MALM: A Framework for Mining Sequence Database at Multiple Abstraction Levels *

Chung-Sheng Li, Philip S. Yu, and Vittorio Castelli
IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598
Telephone: (914) 784-{6661,7141,7665}
Email: {csli,psyu,vittorio}@watson.ibm.com

Abstract

Similarity searches of sequences are usually performed in the time domain using techniques such as template matching, or in appropriate feature spaces where feature vectors are pre-extracted. In this paper, we propose an object-oriented framework, MALM, where the similarity search is performed on time series objects (or segments), where the matching of each object can be performed at multiple abstraction levels extracted from the time series. In the proposed approach, the time series in the databases are first segmented with a regression algorithm. Features such as regression coefficients, mean square error, and higher-order statistics based on the histogram of the regression residuals are extracted from each segment. A neural network clustering algorithm is then used to assign labels to each segment. The clustered time series segments can then be queried at the symbol level, the feature level, or the sequence level. This framework provides a scale- and translational-invariant mechanism for the user to generalize the query template for similarity and all-pair searches. Numerical results are obtained based on the 20-year stock sequence database from 36 companies.

1 Introduction

Temporal or spatial-temporal data constitutes a large portion of the information stored in computers [18, 5]. Examples of this type of database include: (1) time series such as stock price index, the volume and revenue of product sales, insurance claims, etc.; (2) medical databases such as 1D signals (e.g., ECG), 2D images (e.g., X-rays), and 3D images (e.g., MRI, CT, and PET); (3) multimedia databases which contains audio, image, and video data; (4) multispectral satellite image databases.

Searching for similar patterns in a temporal or spatial-temporal database is an essential operation in many data mining applications [2, 1, 14] in order to discover and predict the risk, causality, and trend associated with a specific pattern. Typical queries for this type of database include

- to identify companies with similar growth patterns, products with similar selling patterns, or stocks with similar price movement;
- to identify data with similar weather patterns, geological features, environmental pollution, or astrophysical patterns.

These queries invariably require similarity matches rather than exact matches. Two query paradigms are usually adopted in various data mining operations: *query-by-example* (e.g., range query or similarity query) in which a search is performed on a collection of objects to find the ones that are within user-defined distance from the queried target, and *all-pairs query* (e.g., spatial join) where the objective is to find all the pairs of elements that are within a user-specified distance from each other. In this paper, we shall consider both types of queries, and we shall devote special attention to databases with very long sequences.

Significant progress has been recently made in sequence matching [1, 9, 8, 3, 4, 13, 17, 7, 6, 16, 11]. Two types of similarity queries for temporal data have emerged thus far: *whole matching* [1] in which the target sequence and the sequences in the database have the same length; *subsequence matching* [9] in which the target sequence could be shorter than the sequences in the database and the match can occur at any arbitrary point. A shape definition language is proposed in [4] to search time series at the semantic level (by specifying the behavior using codewords such as "rise", "decrease", etc.).

These approaches search the time series either in the time domain [3, 13] or in an appropriate feature space, as in [1, 9, 8] where the search is performed on the first few Fourier coefficients. A single similarity index, such as the Euclidean distance, is customarily used for measuring the similarity between the search target and the time series in the database. This approach is inadequate to express complex search targets involving multiple levels of abstractions and details.

This paper proposes a framework, *Multiple Abstraction Level Mining (MALM)*, on time series data, partly inspired by [4, 17, 7]. The major contributions of the proposed framework are (1) a non-redundant representation of the time series, which allows similarity search at multiple abstraction and resolution levels, (2) an algorithm based on existing segmentation and clustering techniques to generate representations of the time series at multiple abstraction levels, and (3) a framework for combining search results at multiple abstraction levels.

