Model-Based Similarity Measure in TimeCloud

Thanh-Nguyen Ngo[1], Hoyoung Jeung[2], and Karl Aberer[1]

[1] École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
{thanhnguyen.ngo,karl.aberer}@epfl.ch
[2] SAP Research, Brisbane, Australia
hoyoung.jeung@sap.com

Abstract. This paper presents a new approach to measuring similarity over massive time-series data. Our approach is built on two principles: one is to parallelize the large amount computation using a scalable cloud serving system, called TimeCloud. The another is to benefit from the filter-and-refinement approach for query processing, such that similarity computation is efficiently performed over approximated data at the filter step, and then the following refinement step measures precise similarities for only a small number of candidates resulted from the filtering. To this end, we establish a set of firm theoretical backgrounds, as well as techniques for processing *k*NN queries. Our experimental results suggest that the approach proposed is efficient and scalable.

Keywords: similar measure, time-series, cloud computing.

1 Introduction

As time-series data becomes ubiquitous, the demand for storing and processing massive time-series in the cloud is growing rapidly. To meet this demand, the LSIR laboratory[1] has been developing a storage-and-computing platform for managing large volumes of time-series in the cloud. TimeCloud is established upon a combination of several cloud systems, such as Hadoop [2] and HBase [3], while gearing various novel approaches towards significantly boosting the performance of large-scale data analysis on distributed time-series. One of the novel features in TimeCloud is to employ *model-based views*—which are database views approximating time-series using well-established models—for efficient data processing. In this paper, we present one mechanism that lies in the core of the feature, i.e., similarity measure of distributed time-series managed in the model-based views.

Measuring a similarity is a fundamental operation in a wide range of applications that process temporally ordered data, such as stock prices, sensor readings, trajectories from moving objects, and scientific data. Despite the importance of similarity measure, computing similar time-series over a large volume of data still remains as a difficult problem. Although a rich body of previous studies

[1] http://lsir.epfl.ch

Q.Z. Sheng et al. (Eds.): APWeb 2012, LNCS 7235, pp. 376–387, 2012.

have dealt with efficient computation of time-series [4, 1, 5], their proposals become limited when the data volume grows.

In this paper, we present a very different approach from the existing works. The key differences are twofold: First, we parallelize the computation using multiple nodes (servers), taking advantages of the cloud serving systems Hadooop and HBase. These systems make TimeCloud scale-out, allowing us to deal with huge volumes of time-series data. Second, we apply the well-known *filter-and-refinement approach* [6] for measuring similarities across different nodes. Specifically, we first approximate a given time-series using an either constant or linear model, and then store the approximated data into a model-based view. Given multiple model-based views containing approximated time-series data, we then find a candidate set that potentially satisfies a given query condition, while facilitating very efficient processing over the model-based views. For each candidate, we next measure an accurate similarity using full-precision data to validate the result.

In order to embody the approach, we establish theoretical foundations that can serve as the basis in computing distances between approximated time-series stored in the model-based views. As we deal with two different approximation models, the computation for similarity measure requires all pairs of different model-approximated data. Obviously, this needs very firm, non-trivial foundations, which we present in this paper. Furthermore, we offer details for processing kNN queries while taking the advantage of the filter-and-refinement approach. The beauty of our approach is to guarantee no false miss at query results, as the query processing technique is built on the foundations. In our experimental study, we will analyze the effect of this approach while applying different parameter settings.

The remainder of the paper is organized as follow: Section 2 offers a set of definitions, and establishes the theoretical foundations for the kNN query process presented in Section 3. We then discuss about experimental results in Section 4, and conclude in Section 5.

2 Similarity Measure over Model-Based Views

2.1 Definition

Definition 1 (Time-Series). *A time-series t of length n is a temporally ordered sequence $t = [t_1, \ldots, t_n]$ where point in time i is mapped to a d-dimensional attribute vector $t_i = (t_{i_1}, \ldots, t_{i_d})$ of values t_{i_j} with $j \in \{1, \ldots, d\}$. A time-series is called univariate for $d = 1$ and multivariate for $d > 1$.*

The work relies heavily on transformed models, and the existing system only converts univariate time-series. Therefore, we only consider the univariate time-series in the scope of the paper. If a time-series has multiple attributes, we consider it as multiple univariate time-series. From now on, when mentioning on time-series, we consider it as univariate time-series unless stated otherwise. In addition, we only consider time-series with the same interval.

