Experimental Study of a Similarity Measure for Two Dimensional Sequences

Şule Gündüz, Eşref Adalı

Department of Computer Engineering,

Istanbul Technical University

Istanbul, 34469, Turkey

gunduz@cs.itu.edu.tr , adali@ cs.itu.edu.tr

ABSTRACT

In data mining and knowledge discovery, similarity between objects is one of the central concepts. A measure of similarity can be user-defined, but an important problem is defining similarity on the basis of data. In this paper we introduce the problem of finding the pair-wise similarities of quantitative valued sequences where each sequence is a list of items. Traditional approaches for defining the similarity between two sequences typically consider only the binary values of items in sequences, not the quantitative values. Such similarity measure is often useful for finding the similarities between genes or protein sequences. However, they cannot reflect certain kinds of similarity where the sequences contain two different kinds of information type, such as quantitative and order information. However, such type of sequence data arise in many applications, for example, marketing and sales data or web log data may contain two different kinds of information. Therefore, we introduce a new similarity measure that takes into account the values of items in sequences. We give an algorithm for calculating the similarity between such quantitative sequences. Finally, we describe the results of using this approach on two different real-life datasets.

Keywords : pair-wise similarity, sequence similarity, data mining

1. Introduction

The rapid development of computer technology in last decades has made it possible for data systems to collect huge amounts of data. Analyzing such large datasets is tedious and costly, and thus, we need efficient methods to be able to understand how the data was generated, and what sort of patterns and regularities there exists in the data. A research area in computer science that considers this kind of questions is called data mining [1].

An important form of data considered in data mining is sequential data. This kind of data occurs in many applications domains, such as biostatistics, medicine, telecommunication, user interface studies, market basket data, and WWW page request monitoring. Abstractly, such data can be viewed as a sequence of events where each event is associated time of occurrence.

A typical dataset considered as sequence database consists of a number of data objects with several attributes in order. An example of such a dataset is market basket data where data objects present customers and attributes are different products sold in the supermarket. Each sequence in this data set is a list of products bought by a customer over period of time. Another example could be user access data for web sites where the objects are users and the attributes are pages on the web site. Each sequence in such a dataset consists of a set of pages that a user requests during his or her single visit to a web site. It is clear that the attributes in such datasets have usually a large value domain. For example, for market basket data, the domain of attributes could be integers where each number is the quantity of a product bought by a customer. Similarly, the values of attributes for user access dataset of a web site could be the time that users spend on each page.

In order to find patterns and regularities in the data, it is necessary to be able to describe how far from each other two data objects are. This is the reason why similarity between objects is one of the central concepts in data mining. When

discussing similarity and databases, one often talks about the similarity between data objects stored in the database. Analyzing similarities between sequences gives us an important knowledge about the behavior and actions of a system or a user which can be used in forming hierarchies or clusters of data objects. Such a hierarchy describes the structure of the data and can be used for forecasting. There are some proposes for finding similarities of sequences where the value of attributes are binary corresponding the existence of an attribute in the sequence [2, 12].

The contribution of this paper is to propose a new similarity measure that considers two different kinds of information in sequences, namely the value of attributes and the order of those attributes. Likewise, within our awareness, existing tools for finding similarities between quantitative sequences are hard to find. Therefore we will concentrate in this study on a measure for pair-wise similarities of two-dimesional sequences.

This paper is organized as follows: Section 2 briefly describes sequence alignment methods. Section 3 presents the proposed similarity measure. Section 4 provides detailed experimental results. In Section 5, we examine related work. Finally, in Section 6 we conclude and discuss future work.

2. Background

2.1 Pair-wise Sequence Alignment

Given two sequences of letters and a scoring scheme for evaluating of matching letters, optimal sequence alignment method finds the optimal pairing of letters from one sequence to letters of the other sequence. A good alignment has zero or more gaps inserted into the sequence to maximize the number of positions in the aligned sequences that match. For example, consider aligning the sequences of visited pages on a Web site "$P_1P_2P_4P_5$" and "$P_4P_1P_2P_5P_3P_6$". By inserting gaps *(-)* in the appropriate place, the number of positions where two sequences match can be maximized:

-	P_1	P_2	P_4	P_5	-	-
P_4	P_1	P_2	-	P_5	P_3	P_6

Here the aligned sequences match in three positions. Algorithms for efficiently solving this type of problem are well known and are based on dynamic programming [2].