*This work was funded in part by grant no. NASA/CAN NCC5-101

CIKM 98 Bethesda MD USA

In this framework, representations at multiple abstraction levels are generated from the same time series. The time series itself is at the lowest level. The time series is then segmented into non-overlapping subsequences with the divide-and-conquer technique proposed in [17], A linear approximation to each segment is generated, using standard regression techniques. Features, such as the slope of this linear regression, the mean square error and other statistics of the residuals, are then extracted, and constitute a feature level representation of the data. Unsupervised classification algorithms, such as Kohonen self-organization map, or supervised classification algorithms, such as back-propagation neural networks, can then be applied to the extracted features to produce a semantic representation of the data. Query of the time series can then be expressed using, for example, the shape definition language proposed in [4] with constraints at the time series level, the feature level, or the semantic level. The performance of this framework is evaluated on a stock sequence database of the 36 companies spanning from 1964 to 1984.

2 Multiple Abstraction Level Mining Algorithms

2.1 Representing Time Series at Multiple Abstraction Levels

In this work, time series are represented by means of objects. We distinguish between atomic and composite time series objects. An atomic object is defined as a homogeneous subsequence, while a composite object consists of multiple atomic objects composed via Boolean, temporal or set operators.

An atomic object can be specified at different abstraction levels. The lowest level is the raw time series. The simplest definition of an object type at the raw data level consists of specifying a template sequence, a distance measure and a threshold value: all the subsequences whose distance to the template sequence is smaller than the threshold belong to the defined type. The next higher level is the feature level; here some defining characteristics of the subsequences are extracted by means of signal processing operators, and used as descriptors. Examples of features are quantities derived from the power spectrum of the data, parameters of ARMA or of martingale models fitted to the data, etc. The simplest object type definition at the feature level consists of specifying a feature, a value, a distance measure, and a threshold: all the subsequences whose feature is closer to the specified value than the threshold belong to the defined object type. The highest abstraction level is the semantic level. Subsequences here are described in terms of semantic attributes, and objects with the same attribute value belong to the same type. Examples of semantic descriptions are the labels of electrocardiogram cycles and rhythms.

The construction of a multiple-level description of the data can often be performed automatically, or with partial involvement of a human expert, and leads to the data structure shown in Fig. 1.

The original time series is stored in its entirety in the database. The extracted feature sets are stored in a feature vector table, where each vector points to the starting and ending location of a subsequence derived from segmenting the original time series. The cluster or class labels are stored in another table with pointers to the corresponding feature vectors.

It is worth noting that by applying different segmentation algorithms, one can generate different representation of

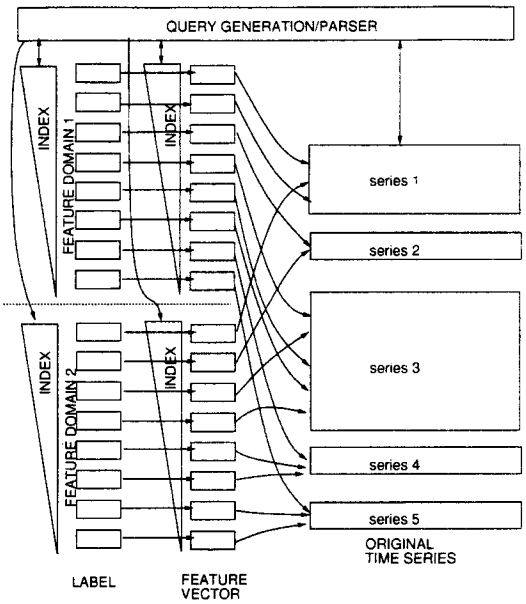

Figure 1: Data structure for representing time series at multiple abstraction levels.

the same time series. Furthermore, different feature sets can be extracted from the same set of subsequences, and therefore the sequences can be represented in a multiplicity of feature domains. (Two feature domains are shown in Fig. 1 (a).) Indices can be built to facilitate the search of objects at either the semantic or the feature level. A simple B-tree is usually sufficient for indexing the objects at the semantic level, while sophisticated spatial indexing techniques (such as the R-tree family) are required for feature space indexing. The storage requirement for a database consisting of D time series using the described representation is

$$\begin{aligned} S &= D(T + k_f F(T/S) + k_s(T/S)) \\ &= DT(1 + (k_f + 2)F/S + k_s/S) \end{aligned}$$

where T is the average length of the time series, S is the average length of a time series segment, F is the number of dimensions of the feature vector extracted from the time series, k_f and k_s are the coefficients associated with the overhead associated with storing the features and the semantics of each time series segment, including the storage space required for indices, and the constant 2 is the storage required for the starting point and the length of each subsequence. Consequently, the storage space only grows linearly with the number of entries in the database and length of each entry.