Definition 2 (Common Points). *Two points of two time-series are called common if they occur at the same time.*

Definition 3 (Common Interval). *The common interval of two segments or two time-series is the greatest interval $[a, b]$ such that time a and b belong to both segments or time series. Two segments limited by the common interval are called common segments.*

With the Definition 3, two time-series may not have common segments if one time-series starts after another time-series ends. Or, the common segments of two time-series are time-series themselves if their starting points and end points are common.

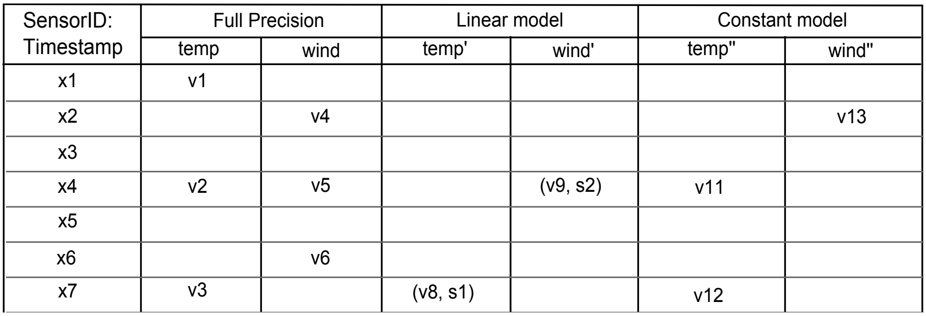
Definition 4 (Euclidean Distance). *The Euclidean distance between two time-series is also the Euclidean distance of their common segments $s = [s_1, \ldots, s_n]$ and $t = [t_1, \ldots, t_n]$ of length n, and it is defined as:*

$$Eucl(s,t) = \sqrt{\sum_{i=1}^{n}(s_i - t_i)^2}$$

We can consider a time-series of length n as an n-dimensional point in space. The value at time t_i is mapped into the i^{th} dimension. So, two common points will be mapped into the same dimension. And when evaluating the Euclidean distance between two points in the space, we only consider dimensions having two points of both time-series.

2.2 Model-Based Views

Since the major component of the back end consists of an HBase instance, the way the data is stored inside HBase becomes of major concern, not only for full precision data but also for the parameters used for model-based approximations. Fig. 1 represents in a schematic way how the data is organized in TimeCloud.



SensorID: Timestamp	Full Precision		Linear model		Constant model	
	temp	wind	temp'	wind'	temp"	wind"
x1	v1					
x2		v4				v13
x3						
x4	v2	v5		(v9, s2)	v11	
x5						
x6		v6				
x7	v3		(v8, s1)		v12	

Fig. 1. A snapshot of a model-based view

2.3 Calculating the Euclidean Distance over Models

Full Precision Model vs. Other Models. To determine the Euclidean distance between a full precision model and an another model, at first, we determine the common interval between these two time-series. Then, we apply the formula in Definition 4 to calculate the distance between every common points of two time-series. Finally, we square root the sum of all square of individual distances to get the Euclidean distance between these two time-series.

Constant Model vs. Constant Model. At first, we divide constant segments of two time-series into common segments. Then we calculate the distance of those common segments and then aggregating them. Since the data does not change within a common segment, the distance of common segments is equal to the distance of two common points multiply by the square root of number of common points in those segments.

When determining the common segments, we know the starting and end time of those segments, and we also know the interval of those time-series. Therefore, we can determine the number of common points of those common segments. In addition, we also know the values of these segments. Hence, we can determine the distance of these segments without aggregating all individual distances of common points.

Linear Model vs. Linear or Constant Model. At first, we evaluate the Euclidean distance of two time-series in linear models. As the implementation on constant models, we devise a similar algorithm to quickly return the Euclidean distance between two common linear segments. Assume the formula representing those segments are $y = ax + b$ and $y = cx + d$. Apply the formula in Definition 4, the square of the Euclidean distance of two common segments s, t with k common points is:

$$\begin{aligned} Eucl^2(s,t) &= \sum_{i=1}^{k}(s_i - t_i)^2 = \sum_{i=1}^{k}((ax_i + b) - (cx_i + d))^2 \\ &= (a-c)\sum_{i=1}^{k} x_i^2 + 2(a-c)(b-d)\sum_{i=1}^{k} x_i + k(b-d)^2 \end{aligned} \tag{1}$$

