Time Series:Similarity Search and its Applications

Tripti Negi and Veena Bansal

Abstract

A model for identifying similar time series has been developed. Two time series are considered similar if they have enough non overlapping time ordered subsequences that are similar. The two subsequences are considered to be similar if one is enclosed within an envelope of a user defined width around another. Amplitude of one of the two sequences is scaled by a suitable amount to make the matching scale invariant. We also allow a small gap between the matching subsequences. A static index structure namely kd-tree is used to index the time series and the Range search methodology is used to search for the similar subsequences. The highlight of this model is that the similar subsequences are not stored repeatedly in the index structure, but only the time stamp corresponding to it is updated in the external node of the kd-tree.Interesting results are found to show the validity of the model.

1 INTRODUCTION

A time series is a sequence of real numbers, each number representing a value of an attribute of interest, observed at different point of time. Typical examples include stock prices, currency exchange rates, the volume of product sales, biomedical measurements, weather data, etc, collected over monotonically increasing time. There are several important aspects of mining time series including trend analysis, similarity search and mining of sequential and periodic patterns in time related data. Our work mainly focuses on the similarity search in time series. Unlike normal queries, which find data that match the given query exactly, a similarity search finds data sequences that differ only slightly (user defined parameter) from the given query sequence. Given a set of time series sequences, there are two types of similarity search. Subsequence Matching finds all the data sequences that are similar to the given sequence, while whole matching finds those sequences that are similar to one other.

The following are some examples of the similarity queries

- Identify companied with similar growth pattern.
- Determine products with similar selling patterns.
- Discover Stocks with similar movement in stock prices.
- Find if a musical score is similar to one of the copyrighted scores.
- Find portions of seismic waves that are not similar to spot geological irregularities.

Given two time series,a standard approach for similarity is to compute Euclidean distance. In order to support efficient retrieval and matching of time series, indexing of time series data is needed. An indexing structure [11] was proposed for fast similarity searches over time-series, when the data as well as query sequences were of the same length. Discrete Fourier Transformation (DFT) is used to map a time sequence to the frequency domain, the first few frequencies are retained and the retained frequencies are used to store the subsequences in the R* tree [10] structure, to create the index for efficient and fast search. This work was generalized [3] to allow subsequence matching. Data sequence could now be of different lengths and the query sequence could be smaller than any of the data sequence.

This earlier work, while pioneering, has the following limitations for employing it in practical applications:

- The problems of amplitude scaling and offset translation have not been addressed.
- The problem of ignoring unspecified portions of sequences while matching sequences is not addressed.

The above mentioned shortcomings are taken care in the technique proposed by Agrawal et.al [12].According to the paper,two sequences are considered to be similar if they have enough non overlapping ordered pairs of subsequences that are similar. Amplitude of one of the two sequences is allowed to be scaled by any suitable amount and its offset adjusted

appropriately.For testing similarity, it is checked if one lies within an envelope of specified width around the other.

For time series numeric sequences, the sequence is divided into subsequences of length *'w'* called as window. *'k'* features are extracted from every subsequence forming a feature vector. Then a multidimensional indexing technique indexes these feature vectors for efficient search of atomic subsequences. R-tree [6] and its variant [10] [15], being dynamic indexing structures; is used for indexing.

kd-tree [1] was studied extensively by Friedman et.al [7] is used here for indexing the normalized subsequences of user defined length. Range Search [2] is used to find similar subsequences within an user defined distance. The index once created is neither updated nor deleted, hence this method is proposed to use. The extracted feature in this case are the normalized subsequences of length equal to the window size.For Range Search a target subsequence is considered similar to source subsequence, if the target lies within the user defined distance ϵ from source. The main highlight of kd-tree is that if more than one subsequences of the source are found to be similar, only one of them is stored as the external node. For the other similar subsequences, only the time stamp of the new subsequence is updated to the already existing node.