The structure and the function of lots of sequences in real world are unknown. Only a few sequences are with known structure and function. However, if they align they are similar. If they are similar then they might have same structure or function. Thus, aligned sequences give us important knowledge about how similar the two sequences are. Such knowledge can, for example, be used in determining the behavior of Web users and predicting the next action of them. Similarly, from financial time series data a user may be interested in finding, for example, stocks that had last week a large price fluctuations.

3. Two-Dimensional Similarity Measure

In this section, we propose a similarity measure for two dimensional sequences. The meaning of similarity depends however, largely on data type. As mentioned in the Introduction, the attributes of sequences may have values other then binary. The meaning of similarity may also vary depending on what kind of similarity we are looking for. Two sequences can be determined to be very similar by using only the order information of attributes and to be very different by using the values of attributes. The following are some definitions to formulate the two dimensional sequences.

Definition 3.1 (Two-Dimensional Sequence) A two - dimensional sequence of length n is a n-length sequence of ordered pairs given by $P=[(\alpha_1, \lambda_1^{\alpha}) (\alpha_2, \lambda_2^{\alpha}) \ldots (\alpha_n, \lambda_n^{\alpha})]$ where α_i is the attribute over the set of attributes $\mathbf{A}=\{a_1, a_2, \ldots, a_n\}$ and λ_i^{α} being the corresponding value of α_i over the integers.

In the following definitions $P=[(\alpha_1, \lambda_1^{\alpha}) (\alpha_2, \lambda_2^{\alpha}) \ldots (\alpha_n, \lambda_n^{\alpha})]$ and $Q=[(\beta_1, \lambda_1^{\beta}) (\beta_2, \lambda_2^{\beta}) \ldots (\beta_m, \lambda_m^{\beta})]$ denote two two-dimensional sequences of length n and m respectively. The problem of finding the optimal sequence alignment of two two-dimensional sequences is solved using a dynamic programming formulation. We develop an algorithm based on the algorithm in [2] for this purpose. Briefly, the algorithm consists of three steps. The first step is initialization (Figure 1), where a scoring matrix is created with $n + 1$ columns and $m + 1$ rows where n and m correspond to the size of the sequences to be aligned. One sequence is placed along the top of the matrix (Q) and the other one along the left-hand-side of the matrix (P). A gap is added to the end of each sequence which indicates the starting point of calculation of similarity score. There are three scores in this matrix: $Score_{l,r} = s_m$ which means that the residue at position l of sequence P is the same as the residue at position r of sequence Q (match score); otherwise $Score_{l,r} = s_d$ (mismatch score) or $Score_{l,r} = s_g$ (gap penalty). From this starting point, the last row is filled from right-to-left such that each cell in the last row is the sum of the previous cell and the gap penalty. The last column of the matrix is filled from bottom-to-top in the same manner.

The second step of the algorithm is FindScore (Figure 2), in which we calculate the two dimensional similarity of

```
Input: s_g
Output: Score Matrix M
    M(n + 1 , m + 1) ← 0
    for l = n down to l = 1 do
        M(l , m + 1) ← M(l + 1 , m + 1) + s_g
    end for
    for r = m down to r = 1 do
        M(n + 1 , r) ← M(n + 1 , r + 1) + s_g
    end for
```

Figure 1 : Algorithm of the initialization step

sequences. We implemented the algorithm with a module that takes into account the values of attributes. In our implementation the identical matching of attributes and the values of those attributes is given a score $s_m = 2$ and each mismatch or gap inserted to the sequences is given a penalty score of -1, i.e. $s_d = s_g = -1$. Then the two-dimensional matching score $Score_{l,r} = s(\alpha_l , \beta_r)$ of the matrix is calculated for a pair of matching attributes $\alpha_l = \beta_r = a_k$ where $a_k \in \mathbf{A}$ as follows:

$$Score_{l,r} = s(\alpha_l , \beta_r) = s_m \frac{min[\lambda_\alpha^l , \alpha_\beta^r]}{max[\lambda_\alpha^l , \alpha_\beta^r]}$$

In that step the scoring matrix is filled by starting in the lower right hand corner in the matrix and finding the maximal score M(i , j) for each position in the matrix. Going left corresponds to inserting a gap in sequence P. Going up inserts a gap in sequence Q. Going diagonally up-left corresponds to matching. For each position, M(i,j) is defined to be the maximum of the three incoming values:

```
Input : P, Q, s_m, s_g
Output : Score Matrix M
    for r = m down to r = 1 do
        for l = n down to l = 1 do
            if α_l = β_r then
                Score_{l,r} = s_m · min[λ_α^l , α_β^r] / max[λ_α^l , α_β^r]
            else
                Score_{l,r} = s_d
            end if
            Vertical = M(l + 1 , r) + s_g
            Diagonal = M(l + 1 , r + 1) + Score_{l,r}
            Horizontal = M(l , r + 1) + s_g
            M(l , r) = Max[Vertical , Diogonal, Horizontal]
        end for
    end for
```