Let t be the interval of two time-series, so $x_{i+1} = x_i + t$. We have:

$$k = \frac{x_n - x_1}{t} + 1 \tag{2}$$

$$\sum_{i=1}^{k} x_i = \frac{k}{2}(x_1 + x_n) = \frac{x_1 + x_n}{2}(\frac{x_n - x_1}{t} + 1) \tag{3}$$

$$\sum_{i=1}^{k} x_i^2 = \frac{1}{6t}[x_n(x_n + t)(2x_n + t) - x_1(x_1 - t)(2x_1 - t)] \tag{4}$$

Replace (4), (3) and (2) into (1), we have:

$$\begin{aligned}Eucl^2(s,t) =& \frac{a-c}{6t}[x_n(x_n+t)(2x_n+t) - x_1(x_1-t)(2x_1-t)] \\ &+ (a-c)(b-d)(x_1+x_n)(\frac{x_n-x_1}{t}+1) \\ &+ (b-d)^2(\frac{x_n-x_1}{t}+1) \end{aligned} \tag{5}$$

Thus, similar to the implementation on constant models, we divide two time series in linear models into common segments. Then we calculate the square of the distance of common segments. The formula to determine this value only depends on the starting and end time, the coefficients of segments and the interval of those time-series.

A time-series in constant model is a special case of the linear model when the slope is equal to zero. So, we can apply the implementation on two linear models to calculate the distance of a time-series in linear model and a time-series in constant model.

2.4 Maximum Error of the Euclidean Distance of Two Time-Series

Definition 5 (Maximum Error Bound of Time-Series). *Given a time-series $t = [t_1, \ldots, t_n]$ and its representation $t' = [t'_1, \ldots, t'_n]$ in its model. The maximum error bound of t over its model is a value $meb(t)$ such that:*

$$|t_i - t'_i| \leq meb(t), \qquad \forall i = 0..n$$

In general, the value of maximum error bound of a time-series over its model is predefined. Then we construct the model such that it satisfies the formula in Definition 5 and we try to maximize the number of time-series points in a segment. With this approach, the number of segments in the model is minimized, and it is efficient to access and compute.

Definition 6 (Maximum Error Bound of Euclidean Distance). *Given time-series s and t and their representations s', t' in their models. The maximum error bound of the Euclidean distance between s and t over their models is a value $MEB(s,t)$ such that:*

$$|Eucl(s,t) - Eucl(s',t')| \leq MEB(s,t), \qquad \forall s', t'$$

From the Definition 6, the estimated distance between two time series differs from the real distance by an upper bound. Hence, when calculating the Euclidean distance on models, we do not know exactly the distance between two time-series, but we can estimate the range in which the distance belongs to.

Before giving a formula to determine the value of the maximum error bound of the Euclidean distance between two time-series, we need to prove the following lemma.

Lemma 1. *Given two time-series s, t and their representations s', t' in their models. Assume the common segments of s and t have n time series points. Then,*

$$||s_i - t_i| - |s'_i - t'_i|| \leq meb(s) + meb(t),\ \forall i = 1..n$$

Proof. Based on the Definition 5, $\forall i = 1..n$, we have:

$$-\,meb(s) \leq s_i - s'_i \leq meb(s) \tag{6}$$

$$-\,meb(t) \leq t_i - t'_i \leq meb(t) \tag{7}$$

Without loss of generality, assume $s_i \geq t_i$, let (6) - (7):

$$\begin{aligned} &(s_i - t_i) - (s'_i - t'_i) \leq meb(s) + meb(t) \\ &\Rightarrow |s_i - t_i| - |s'_i - t'_i| \leq meb(s) + meb(t) \end{aligned} \tag{8}$$

Because of the equality in the role of t_i and t'_i in Definition 5, we also have:

$$|s'_i - t'_i| - |s_i - t_i| \leq meb(s) + meb(t) \tag{9}$$

From (8) and (9), we have:

$$||s_i - t_i| - |s'_i - t'_i|| \leq meb(s) + meb(t)$$

□

Theorem 1 (MEB(s,t)). *Given two time-series s, t and their representations s', t' in their models. Assume the common segments of s and t have n time series points. Then,*

$$MEB(s,t) = \sqrt{n}(meb(s) + meb(t))$$

Proof. Let $d_i = s_i - t_i$, $d'_i = s'_i - t'_i$, and $m = meb(s) + meb(t)$

From the Lemma 1, we have:

$$\begin{aligned} \sum_{i=1}^{n} {d'}_i^2 &\leq \sum_{i=1}^{n} (d_i + m)^2 \\ &= \sum_{i=1}^{n} d_i^2 + 2m \sum_{i=1}^{n} d_i + nm^2 \end{aligned}$$