2 Literature Review

The work for time series similarity began with work done by Agrawal and group [11] proposing the technique for fast similarity search for time series. He [12] also developed the similarity model with noise tolerance and translation. Wavelet transformation [8] is introduced for time series similarity.The main advantage of using Discrete Wavelet Transformation is multi-resolution representation of signals in both the time and frequency domain. The paper by Xianping et.al [5] addresses the problem of detecting specific patterns or shapes in time-series data based on segmental Semi-Markov models. Pyramid technique [13] adapts well to high dimensional data queries. It is based on a special partitioning strategy, which divides data space into pyramids.The performance of this technique does not deteriorate when processing range queries on data of high dimensionality.But it performs worse than the sequential scan for much skewed queries. Landmark Similarity Model [4] is a general model of similarity that is consistent with human intuition and episodic memory.The gist of the model is to use landmarks, which are important observations taken at particular time instead of the raw data for processing.The method proposed by Wang et.al [16], is an indexing method for the subsequence matching of spatial objects motivated by String searching technique.The main drawback of this technique is that as the size of alphabet continues to increase, the percentage of false alarms increases.

All the above proposed time series matching templates require multidimensional indexing techniques. G-tree[9] supports efficient storage and retrieval of point data.It allows a region to be divided into subregions of different sizes.The most important application is its used to index structure for composite keys. Quadtree [14] are a class hierarchical data structure used for efficient indexing of the images.

3 SIMILARITY MODEL

Two sequences are considered similar if they have enough non overlapping time ordered pairs of subsequences that are similar. The amplitude of one of the two sequences is allowed to be scaled by any suitable amount and its offset adjusted properly. For testing the similarity of two subsequences, it is checked if one lies within an envelope of a specified width around the other. Envelope width is the user defined parameter. The matching subsequences need not be aligned along the time axis. Corresponding to the target sequence, once the pair of matching subsequences are found, they are stitched together, within some user defined permissible gap.

3.1 Technique Used

This technique is mainly divided into three phases. The first phase consist of preprocessing the input. It prepares the data so that it can be used by the

other phases. The second phase is Atomic Matching which finds the source subsequences matching the target subsequences. The basic idea is to create a fast, static, indexable data structure that will facilitate matching of small, atomic subsequences that represent sequences.Certain considerations are essential before selecting the indexing structure.They are described as follows:

- *Dimensionality*: The window size varies between 5-20. Thus the index structure selected should be capable of handling dimensions in this range.
- *Static Indexing Technique*: Once the subsequences are indexed,there is no need to either update or delete them.So static indexing is most appropriate for our model.
- *Data Values*:All windows are normalized in a range of(-1,1), any point will always have a -1 and 1 value for is coordinates. Thus, many points will lie on same hyperplane.

Hashing [**?**] is one of the static indexing scheme but due to the following limitations is not used.

- The number of entries in the hash table can become very large, making it unmanageable.
- Many windows may hash on the same location, thus most of the time rehashing has to be used.

Thus the kd-tree [1] is used for indexing purpose. It stores the subsequences of window size *'w'*, in the external node of the tree. For the *'w'* dimensional vector, all points are important and they act as the key for the tree. At the root the split is done along the first dimension taking median value of the subsequences.The hyper rectangle to be stored at the root is specified.Here the subsequences are normalized in the range of (-1,1). Hence, the initial hyper rectangle has the minimum value -1 and maximum value 1 along all the dimensions. At each node, hyperrectangles to be stored at the left and right subtree. This process of splitting and creating the median is repeated, taking all dimensions into consideration.

The construction time of the tree requires a pivot and a splitting plane to be selected, which is the most expensive step.The splitting plane is determined by finding the median of the points along all dimensions. This process can be done in linear time by presorting the set of points along all dimensions. The building time *T(n)* satisfies the recurrence

$$T(n) = \begin{cases} O(1), & \text{if } n = 1 \\ O(n) + 2T(n/2) & \text{if } n > 1 \end{cases}$$

which solves to *O(nlogn)*. This bound subsumes the time spent for presorting the points on all dimensions. After the index has been built, the Range Search method is employed for finding all the matching subsequences. The third phase is Subsequence Matching, which stitches all the windows to form a long subsequence matching the target subsequence.

