

Optimal Distance Bounds on Time-Series Data

Michail Vlachos[†] Suleyman S. Kozat[‡] Philip S. Yu^{*}

[†]IBM Zürich Research Laboratory, Switzerland

[‡]Koç University, Dept. of Electrical Engineering, Istanbul, Turkey

^{*}University of Illinois at Chicago, Dept. of Computer Science

Abstract

Most data mining operations include an integral search component at their core. For example, the performance of similarity search or classification based on Nearest Neighbors is largely dependent on the underlying compression and distance estimation techniques. As data repositories grow larger, there is an explicit need not only for storing the data in a compressed form, but also for facilitating mining operations *directly* on the compressed data. Naturally, the quality or tightness of the estimated distances on the compressed objects directly affects the search performance.

We motivate our work within the setting of search engine weblog repositories, where keyword demand trends over time are represented and stored as compressed time-series data. Search and analysis over such sequence data has important applications for the search engines, including discovery of important news events, keyword recommendation and efficient keyword-to-advertisement mapping.

We present new mechanisms for very fast search operations over the compressed time-series data, with specific focus on weblog data. An important contribution of this work is the derivation of optimally tight bounds on the Euclidean distance estimation between compressed sequences. Since our methodology is applicable to sequential data in general, the proposed technique is of independent interest. Additionally, our distance estimation strategy is not tied to a specific compression methodology, but can be applied on top of any orthonormal based compression technique (Fourier, Wavelet, PCA, etc). The experimental results indicate that the new optimal bounds lead to a significant improvement in the pruning power of search compared to previous state-of-the-art, in many cases eliminating more than 80% of the candidate search sequences.

1 Introduction

Internet search engines collect vast amounts of data with regards to their usage, which can effectively assist in describing the evolution of user behavior and search preferences over time. The work presented here deals with the compression and distance estimation on time-series data in general,

with specific focus in the efficient search of weblog time-series. The temporal sequences that we consider capture the daily demand of search queries/keywords.

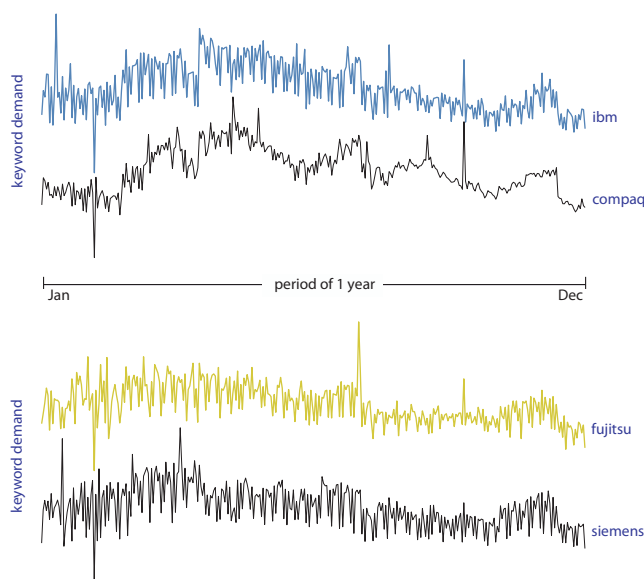


Figure 1. Query demand at a search engine for four keywords, over the period of one year.

Figure 1 depicts four such temporal sequences, where each point of the sequence describes the daily demand at a search engine for a particular keyword. The queries captured in our example are: *IBM*, *Compaq*, *Fujitsu* and *Siemens*. Each sequence contains 365 values, describing the demand pattern for the specific keyword during the period of one year. In the past, similar datasets were generally unavailable to the public, however, nowadays one can search and download such data using websites like [GoogleTrends](http://www.google.com/trends)¹. This temporal representation of a query is useful, because it visually captures important trends in the keyword demand, but it also highlights important semantic