Apply the Cauchy-Schwarz inequality $(\sum_{i=1}^{n} d_i)^2 \leq n \sum_{i=1}^{n} d_i^2$, we have:

$$\begin{aligned} \sum_{i=1}^{n} {d'}_i^2 &\leq \sum_{i=1}^{n} d_i^2 + 2m\sqrt{n \sum_{i=1}^{n} d_i^2} + nm^2 \\ &= \left(\sqrt{\sum_{i=1}^{n} d_i^2} + \sqrt{n}m \right)^2 \\ \Rightarrow Eucl(s', t') &\leq Eucl(s,t) + \sqrt{n}(meb(s) + meb(t)) \end{aligned} \tag{10}$$

Similarly, we also have:

$$\sum_{i=1}^{n} d_i^2 \leq \sum_{i=1}^{n} (d_i' + m)^2$$
$$\Rightarrow Eucl(s,t) \leq Eucl(s',t') + \sqrt{n}(meb(s) + meb(t)) \tag{11}$$

From (10) and (11), we have:

$$|Eucl(s,t) - Eucl(s',t')| \leq \sqrt{n}(meb(s) + meb(t)) \tag{12}$$

Apply the Definition 6, we have:

$$MEB(s,t) = \sqrt{n}(meb(s) + meb(t))$$

□

The equality in (12) occurs when common segments of two time-series are parallel and $d_i' = d_i - m, \forall i = 1..n$ or $d_i' = d_i + m, \forall i = 1..n$. This condition may occur in theory, but it does not exist in our implementation, because we try to minized the total distance of raw data and its model.

3 KNN Processing

In this section, we introduce our implementation on solving the similarity measure problem using the filter-and-refinement method. At first, in the filter stage, we calculate the Euclidean distances of all time-series and the query time-series on models, we get the approximate distances and their maximum error bounds. From those values, we build a minimum candidate set which contains the result set. Then, we apply the refinement stage in which we calculate the true Euclidean distance between all time series in the candidate set and the query time-series to return exactly k nearest neighbors of the query time-series.

3.1 The Filter Stage

Given a query time-series q, and a database with n time-series $t_1, \ldots, t_n$, we aim to find k time-series from the database that are closest to q. To this end, we first compute the candidate set which definitely satisfy the given query condition.

Theorem 2. *Let t_i' and q' be representations of t_i and q in their models respectively. Let d_i' be the distance between t_i' and q' with the maximum error e_i. Let $a_i = d_i' - e_i$ and $b_i = d_i' + e_i$. Without loss of generality, assume $b_1 \leq \ldots \leq b_n$. The candidate set $S = \{t_i | a_i \leq b_k\}$ contains k nearest time-series of q and is minimal.*

Proof. First, we prove that S contains k nearest time-series of q. Take a time-series $t_j \notin S$, we need to prove that t_j is not one of k nearest neighbors of q. Since $t_j \notin S$, we have:

$$Eucl(t_j, q) \geq a_j > b_k$$

We also have:

$$Eucl(t_i, q) \leq b_i \leq b_k \qquad \text{with } i \leq k$$

Hence:

$$Eucl(t_j, q) > Eucl(t_i, q) \qquad \text{with } i \leq k$$

Because the true distance from t_j to q is greater than from k other time series, t_j will not belong to the result set.

Now, we prove that the set S is minimal. Let S' be the candidate set that contains k nearest time-series of q and is minimal. If $S \neq S'$, then $S \backslash S' \neq \emptyset$. Take $t_j \in S \backslash S'$.

Consider the following case: $Eucl(t_i, q) = \begin{cases} a_i, & \text{if } i = j \\ b_i, & \text{otherwise} \end{cases}$

$$\begin{aligned} \text{Because } t_j \in S \Rightarrow\ & a_j \leq b_k \leq b_i \text{ with } i \geq k \\ \Rightarrow\ & Eucl(t_j, q) \leq Eucl(t_i, q) \text{ with } i \geq k \\ \Rightarrow\ & t_j \text{ is one of } k \text{ nearest neighbours of } q \\ \Rightarrow\ & S' \text{ does not contain } k \text{ nearest time-series of } q, \\ & \text{contradicts to the assumption} \end{aligned}$$

Therefore, $S = S'$. □

From Theorem 2, we calculate the Euclidean distance of all time-series and the query time-series in their models to retrieve the candidate set. This calculation is much faster than calculating the true distance of all time-series in the database because the time-series in their models have smaller number of segments.