2.2 Generation of Multiple Abstraction Levels

In the proposed method, different abstraction levels are extracted from the sequence database through the following steps:

1. Partition each sequence in the database into disjoint segments: Each of these segments has homogeneous features. Methods for segmenting time series exist, such as the one proposed in [17], in which a sequence divided into two subsequence if the maximum error between the original sequence and the fitted curve is larger than a certain tolerance.

2. Extract features from each segment: The features extracted in this paper include the regression coefficients, the length of each segment, the mean square error between the original subsequence and the regression curve, (i.e., second moment of the residuals) and other statistics derived from the histogram of the residuals.

3. Assign a label to each segment by using one of the clustering/classification algorithms such as k-means, Kohonen self-organization map, etc.

Note that the segmentation step is closely related to the clustering step, and heavily depends on the *vocabulary* of the clustering/classification. As a simple example, the vocabulary of the clustering/classification could consists of only *rise, flat, and fall.* The first and third categories can be further divided into steep and slow.

3 Queries with Constraints at Multiple Abstraction Levels

A similarity query on the objects in the sequence database can be specified as a set of atomic or composite objects at the semantic, the feature, or the raw time series levels. Here we distinguish between two types of queries: (1) search for an atomic object with Boolean or temporal constraints, and (2) search for a composite object, defined in the previous section as a set of atomic objects related via Boolean, temporal or set operators. In the first case, the emphasis is on the atomic object, that represents an event satisfying the Boolean and temporal relationships; for instance, we might ask to find all the sharp rises of technology stocks longer than three days and immediately following a sharp decline of the value of the U.S. dollar relative to the Yen. The second case the emphasis is on the sequence of events: for instance, we might ask to find all the stocks that declined for less than two days and then rose sharply for at least three days after a fall of the Dow industrial average of more than 100 points in a day.

A sample query searching for an atomic object with temporal constraints is "FIND S_1 *where* F_1 *before and* F_2 *after*". This query searches for subsequence objects that are similar to the semantic object S_1, are preceded by the feature object F_1, and are followed by the feature object F_2. A sample query, searching for a composite object with temporal constraints, is: "F_1 S_1 F_2," where the target object consists of a feature object F_1, followed by a semantic object S_1, followed by another feature object F_2. These objects are processed as follows:

- Matching at the semantic (symbol) level: Objects at the semantic (or symbol) level are designated by their cluster numbers, or their class *labels.* Well known search techniques such as B-tree can significantly improve the performance of this kind of search.

- Matching at the feature level: Objects at the feature level are defined by their corresponding feature vectors. The search is usually based on *similarity* comparison rather than on exact match, and the retrieved results are ranked according to a *similarity index*, such as the Euclidean distance between the target sequence and the sequence in the database. To facilitate nearest neighbor or range query in the feature space, spatial indexing techniques such as the R-tree family are commonly adopted.

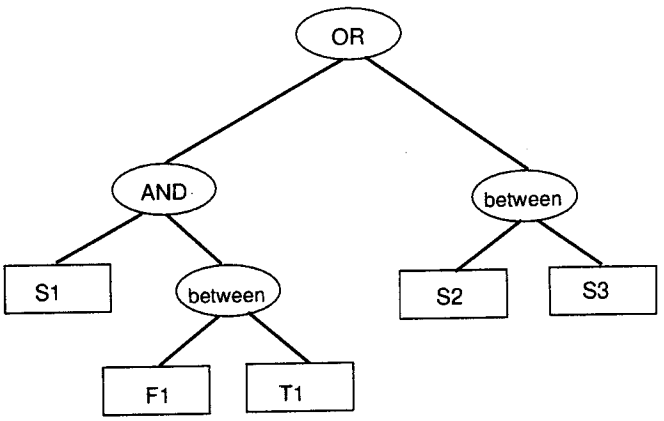

Figure 2: Parse tree of a typical query.