¹<http://www.google.com/trends>

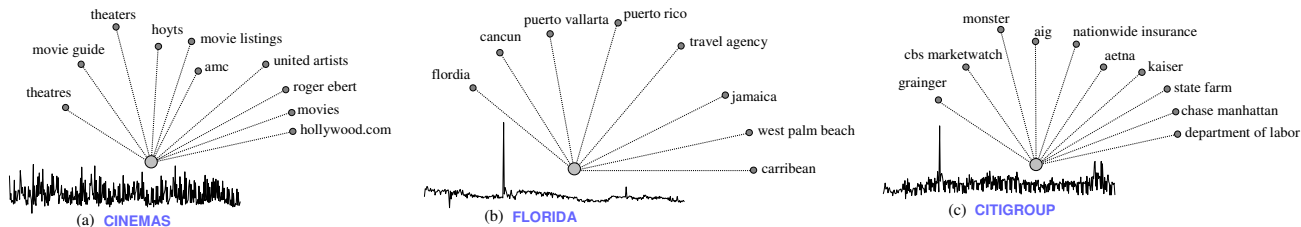


Figure 2. 3 queries and other semantically related keywords, based on the similarity of demand patterns

characteristics. For the aforementioned example, one can notice that the yearly demand for the keyword *IBM* is more similar to the demand for the keyword *Compaq* than to the remaining keywords. This affinity in the demand trends clearly suggests a semantic relation between the specific keywords. Generally speaking, as previous studies note: “user behavior is deeply related to search keyword[s]” [1]. One can distill this behavior, which can prove beneficial in a variety of applications:

(1) *Search engine optimization:* Understanding the semantic similarity between keywords can assist in constructing more accurate keyword taxonomies and achieving better clustering of keywords [2]. This can serve in providing better search results and ultimately help understand the true relationship between web pages. A number of features can assist in this process, such as repetition in the search behavior [3], something that is easily conveyed by the temporal representation of the query demand.

(2) *Keyword recommendation:* Related queries are manifested as similar demand patterns. A search engine can exploit this characteristic for offering a “maybe you would also be interested in this” functionality. As an illustrative example, Figure 2(a) shows some of the queries with similar demand patterns to the keyword ‘cinemas’. All the results are highly interpretable and include queries such as ‘movie guide’, ‘hollywood.com’ and ‘roger ebert’.

(3) *Better spelling correction:* No dictionary or ontology can cover the wide range of keywords that appear on the web. However, relationships between keywords can be deduced by the systematic study of the query logs [4]. Figure 2(b) illustrates an instance of such an example, for the query ‘florida’ and the misspelled keyword ‘flordia’, which exhibits an almost identical demand pattern.

(4) *Identification of news events:* Query logs can help understand and predict behavioral patterns [5]. Important events usually manifest themselves as bursts in the query demand [6, 7]. News travel fast, and web queries travel even faster. By monitoring increasing demands in a query, search engines can accurately pinpoint developing news events.

(5) *Advertising impact:* The financial aspect of search engines is materialized by the carefully selected matching of keywords to advertisements. Semantic clustering of queries can, first, assist the search engine in recommending

related keywords to the advertisers. Secondly, seasonal query demand can help define in a more relevant way the price of an advertisement by elevating the price during times of greater demand for the keyword. This paradigm is similar to the pricing of the TV or radio advertisements, where ‘prime-time’ commercials are valued more highly than the remaining time-slots.

A common denominator in all of the above applications is a set of operations that allow for the effective storage and retrieval of the weblog data. Given the excessive amount of collected data, there is a pragmatic need for effective data compression. Popular search engines like Google, MSN and Yahoo! have data retention periods that lie in the range between 18-30 months ². However, data compression on its own has little to offer if it cannot be combined with a fast search mechanism. This mechanism ideally should be tailored to function over the compressed data.

Since the weblog data exhibit distinct patterns and periodicities, they can be effectively compressed using orthonormal transforms (such as Fourier, wavelets, etc) that effectively capture the energy content of the sequence in just a few components. We will retain the components with the highest energy, in order to accurately and concisely describe the inherent data variability. While this provides an excellent compression technique, comparison between the compressed sequences is difficult since they are described by a (possibly) diverse set of coefficients. In this work we present techniques that overcome this obstacle. A major contribution of this work is a technique for calculating the *optimal* distance bounds that can be derived using the aforementioned compressed representations. The algorithm is based on solid optimization principles and offers a significant boost in the search performance compared to the current state-of-the-art. The technique that we propose here is also of independent interest for general sequence data and is applicable using any orthonormal data transformation.