3.1.1 Preprocessing the Input

For finding the similarity between two time sequences the amplitude scaling and offset translation problem are taken into account.To account for this, a sequence within each window *'w'* is normalized to a range of (-1,1). The new window *'W'* is formed using formula

$$W[i] = w[i] - \tag{1}$$

where w_{min} and w_{max} are the minimum and maximum values in the window *'w'*. To build the index structure the points in each subsequence are stored.For this purpose each sequence is scanned from beginning to end, extracting and normalizing the *w*-dimensional point corresponding to each window. Attached with each points are its coordinates, time stamp, maximum and minimum values.

3.1.2 Atomic Matching

All pairs of gap free subsequences of length *'w'* called windows are found in source and target sequences. Each window is considered as a point in *'w'* dimensional space. To build the index, each sequence is scanned from beginning to end, extracting and normalizing the point corresponding to each window. The normalized point is then inserted in the index. If more than one *'w'* dimensional point is found to be similar, they are not inserted in index, but only the

time stamp are updated. Attached with each point are its coordinates, time stamp, maximum and minimum values. The algorithm below gives the method to create the kd-tree. In the algorithm given below ***P*** indicates the set of *'w'* dimensional points to build the index and ***depth*** has the maximum value w.

Algorithm CreateIndex(P, depth)

depth=0
if P contains only one point
    then return only one point
        while depth not equal to w then
        split P into two subsets with a line
        along dimension depth and finding
        median along that dimension
    Let P1 and P2 be the two subsets
    of points formed by splitting
        increment depth
        vl= CreateIndex(P1,depth+1)
        v2= CreateIndex(P2,depth+1)
        if depth== w
        depth=0
Create node v storing median, making vl
left child of v and vr the right child of v.
return v.

Once the kd-tree is constructed, Range Search [2] is performed to find all subsequences which lie within hyper rectangle having range *epsilon* and ϵ. In a Range query,a range of values are specified for each of the *'k'* keys and all records that have every value in the proper ranges are then reported as answer.To accomplish the search hyper rectangle is created using the target subsequence and the user defined ϵ. For example, N is the hyper rectangle at the root and P is the query rectangle created by the target subsequences points.There are three scenarios at the root.

- If N is in P, then all points under the root node are inside P.
- If N does not intersect with P, then the tree is not traversed.
- If there is intersection between N and P, then depending upon the median value at the node, either left or right subtree is traversed.

The asymptotic searches required is logarithmic as the time required to descend from the root to the leaves is logarithmic, which results in efficient search. The algorithm given below describes the Search process. The ***node*** indicates the node created by the kd-tree. If the value of this parameter is zero it indicates that it is the leaf node else its internal node. The keyword ***hyper*** indicates the hyper rectangle created by the target subsequence and the ***rect*** is the rectangle which is stored at the node of the tree.***index*** is the index corresponding to the matched source subsequence.

Algorithm Search(hyper,node,index)

```
if(node == 0)
   if(rect lies in hyper == true)
       return index
 if (node ==1)
       if(rect intersects hyper == true)
       index = Search(rect, node.left,index)
       index = Search(rect, node.right.index)
  return index
```

3.1.3 Subsequence Matching

This module stitches the atomic window obtained in the previous step to form the target subsequence that matches the source. The output of this step is the pairs of matching windows for every pair of sequences S and T. Let $M=(S_i, T_i)$ and $N=(S_j, T_j)$ be the source and the target subsequence obtained from atomic matching. They can be stitched if and only if the following conditions are satisfied.