- Raw time series search: In this paper, we use correlation as similarity measure between the target sequence and the sequences in the database. This measure both gives the relative similarity as a function of location and eliminates the need to generate all the subsequences of given length n of each time series in the database. Efficient correlation evaluation techniques exist, such as *HierarchyScan*[13].

- Combining results: The query results from a similarity search are usually ranked according to a similarity index, or score, with a higher score indicating a higher similarity. This score can result from a search in the feature space, at the semantic level or from computing correlation coefficients in the raw time series domain. Multiple constraints can be composed using (1) Boolean operators, such as AND, OR, NOT; (2) Temporal operators such as before, after;

The following example illustrates how to formulate a query at multiple abstraction levels. In this case, we like to retrieve subsequences that are *similar to* S_1, *and occur between* F_1 *and* T_1, *or between* S_2 *and* S_3. S_i are subsequences specified at the semantic level, F_i are subsequences specified at the feature level, and T_i are subsequences specified in the time domain. These constraints are parsed into a tree representation, as shown in Fig. 2. The tree traversal order determines the order of execution. Each specification in a query phrase (e.g., S_i, F_i, and I_i) is a similarity search, and the search results need to be combined.

We use a fuzzy logic approach to combine the rankings of the specifications. Much as in fuzzy set theory, we define a membership function to measure the similarity of the object X to the predicate p, and denote it by $\mu_p[X]$. A membership function value of 1 indicates complete similarity, i.e., equality with respect to the constraints.

Currently, there are several fuzzy frameworks. Both of the following definitions for the fuzzy intersection (AND) operation have been adopted in this investigation:

$$\mu_{A\ AND\ B}[X] = \forall_{i=0}^{N} \min(\mu_A[x_i], \mu_B[x_i]), \qquad (1)$$

$$\mu_{query} = \sum_{i=1}^{N} \mu_{exp_i}[x_i] w_i \qquad (2)$$

where w_i is the relative weight of the ith predicate of the query phrase.

4 Experimental Study

We have implemented the *MALM* framework described in the previous section in matlab and C++. The experiments are run on a stock price database consisting of 144 long sequences extracted from the stock price of 36 large companies between 1962 and 1984. Each sequence contains 1413 points (approximately 6 years of data). We note that this is considered to be a stress case as various studies [10, 15] have found that the stock price changes comport to Brownian motion or random walk.

The sequence database is segmented using a divide-and-conquer technique similar to that described in [17]. For a sequence x_i of length T, the line connecting x_1 and x_T is $x_i = ai+b$ where $x_1 = a+b$ and $x_T = aT+b$. Consequently, it is straightforward to show that $a = (x_T - x_1)/(T-1)$ and $b = x_1 - a$. The maximum deviation between the line and the time series, tol_{max} can then be computed: $tol_{max} = max_{i=1}^{T}\{abs(x_i - ai - b)\}$. If this tolerance is greater than a preset maximum, the location where their maximum deviation occurs is chosen as a break point, and a straight line is then constructed to test whether all the points within the segment are within the preset tolerance. This procedure can be continued recursively until all the points within each segment are within a predetermined tolerance from the line constructed from the end points.

Figure 3 (a) shows the original time series of the IBM stock from 1968 to 1973 and the segmented result using a segmentation tolerance (or threshold) of 0.01. Here the segmented sequence appears to overlap almost perfectly with the original one. In order to avoid over-fragmentation, a merge operation can be applied to the segmented subsequences to merge adjacent segments that are shorter than a fixed minimum length. The result is shown in Fig. 3 (b). Using a higher threshold at 0.05 can also reduce the fragmentation, as shown in Fig. 3 (c). The segmented result of using a threshold of 0.08 is shown in Fig. 3 (d). Different segmentation tolerance not only determines the number of segments that will be produced from each sequence, but also affects the statistical distribution of the extracted features.