Figure 2 : FindScore Algorithm

$M(i,j) = max[M(i+1,j+1)+ Score(match / mismatch in the diagonal),$
$M(i,j+1)+ s_g(gap in sequence P),$
$M(i+1,j)+ s_g(gap in sequence Q)]$

The third step is FindPath which determines the actual alignment(s) that lead to the maximal score (Figure 3). FindPath traverses the matrix beginning from the destination point (upper left corner) to the start point (lower right corner) of the matrix. It takes the current cell and looks at the neighboring cells that could be direct predecessors. This means that it looks at the neighbor to the right (gap in sequence P), the diagonal neighbor (match/mismatch), and the neighbor below it (gap in sequence Q). The algorithm for FindPath chooses as the next cell in the sequence one of the possible predecessors that leads to the maximal score.

```
Input : Score Matrix M
Output: Aligned Sequences Seq1 and Seq2
    Seq1 ← {∅}
    Seq2 ← {∅}
    r ← 1 , l ← 1
    while r < m + 1 do
        while l < n + 1 do
            Vertical = M(l + 1 , r)
            Diagonal = M(l + 1 , r + 1)
            Horizontal = M(l , r + 1)
            max = Max[Vertical , Diogonal , Horizontal]
            if max = Vertical then
                Seq1 ← Seq1 + {-}
                Seq2 ← Seq2 + {α_l}
                l ← l + 1
            else
                if max = Diagonal then
                    Seq1 ← Seq1 + {β_r}
                    Seq2 ← Seq2 + {α_l}
                    l ← l + 1, r ← r + 1
                else
                    if max = Horizontal then
                        Seq1 ← Seq1 + {β_r}
                        Seq2 ← Seq2 + {-}
                        r ← r + 1
                    end if
                end if
            end if
        end while
    end while
```

Figure 3 : FindPath Algorithm

The similarity between sequences is then calculated such that only the identical matching of sequences has a similarity value of 1. The similarity measure has two components, which we define as *alignment score component* and *local similarity component*. The alignment score component computes how similar the two sequences are in the region of their overlap. If the highest value of the score matrix of two sequences, P and Q is σ and the number of matching attributes is M in the aligned sequences, then the alignment score component is:

$$s_a(P,Q) = \frac{\sigma}{s_m * M}$$

The intuition behind this is that score σ is higher if the sequences have more consecutive matching attributes. This value is normalized by dividing it by the matching score and the number of matching attributes. The local similarity component computes how important the overlap region is. If the length of the aligned sequences is L, the local similarity component is:

$$s_l(P,Q) = \frac{M}{L}$$

Then the overall similarity between two sessions is given by:

$$sim(P,Q) = s_a(P,Q) * s_l(P,Q)$$

Example 1 Let us illustrate the calculation of two two-dimensional sequences P = [(a_2 , 1) (a_8 , 1) (a_4 , 2) (a_3 , 2)] and Q = [(a_2 , 1) (a_8 , 2) (a_6 , 2) (a_5 , 1) (a_3 , 1)] where $a_i \in A$ and A={a_1, …, a_n} being the set of attributes. The scoring matrix used in the computation of the similarity is given in Figure 4. Since a_2 is identical in both sequences the matching score of this attribute is $Score_{1,1} = 2 * (1/1)$. Then, the maximum score of that cell in the matrix is $M(1,1) = M(2,2) + Score_{1,1} = 0 + 2 = 2$. However, the value of attribute a_3 is not equal in both sequences. The matching score of that attribute is $Score_{4,5} = 2 * (1/2)$. Then, the maximum score of that cell in the matrix is $M(4,5) = M(5,6) + Score_{4,5} = 0 + 1 = 1$. Since the length of aligned sequences is 5 and the number of matching attributes in the aligned sequences is 3, the overall similarity between these two-dimensional sequences is $sim(P , Q) = (2 / (2 * 3)) * (3 / 5)$.