²<http://googleblog.blogspot.com/2007/06/how-long-should-google-remember.html>

2 Related Work

Previous work considered various applications of temporal sequences on weblogs. [8] examines the discovery of causal relationships across query logs by deploying an event causality test. [9], [10] study similarity search and clustering in query data based on metrics such as correlation and periodicity. While the above utilize linear metrics to quantify the similarity, [5] examines the use of non-linear metrics such as Time-Warping. Finally, in [11] the authors examine a similar application of search on temporal logs, but using click-through data. However, none of the above work examines how to tailor search based on compressed representations of the weblogs. Our work, in that sense, is complementary to all the above approaches, by allowing them to scale up to even larger dataset sizes.

In the data-mining community, search on time-series under the Euclidean metric has been studied extensively [12, 13, 14] but, typically, compression using the first Fourier or wavelets are considered. [7] studies the use of diverse sets of coefficients, but this is the first work that offers the *tightest possible* lower/upper bounds. In the experimental section we offer a thorough performance comparison of our approach across the most predominant methodologies in the time-series literature.

3 Searching temporal log data

We consider a database \mathcal{DB} that stores the temporal weblog sequences $x^{(i)}$, $i = 1 \dots M$. The general problem that we examine can be abstracted as follows: A user is interested in finding the k most similar sequences to a given query sequence q , under a certain distance metric d . This operation is also known as k -Nearest-Neighbor (NN) search. It is a core function in database search and also a fundamental operation in many data-mining and machine-learning algorithms, including classification (NN-classifier), clustering, and so on. Therefore, the provision of such functionality is important for any system that attempts to analyze the data or make useful deductions. The distance function d that we consider in this work is the Euclidean distance. More flexible measures, such as time-invariant distances [15] (essentially a euclidean distance on the periodogram) could also be used with little to no modifications of our main algorithm. However, for ease of exposition here we focus on the Euclidean distance³, which is also the distance measure of preference in most of the related work [9, 10].

In Figure 2 we plot some of the nearest neighbors of 3 queries; ‘cinemas’, ‘florida’ and ‘citigroup’. We observe that the results have a high semantic affinity with the posed query. For example, the query ‘citigroup’ (Fig. 2(c)) returns other financial or insurance companies.

³Note that correlation is also an instance of Euclidean distance on properly normalized sequences.

Generally speaking, search operations can be quite costly, especially in cases where the cardinality of the database sequences is quite extensive and the sequence length is also substantial (both statements are true for our scenario). This is observed, because sequences need to be retrieved from disk in order to be compared to the query q . An effective way to mitigate this cost, is to create a smaller, compressed representation of the sequences, which will be used as an initial pre-filtering step. Therefore, each sequence $x^{(i)}$, will be transformed into some condensed representation $X^{(i)}$. Essentially, one is employing a *multilevel* filtering mechanism. When examining the compressed sequences X , we obviously cannot derive the exact distance between the query q and any sequence $x^{(i)}$ in the database. Underestimates and over-estimates of the distance will be calculated, which in the literature are also known as *lower* and *upper bounds* on the distance function. Using these bounds, a superset of the k -NN answers will be returned, which will be verified against the uncompressed disk-resident sequences. These will be fetched and compared with the query, so that the exact distances can be computed. This methodology is very widely used in the data mining time-series field and it is the methodology also used in this work. The above steps are summarized in Fig. 3.

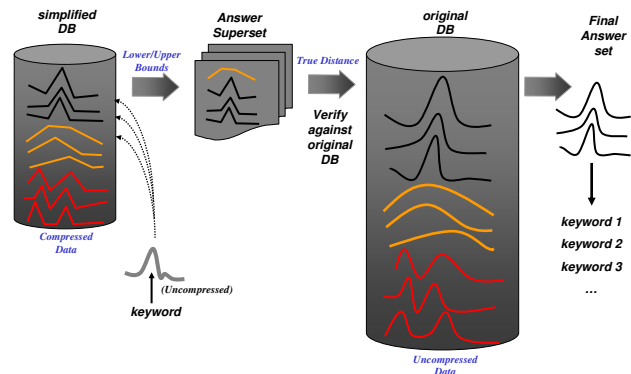


Figure 3. General framework for speeding up Nearest-Neighbor search operations

3.1 Use of upper and lower bounds. Lower/upper bounds on the distance function serve three purposes: (1) Eliminate from examination candidate sequences that are provably worse than the current best match during the search procedure; (2) dictate a search order of the disk-resident sequences, so that more promising candidates are examined first, hence providing at an early stage of the search a good candidate match. This will help eliminate subsequent distance sequences from examination; (3) provide guarantees that the initial data filtering using the compressed data will return the same outcome as when scanning sequentially the original, uncompressed data.