The features extracted from each sequence, $\{x_i\}$, include:

- *Regression slope*: A linear regression model is applied to each segment where the regression coefficients, the slope a_R and the intercept b_R, are given by $a_R = \frac{\sum_{i=1}^{T}(x_i-\overline{x})(i-(T+1)/2)}{\sum_{i=1}^{T}(i-(T+1)/2)^2}$ $b_R = \overline{x} - a_R(T+1)/2$.
- *Sum of error*: The error between the ith element of the time series and the value predicted by the regression model is $e_i = x_i - a_R i - b_R$. Several features can be derived based on the error between the original time series and the regression model. The most obvious one is the total sum of errors, defined as $ErrorSum = \sum_{i=1}^{T}|e_i|$.
- *Mean square error*: Another feature is the mean square error, defined as below: $MSE = \frac{1}{T}\sum_{i=1}^{T} e_i^2$. This feature measures how well does the regression model coincide with the time series.
- *Segment length*: the length of the segment can be a useful feature in discriminating a long-term vs. a short term trend when they have the same slope.

Feature vectors are thus extracted from each segmented subsequence. The tolerance for segmenting the time series

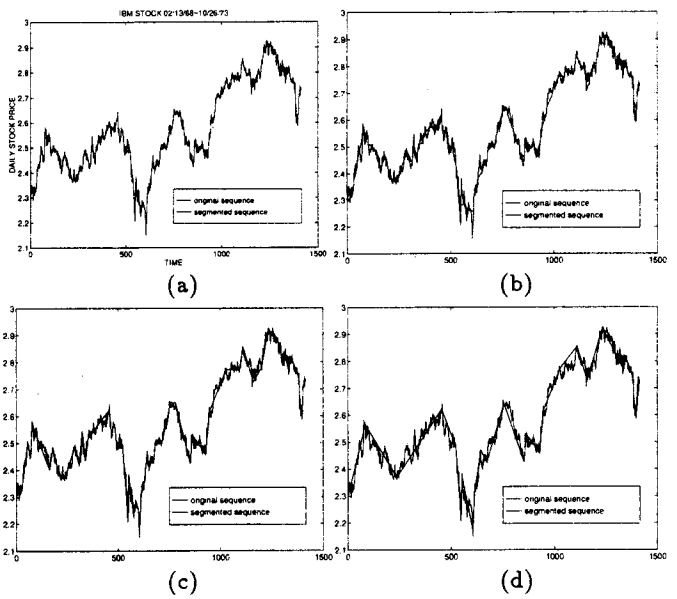

Figure 3: Time series sample of IBM stocks from 1968 to 1973 with tolerance 0.01, before merging (a) and after merging short adjacent segments with tolerance (b) 0.01 (c) 0.05, (d) 0.08.

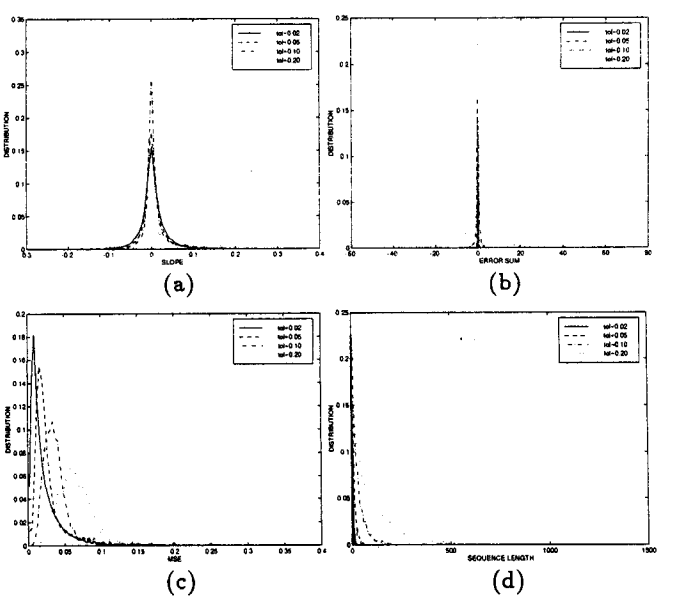

Figure 4: (a) Slope distribution of the segmented subsequences, (b) Distribution of error sum of the segmented subsequences, (c) Mean square error distribution of the segmented subsequences, (d) Length distribution of the segmented subsequences.