3.1 Complexity Considerations

The size of the scoring matrix is is $(n + 1) x (m + 1)$ when we consider two dimensional sequences of length n and m respectively. It takes a constant number of cell examinations, arithmetic operations and comparisons to fill in one cell of the matrix. There the FindScore algorithm takes $O(mn)$ time and $O(mn)$ space to compute. If the sequences compared are fairly short, the quadratic behavior of FindScore algorithm is not a problem. However, if the sequences are typically very long, it should be better to compute the similarity of sequences with more efficient algorithms than just dynamic programming [3].

On each iteration in FindPath algorithm, either the index l, the index r or both of them, is decremented. This means, that the maximum number of iterations is $m + n$, and that the best alignment for two sequences is done in $O(m + n)$ time.

	a_2	a_8	a_6	a_5	a_3	-
a_2	**2**	-1	-2	-2	-2	-4
a_8	-1	0	-1	-1	-1	-3
a_4	-3	-2	-1	0	0	-2
a_3	-3	-2	-1	0	1	-1
-	-5	-4	-3	-2	-1	0

Figure 4: The scoring matrix for two-dimensional sequences

4 Experimental Results

In this section, we present some results of experiments on similarity between two-dimensional sequences. In subsection 4.1, we describe the data sets used in the experiments. All Experiments were performed on a Pentium II, 333 MHz PC with a 512 MB main memory running on Microsoft Windows 2000. The programs are coded in Java without code optimization

4.1 Data Sets

The experiments on similarity between two-dimensional sequences were made with two different data sets of WWW logs. The first data set is from the NASA Kennedy Space Center (NASA) server over the months of July and August 1995 [4]. The second log is from ClarkNet (C.Net) Web server which is a full Internet access provider for the Metro Baltimore-Washington DC area [5]. This server log was collected over the months of August and September, 1995.

A Web server log is an important source for performing Web usage mining because it explicitly records the browsing behavior of site visitors. The server records the time and date of the transaction. It records the name of the file that was sent and how big that file was. It records the Internet address to which the file was sent. If the user goes to a page by clicking a link on some other page, the server records the address of the page with that link. It also records some details about how the file was sent and any errors that may have occurred as well as information about the browser that the user is using. The data recorded in the server logs reflects the (possibly concurrent) access of a Web site by multiple users. The information provided by the Web server can all be used

to construct a data model consisting of several abstractions, notably, users, pages, click-streams, server sessions. Since user sessions are ordered URL requests, we can refer to them as sequences of Web pages. The behavior of Web users can be modeled using sequences. However, an important feature of the user's navigation path is the time that a user spends on different pages [6]. Thus, the user navigation path turns into two-dimensional sequences. Each data set is cleaned in order to obtain user sessions where each user session is defined as the sequence of page views for a single visit of a user to a Web site [7]. The cleaning step is beyond the scope of this paper and the details of this step are given in [8]. After this step each user session is represented by the set of ordered Web pages that a user accessed during his or her single visit to the Web site and corresponding time information.

4.2 Results and Discussion

Approximately 30% of the cleaned user sessions of each data set are randomly selected as the test set, and the remaining part as the training set. After each request of user sessions in the test set we query the training set in order to find the most matching sequence in it. If two user sessions from the test set and training set are aligned with a high similarity value it means that they have similar structure and function. Thus, the pages of the matching user session in the training set can be used to find out the next page request of users in the test set. Based on the best matching sequence in the training set three pages are recommended to the user in the test set. A hit is declared if any one of the three recommended pages is the next request of the user in the test set. The hit-ratio is the number of hits divided by the total number of recommendations made by the system. A high hit-ratio means that the similarity measure is successful in finding most similar two-dimensional sequences.

We have made three tests in order to evaluate the similarity measure and the effect of using quantitative values of attributes. In the first test we compare the two-dimensional sequences with binary sequences. Figure 5 shows these results. As can be seen from the figure using two-dimensional sequences improves the hit-ratio. These results prove that using quantitative values of attributes in the sequences provides useful knowledge in web domain.

Data Set	Q-Values	B-Values
NASA	61.61	59.9
C.Net	55.76	51.29

Figure 5: Results in %. (Q-Values = Quantitative values, B-Values = Binary values)

In the second test we perform graph based clustering [9] based on the pair-wise similarities of user sessions in order to examine the effect of the similarity measure more deeply. Since the clustering is based on the proposed similarity measure the structure of the resulted clusters is tightly correlated with this measure. Figure 6 illustrates these results with 5 clusters. The effects of the number of clusters are examined in [8].

Data Set	Q-Values	B-Values
NASA	57.41	56.19
C.Net	53	51.09

Figure 6: Results in % for 5 clusters.