3.2 The Refinement Stage

Given a query time-series q, and a candidate set S with m time-series $t_1, \ldots, t_m$. Our problem is to find k time-series from S that are closest to q.

Theorem 3. *Let t'_i and q' be representations of t_i and q in their models respectively. Let d'_i be the distance between t'_i and q' with the maximum error e_i. Let $a_i = d'_i - e_i$ and $b_i = d'_i + e_i$. Without loss of generality, assume $a_1 \leq \ldots \leq a_m$. The set $R = \{t_i | b_i \leq a_{m-k+1}\}$ is a subset of the result set.*

Proof. Take a time-series $t_j \in S$, we need to prove that t_j is one of k nearest time-series of q. We have:

$$Eucl(t_j, q) \leq b_j \leq a_i \leq Eucl(t_i, q) \qquad \text{with } i \geq m - k + 1$$

Therefore, we cannot find k time-series in S such that their distances to q are strictly smaller than the distance from t_j to q. In addition, the set S contains k nearest time-series of q, so t_j is one of k nearest time-series of q. □

Based on the Theorem 3, at first, we retrieve time-series that definitely belong to the result set to not waste time to calculate the distances between them and the query time-series. Then we use the full precision model to calculate the true distances from the remaining time-series in the candidate set and the query time-series to determine the result set.

4 Experiments

All the experiments were executed on 2.4GHz Intel Core2 Quad CPU running Java implementation on Ubuntu 10.10 and the following parameters are used as default unless stated otherwise: length of time-series $l = 512$, number of nearest neighbors $k = 10$, error ratio $e = 3\%$, and number of time-series in the database $N = 1,000$.

4.1 Model-Based View Construction

At first, we evaluate the reduction of number of entries of time-series in model-based views on different error ratios. The result in Fig. 2 presents the total number of entries of all time-series in the database with respect to their models. It shows that the linear model always has smaller entries than the constant model, and of course, the full precision model is always the largest one. With respect to error ratios, the number of entries on the linear model are 27.55% with $e = 0.55\%$, up to 5.4% with $e = 5\%$ compared to the number of entries of the full precision model. The corresponding values on the constant model are 50.3% and 9.0% respectively.

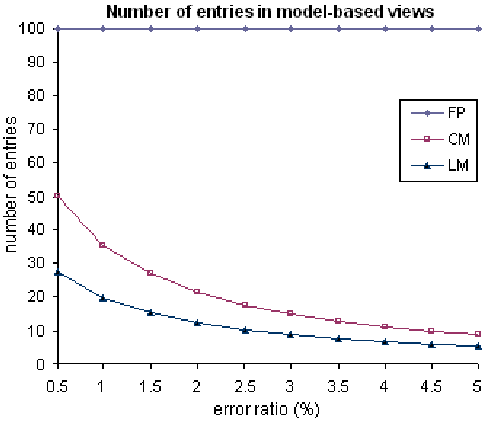


Fig. 2. Number of entries in model-base view on different error ratios

4.2 Effect of Maximum Error Ratios

In this experiment, we evaluate the effect of the maximum error ratio on the query processing time of the similarity measure problem. We evaluate it on three approaches: (1) running on the full precision model without using improvement technique, (2) using the filter-and-refinement method on the constant model, and (3) using the filter-and-refinement method on the linear model.

As depicted in Fig. 3, the linear model is always the fastest model in query processing and then is the constant model. When not using optimization technique, the response is too long. In addition, the experiment shows that the performance peaks when the error ratio is 4% for the linear model and 4.55% for the constant model. The reason is that it takes much time on the filter stage if the error ratio is too small, and it takes much time on the refinement stage if the error ratio is too large. Therefore, choosing the appropriate error ratio is crucial to improve the system performance.