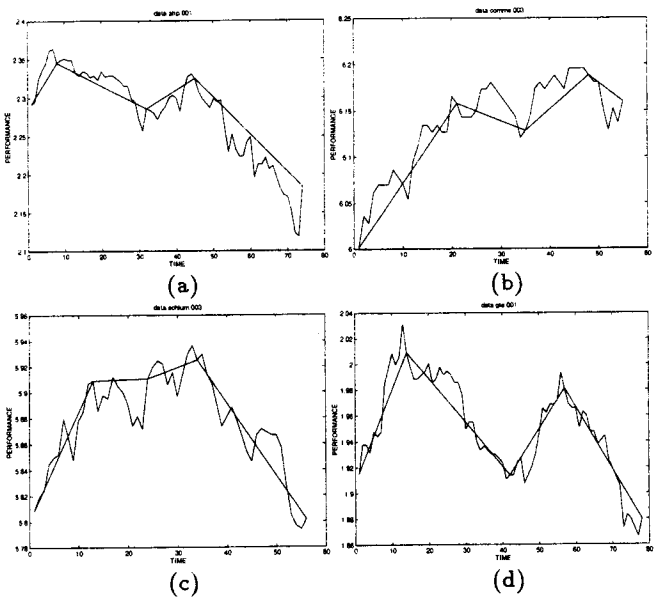

Figure 5: Best 4 matches to the subsequences from AHP using weight (1.0,0.0,0.0,0,0) and Eq. (2) for fuzzyAND.

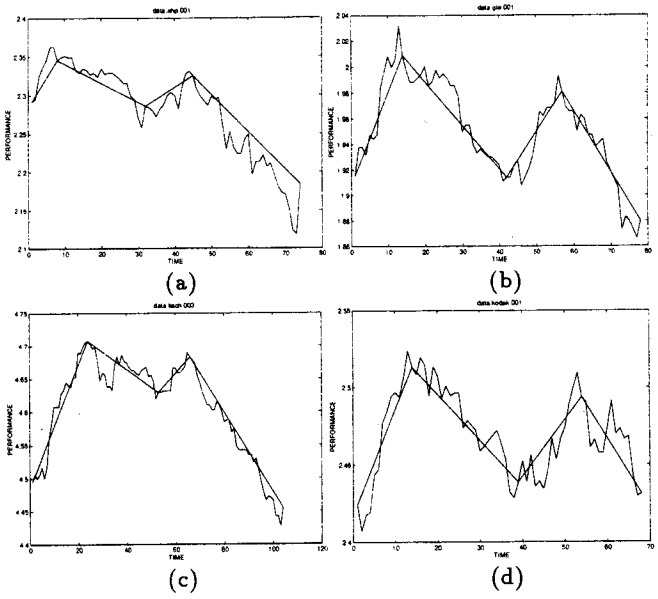

Figure 6: Best 4 matches to the subsequences from AHP using weight (1.0,0.0,0.0,0,0) and Eq. (3) for fuzzyAND.

affects the distribution of the feature, as it is evident from Fig. 4. A total of 50115, 17654, 6156, and 1785 segments are produced from the 144 time series, with tolerance level set at 0.02, 0.05, 0.10, and 0.20 respectively.