- The time stamps of the subsequences in N are later than the time stamps of the subsequences in M.
- Either one of the following is true:
    1. The corresponding windows in the two matches do not overlap and the gap between them is $\leq \gamma$.
    2. The amount of overlap between S_j and S_i in S is same as the amount of overlap between T_j and T_i in T.

The above steps are repeated for all the target subsequences having some matching source subsequences corresponding to it.

4 Experimental Results

In this section we present the experimental results.The data for the experiments consists of various time series database collected from National Stock exchange (www.nse-online.com) and for the Automobile Industry (wwww.indianinfoline.com) .

4.1 Analysis of the Model

Experiments are conducted for different combinations of window sizes, distance and gap values.All these parameters affect the similarity model in different ways. Distance deals with the precision of the match.For example if the distance is taken as zero then exact match is found, but for non zero values approximate match is found. The first experiment was conducted to view the effect of the window size and the distance parameter on the model. Experiments are conducted for Infosys data for a duration of five years from 1^{st} January 1997 to 1^{st} January 2002 excluding weekends and holidays. The distance is varied from 0.1-0.8 and for each distance parameter the window size considered was 5, 10, 15. The results are shown in the graph in figure 1. As the user defined distance parameter increases the number of matching subsequences increases. The reason behind this increase is with the increase in distance the permissible range for similarity also widens. As the window size increases there is no significant variation in the number of matching subsequences.This is attributed to the fact that the range within which the subsequences are searched depends only on the distance value.

4.2 Results

All the experiments here are done for following input parameters. Window size=10, distance=0.2 and gap=8.

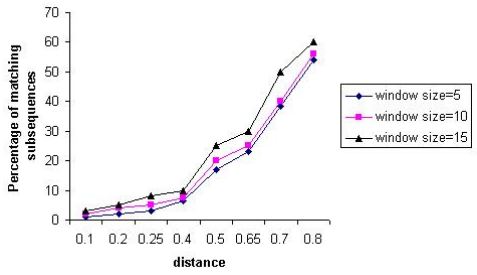

Figure 1: Effect of Distance parameter on the number of matching subsequences

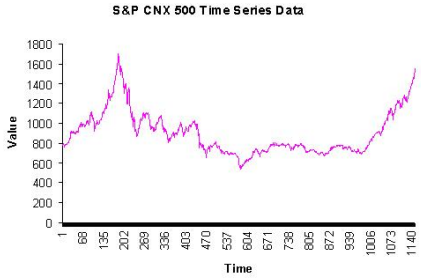

Figure 2: Source Data

4.2.1 Similarity between two time series

SP CNX 500 Index data from NSE for the duration of four years from January 1999-July 2003 is taken as the source data. Target data is some variation of the source. The time stamp 1^{st} January 1999 is changed to one and 29^{th} to 1149. The source and the target time series are shown in the figures 2 and 3 respectively.

The figure 4 shows the portion of the time series that are matching. Non matching areas are shown in the figure.

4.2.2 Analysis of Growth for Exports in Automobile Industry

Automobile Industry, specially two wheelers industry, plays an important role in the economy of the system.

Figure 3: Target Data

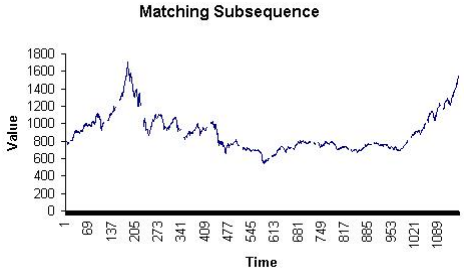

Figure 4: Matching Subsequences between the source and target data

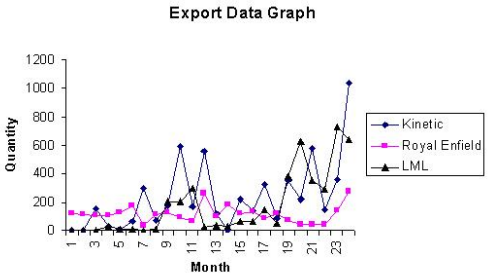

Figure 5: Export Data Quantity

For the analysis motor cycle industry data for exports quantity for a duration from April 2002 - March 2004 is used. Three leading motor cycles production units namely Kinetic Engineering, Royal Enfield and LML are considered. The graph in the figure 5 shows the number of bikesof each type exported.