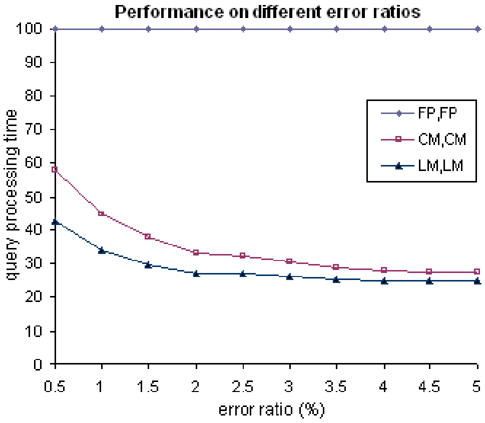


Fig. 3. Effect of the maximum error ratio on the query processing time

4.3 Effect of Number of Nearest Neighbors

In this experiment, we evaluate the effect of the number of nearest neighbors on the query processing time. Similar to the experiment in Sect. 4.2, we evaluate on three models and the result is depicted in Fig. 4.

The figure shows that it takes slightly more time to process the query if we increase the number of nearest neighbors. And this affects both constant and linear models. This is because when we increase the number of nearest neighbors, after the filter stage, the candidate set will be larger, and it takes more time in the refinement stage to calculate the real distance of time-series in the candidate set.

4.4 Effect of Number of Time-Series

In the last experiments, we evaluate the effect of the number of time-series in the database on the query processing time. The result depicted in Fig. 5 shows that the processing time of the converted models decreases when the number of time-series in the database increase. The reason is that the number of time-series in the candidate set does not increase linearly as the size of the number of time-series. Therefore, the refinement stage takes less time when we enlarge the database.

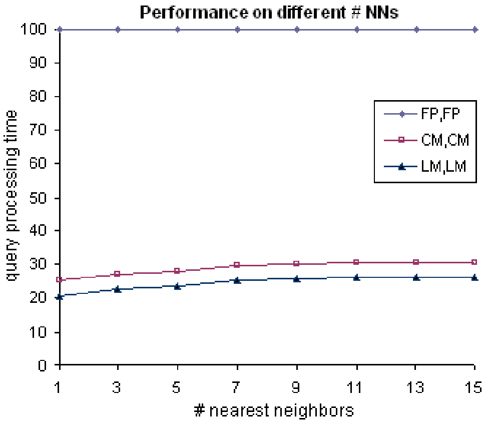


Fig. 4. Effect of the number of nearest neighbors on the query processing time

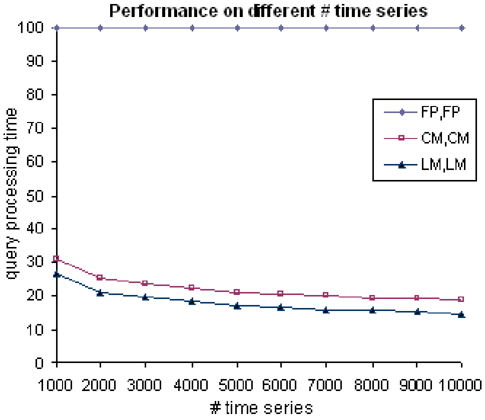


Fig. 5. Effect of the number of time-series on the query processing time

5 Conclusion

In the paper, we provide an efficient approach to processing kNN queries based on model-based similarity measures. To this end, we have established a set of important theoretical foundations for approximated time-series data processing. We then presented our query processing mechanisms built on the filter-and-refinement approach. The experiments showed that our approach runs more than three times faster than straightforward processing, while facilitating scalability of the computation using the TimeCloud system.

Acknowledgement. This work was supported by the European Commission in the PlanetData NoE (contract nr. 257641), the Nano-Tera initiative (http://www.nano-tera.ch) in the OpenSense project (reference nr. 839-401), and NCCR-MICS (http://www.mics.org), a center supported by the Swiss National Science Foundation (grant nr. 5005- 67322).

References

[1] Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient Similarity Search in Sequence Databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993)

[2] Apache. Hadoop. Website, `http://hadoop.apache.org/`

[3] Apache. Hbase. Website, `http://hbase.apache.org/`

[4] Chan, F.K.-P., Fu, A.W.-C., Yu, C.: Haar wavelets for efficient similarity search of time series: with and without time warping. IEEE Trans. on Knowl. and Data Eng. 15, 686–705 (2003)

[5] Korn, F., Jagadish, H.V., Faloutsos, C.: Efficiently supporting ad hoc queries in large datasets of time sequences. In: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, SIGMOD 1997, pp. 289–300. ACM, New York (1997)

[6] Seidl, T., Kriegel, H.-P.: Optimal multi-step k-nearest neighbor search. SIGMOD Rec. 27, 154–165 (1998)