Fig. 4(a) shows the distribution of the regression slope for tolerance levels set at 0.02, 0.05, 0.10, and 0.20 respectively. On most occasions, both positive and negative slopes are generated simultaneously when each subsequence is broken up into two. Consequently, the distribution of the regression slope is bell shaped, nearly symmetrical and centered around zero. A larger tolerance produces fewer segments and a larger standard deviation. The distributions of the error sum for various tolerance levels are shown in Fig. 4(b). The distributions are also bell shaped, nearly symmetrical, and centered around zero. This indicates that the probability of causing positive error and negative error from applying linear regression to approximate each time series segment is identical. However, larger tolerance levels produce larger error spread. The distributions of the mean square errors for various tolerance levels are shown in Fig. 4(c), which is approximately Chi-square distributed. As with the regression slope distribution, a larger tolerance produces a larger mean and wider distribution. The distributions of the subsequence length for various tolerance levels are shown in Fig. 4(d). The distributions of the subsequence length are nearly exponential. Larger tolerance levels give rise to larger mean.

Figures 5 and 6 shows the best 4 matches to the same query subsequence which was extracted from the AHP (American Home Product) time series. The AHP subsequence is segmented into four segments using the divide-and-conquer algorithm described earlier. The effect of using different mechanism for combining the similarity score for a subsequence consisting of multiple segments is compared in Fig. 5 (based on Eq. 1) and Fig. 6 (based on Eq. 2). The first similarity measure computes the overall similarity measure of a multisegment time series by using the measure from the worst matched segment. In contrast, the second similarity measure computes the weighted sum of each individual similarity measure of a multisegment time series. The retrieved result sets are completely different. In both cases, only the slope feature is used for the retrieval, as indicated in the weight vector (1.0, 0.0, 0.0, 0.0).

The feature vectors are then clustered into 25 clusters (0-24) using two-dimensional Kohonen self-organizing map [12], a well known neural network technique for unsupervised learning.

Figure 7 shows the results of searching at the semantic level. In the first example, the query is to search for time series that are similar to the performance of GULF (Gulfwest oil) between two specific dates (shown in Fig. 7(a)). The sequence GULF is segmented using the same divide-and-conquer algorithm that is used for segmenting the whole database. In this case, the sequence is divided into four segments, with symbols 3-2-3-8. The results are displayed in Fig. 7(b)-(d).

In spite of the recent efforts (such as those by [7]) in defining a better similarity measure for time series, the purpose of this paper is not just to retrieve visually similar subsequences. As shown in previous figures, the retrieved subsequences are all similar to some degrees. The contribution of this framework is to provide an interactive and flexible framework to those domain experts, who ought to be the ultimate judges of the similarity. This framework also allows the experts to dynamically adjust the weights of different features and similarity measures in order to retrieve those subsequences that met a certain objectives.

5 Summary and Discussion

In this paper, we propose an object-oriented framework, MALM, where the similarity search can be performed on representation of time series at multiple abstraction levels. In the proposed approach, the time series in the databases

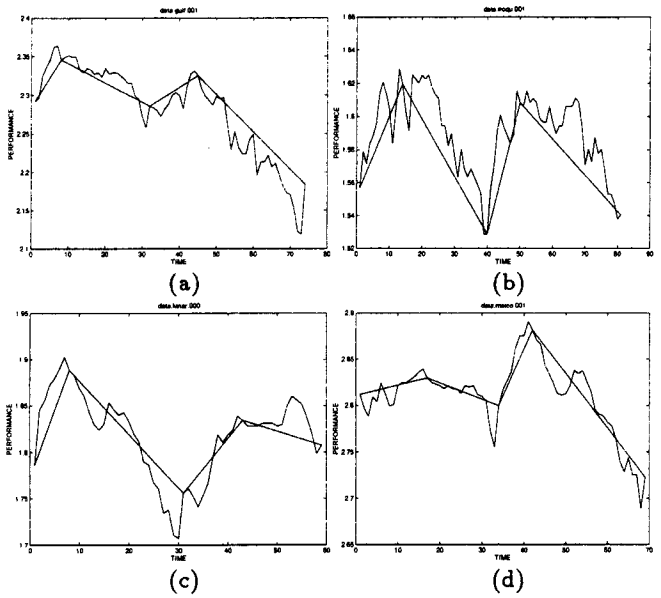

Figure 7: Search results of subsequences with symbols 3/2/3/8.