The matching is done between the following source/target sequences.

- Kinetic Engineering data as source, Royal Enfield as target.
- Royal Enfield as source, LML as target.
- Kinetic Engineering as source, LML as target.

No match is found between any of the source and the target sequences. This is also verified from the graph in figure 5.

5 Conclusion and Future Scope of Work

We addressed the problem of matching two time series for similarity. The similarity model describes a fast search technique to discover all similar subsequences in a set of sequences.Searching a target subsequence requires$O(logn)$ comparisons on an average, where n is the number of subsequences.Thus the search technique is very effective. The highlight of this method is the reduced number of comparisons required for matching source subsequences with the

target. It also does not store the subsequences more than once if they are repeated, thus reducing the storage space. We have been able to demonstrate that this model works well with a variety of time series data. The indexing structure created using kd-tree is stored in the main memory.It is done considering the fact that the cost of the memory is decreasing rapidly. If the data is large and the index structure spills over to secondary memory the performance of the system suffers. We leave a suitable modification to the system that stores the index in the secondary memory as future work.

References

[1] J. Bentley. Multidimensional binary search trees used for associative searching. *Communications of ACM*, 18(9):509–517, September 1975.

[2] J. Bentley and J.H. Friedman. Data structures for range searching. *ACM Computational Surveys*, 11:397–409, 1979.

[3] M. Ranganathan C. Faloutosos and Y. Manolopoulos. Fast subsequence matching in time-series databases. *In Proc. of the ACM SIGMOD Conference on Management of Data*, May 1994.

[4] S.Zhang C. S. Perng, H. Wang and D. S. Parker. Landmark: A new technique for similarity based pattern querying in time series databases. *In Proc ICDE*, 2000.

[5] Xianping Ge and P. Smyth. Deformable markov model templates for time series pattern matching. *Intl Conf. of SIGKDD*, August 2000.

[6] A. Guttman. R-trees: A dynamic index structure for spatial searching. *In Proc. ACM SIGMOD*, pages 47–57, June 1984.

[7] J. L. Bentley J. H. Friedman and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3:209–217, 1977.

[8] Chan. K and A.W. Fu. Efficient time series matching by wavelets. *In Proc. of the 15th Intl. Conf. on Data Engineering*, pages 126–134, 1999.

[9] A. Kumar. G-tree: A new data structure for organizing multidimensional data. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):341–347, April 1994.

[10] R. Schneider N. Beckmann, H. P. Kriegel and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. *In Proc. of ACM SIGMOD*, pages 322–331, May 1990.

[11] Christos Faloutsos R. Agrawal and A. Swami. Efficient similarity search in seqeunce databases. *In Proc. of the Fourth Int. Conf. on Foundations of Data Organizations and Algorithms*, pages 1–15, 1993.

[12] Harpreet S. Sawhney R. Agrawal, K. Lin and K. Shim. Fast similarity search in the presence of noise, scaling and translation in time-series databases. *In Proc of 21st VLDB Conference*, pages 490–501, 1995.

[13] Christain Bohm S. Berchtold and H. Kriegel. The pyramid technique : Towards breaking the curse of dimensionality. *In Proc. of ACM SIGMOD Conference on Management of Data*, 1998.

[14] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260, June 1984.

[15] T. Roussopoulos T. Sellis and C. Faloutsos. The r+ tree: A dynamic index for multi-dimensional objects. *Proceedings of the 13th VLDB Conference*, pages 507–517, 1987.

[16] H. Wang and C. H. Perng. The s^2 tree: An index structure of subsequence matching of spatial objects. *PAKDD, Hongkong*, pages 312–323, 2001.