In the third test, we run the experiments using another similarity measure. The similarity between two user sessions is calculated only using the length of common subsequence and the length of the user sessions. For example, let two user sessions be $P=P_1P_2P_3P_4$ and $Q=P_2P_4P_5P_6$. Since the length of matching subsequence is 2 (P_2P_4) and the length of both sessions is 4, the similarity is $sim(P,Q) = \sqrt{(2/4)*(2/4)}$. Our similarity measure using time information performs about 8% better than this similarity measure. Even without using time information our measure results in about 4% better then this one. Figure 7 illustrates these results.

Data Set	Q-Values	Diff. Measure
NASA	61.61	57.04
C.Net	55.76	52

Figure 7: Comparison of similarity measures.

These results prove that using two-dimensional sequences in web domain improves the prediction accuracy. Furthermore, the proposed similarity measure is effective for finding pair-wise similarities of two-dimensional sequences.

5 Related Works

Similarity searches in sequence databases are important in many application domains, such as information retrieval, web mining and clustering. Detecting stocks that have similar growth patterns, finding web users with similar interests are a few examples of similarity queries. In particular, we usually assume that similar sequences hold a similar function and structure. It is not surprising, thus, that among the first tools developed in the field of sequence analysis were those aimed to determine the degree of similarity between two sequences.

Most of the previous techniques [10, 11] for similarity searches use the Euclidean distance measure as a similarity

measure. However, in many applications, the length of sequences may be different, making it difficult or impossible to use the Euclidean distance as a similarity measure.

A sequence matching method for sequences of different lengths based on dynamic programming is proposed in [2]. However, this approach is appropriate for sequences having binary valued attributes, such as biological sequences. While we focus on the sequences of continous numeric values, the approches of [2, 12] center on the sequences of charecters. Our approach combines both the advantages of Euclidean techniques and the advantages of dynamic programming techniueqes.

6 Conclusion

In this paper, we considered the problem of finding pair-wise similarities between sequences of continous numeric values. We introduce a similarity measure for finding pair-wise similarities of two-dimensional sequences. This similarity measure compares two sequences by means of their ordered attributes and corresponding values of those attributes. The current implementation of this approach is adapted to a recommendation model for web users. Another application domain could be market basket analysis. The proposed similarity measure may be useful to build a predictive model. Such a model will help to categorize and segment customers based on their purchase (or navigation) patterns.

References

[1] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy: "*Advances in Knowledge Discovery and Data Mining*", MIT Press, Cambridge, MA, 1996.

[2] K. Cahrter, J. Schaeffer, and D. Szafron, "Sequence Alignment Using FastLSA", *Proc. Int. Conf. on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'2000).*, pp. 239-245, 2000.

[3] G. A. Stephen, *"String Searching Algorithms"*, World Scientific Publications, Singapore, 1994.

[4] NASA Kennedy Space Center Log., http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

[5] ClarkNet WWW Server Log, http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html.

[6] C. Shahabi, A. Zarkesh, J. Adibi, and V. Shah, "Knowledge Discovery from Users Web-page Navigation", *Proc. 7th Int. Workshop on Research Issues in Data Engineering*, pp. 20--29, April 1997.

[7] R. Cooley and B. Mobasher and J. Srivastava, Data Preparation for Mining World Wide Web Browsing Patterns, Journal of Knowledge and Information Systems, **1**:1, pp. 5-32, 1999.

[8] Ş. Gündüz, M.T. Özsu, A Web Page Prediction Model Based On Click-Stream Tree Representation of User Behavior", *Proc. of Ninth ACM International Conference on Knowledge Discovery and Data Mining* (KDD), Washington, DC, pages 535-540, 2003.

[9] Cluto, http://www-users.cs.umn.edu/~karypis/cluto/index.html

[10] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, "Fasr Subsequence Matching in Time-Series Databases", *Proc. 1994 ACM SIGMOD Conference*, Mineapolis, MN, 1994.

[11] B.K. Yi, H.V. Jagadish, C. Faloutsos, Efficient Retrieval of Similar Time Sequences Under Time Warping, *Proc. IEEE ICDE*, 1998.

[12] J. T. L. Wang, G-W. Chirn, T. G. Marr, B. A. Shapiro, D. Shasha, and K. Zhang, "Combinatorial Pattern Discovery for Scientific Data: Some Preliminary Results", *Proc. 1994 ACM SIGMOD Conference*, Mineapolis, MN, 1994.