are first segmented with a divide-and-conquer algorithm. Features such as regression coefficients, mean square error, and higher-order statistics of the histogram of the regression residuals are extracted from each segment. Labels are then assigned to each segment with a neural network clustering algorithm [12]. The clustered time series segments can then be queried at the symbol level, the feature level, or the sequence level. This framework provides a powerful mechanism allowing the user to generalize the query template for similarity and all-pair searches. The numerical results show that this framework provides the user with a more flexible way to specify query at a desired abstraction level. By controlling the abstraction level, the user can determine the amount of generalization. In other words, the notion of *fast rise* will retrieve more subsequences than slope = 0.05.

References

[1] AGRAWAL, R., FALOUTSOS, C., AND SWAMI, A. Efficient similarity search in sequence database. In *Fourth International Conference on Foundations of Data Organization and Algorithms* (Oct. 1993).

[2] AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering - special issue on Learning and Discovery in Knowledge-Based Databases* (1993).

[3] AGRAWAL, R., LIN, K., SAWHNEY, H. S., AND SHIM, K. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proc. of the 21st Int'l Conference on Very Large Databases* (September 1995).

[4] AGRAWAL, R., PSAILA, G., WIMMERS, E. L., AND ZAIT, M. Querying shapes of histories. In *Proc. VLDB* (1995).

[5] AL-TAHA, K. K., SNODGRASS, R. T., AND SOO, M. D. Bibliography on spatiotemporal databases. *ACM SIGMOD Record 22*, 1 (March 1993), 59–67.

[6] BOLLOBAS, B., DAS, G., AND GUNOPULOS, D. Time-series similarity problems and well-separated geometric sets. In *13th ACM Symposium on Computational Geometry* (1997).

[7] DAS, G., GUNOPULOS, D., AND MANNILA, H. Finding similar time series. In *PKDD'97* (1997).

[8] FALOUTSOS, C., AND LIN, K.-I. Fastmap: A fast algorithm for indexing, data mining, and visualization of traditional and multimedia datasets. In *Proc. SIGMOD'95* (1995), pp. 163–174.

[9] FALOUTSOS, C., RANGANATHAN, M., AND MANOLOPOULOS, Y. Fast subsequence matching in time-series databases. In *Proc. SIGMOD'94* (1994), pp. 419–429.

[10] FAMA, E. F. The behavior of stock market prices. *Journal of Business* (Jan. 1965), 34–105.

[11] KEOGH, E. Fast similarity search in the presence of longitudinal scaling of time series databases. In *Proc. IEEE International Conferences on Tools with Artificial Intelligence* (1997), pp. 578–584.

[12] KOHONEN, T., OJA, E., SIMULA, O., VISA, A., AND KANGAS, J. Engineering applications of the self organizing map. *Proc. of the IEEE 84*, 10 (October 1996), 1358–1384.

[13] LI, C.-S., CASTELLI, V., AND YU, P. S. Hierarchyscan: A hierarchical similarity search algorithm for databases of long sequences. In *Proc. ICDE* (1996).

[14] LU, W., HAN, J., AND OOI, B. Discovery of general knowledge in large spatial databases. In *Proc. Far East Workshop on Geographic Information Systems* (1993), pp. 275–289.

[15] OSBORNE, M. Brownian motion in the stock market. *Operations Research* (March-April 1959).

[16] RAFIEI, D., AND MENDELZON, A. Similarity based queries for time series data. In *SIGMOD* (1997), pp. 13–25.

[17] SHATKAY, H., AND ZDONIK, S. B. Approximate queries and representations for large data sequences. In *Proc. ICDE* (1996).

[18] STAM, R., AND SNODGRASS, R. A bibliography on temporal databases. *IEEE Bulletin on Data Engineering 11*, 4 (Dec. 1988).