustering algorithm to find clusters in that space. Our experimental evaluation in two domains shows that this approach appears promising and leads to reasonably good clusters. In addition, the feature based approach achieves comparable or better accuracy than similarity-based approaches.

References

[1] R. Aggrawal and R. Srikant. Mining sequential patterns. In *Proc. of the Int'l Conference on Data Engineering (ICDE)*, Taipei, Taiwan, 1996.

[2] A. Bairoch and B. Boeckmann. The swiss-prot protein sequence data bank. *Nucleic Acids Research*, 19:2247–2249, 1991.

[3] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5:345–352, 1978.

[4] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[5] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences*. Press Syndicate of the University of Cambridge, New Your, NY, 1997.

[6] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.C. Hsu. Freespan: Frequent pattern-projected sequential pattern mining. In *Proc. 2000 Intl. Conference on KDD*, 2000.

[7] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Academy Science*, 89(10):915–919, 1992.

[8] Mahesh V. Joshi, George Karypis, and Vipin Kumar. Universal formulation of sequential patterns. Technical report, University of Minnesota, Department of Computer Science, Minneapolis, 1999.

[9] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[10] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[11] R. M. Schwartz and M. O. Dayhoff. Matrices for detecting distant relantionships. *Atlas of Protein Sequences*, pages 353–358, 1979.

[12] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 1948.

[13] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of the Fifth Int'l Conference on Extending Database Technology*, Avignon, France, 1996.

[14] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[15] M.J. Zaki. Efficient enumeration of frequent sequences. In *7th International Conference on Information and Knowledge Management*, 1998.