Consider that we are seeking the 1-NN match of the query q . By swiftly scanning the compressed representations lower and upper bounds of q against all sequences in the database can be derived. We extract the minimum upper bounds UB_{min} and all sequences that have lower bound greater than UB_{min} can be safely discarded, since obviously a better match can be found (in the form of the sequence with upper bound equal to UB_{min}). Next, the uncompressed sequences are retrieved from disk in the order suggested by the lower bounds (LB's), since sequences with smaller LB's are more likely to be closer to the query q . The true distance of each sequence to the query is evaluated and the best-so-far match is potentially updated. Once the LB of the currently retrieved sequence is greater than the (true) distance of the best-so-far match, then the search can be terminated, since all the remaining sequences are guaranteed to have greater distance than the best-so-far candidate sequence.

In the general case, where one is searching for the k -Nearest-Neighbors ($k > 1$), the only change introduced in the above process is the introduction of a priority queue that holds the k best results, and the algorithm prunes the search space based on the distance of the k -th best-so-far match.

Many optimized methodologies can be built upon the above procedure to further reduce the search space (e.g. the creation of an index on the compressed features). However, the steps that we described are rudimentary in the majority of search and indexing techniques [16, 17]. Additionally, the aforementioned search procedure constitutes a bias-free approach to evaluating the search performance of a technique, since it does not depend on any implementation details. We utilize such a search procedure in the experimental section, in order to provide an unbiased performance estimator between various lower/upper bounding techniques, since it does not depend on the inherent implementation, but merely relies on the tightness of the derived bounds.

Obviously, techniques that provide tighter bounds will be able to offer better pruning power and enhanced search performance. Later on, we will provide an algorithm that computes the *tightest possible* lower and upper bounds, when utilizing the high-energy coefficients of weblog (and other temporal) sequences. In the upcoming section we describe how this compression is achieved.

3.2 Compressing weblogs. Query demand patterns do not exhibit a random behavior; rather, they have inherent, meaningful and interpretable trends. Looking, for example, at the demand for the query 'Compaq' in Fig. 1, there are underlying low frequency components which describe the long-term seasonal changes, in addition to high frequency components that correspond to the more short-term trends (e.g. weekly demand) and news events (e.g. bursts). Given these data characteristics, the weblog data can be very effectively compressed using widely used orthonormal decompositions (such Fourier, Wavelets, PCA, etc), that can identify

the underlying components/bases, and describe (compress) the data using only those few components.

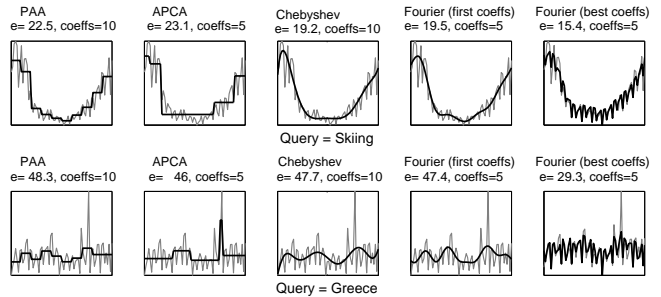


Figure 4. Comparison of various compression techniques on query weblogs. The approximation error e is very low when using the Fourier coefficients with the highest energy.

Figure 4 depicts that much of the data variability is retained even when highly compressing the data. We utilize two query examples and depict their approximation under various compression techniques, such as Piecewise Aggregate Approximation (PAA) [18], Adaptive Piecewise Constant Approximation (APCA) [19] (high energy Haar coefficients), Chebyshev Polynomials [20], first Fourier coefficients [12] and high energy Fourier coefficients ([7]). We observe, that the sequence reconstruction error e is generally lower when using techniques that utilize the highest energy coefficients. This result is not surprising given the observed characteristics of the weblog data. The choice of which transform to use is not crucial to our technique. For the remainder of the paper (and for the experiments), we assume that the Fourier transform will be utilized for compressing the data, since this scenario corresponds to the most elaborate case, because the Fourier coefficients are complex numbers (capturing both magnitude and phase). However, anything that we describe henceforth is applicable on *any* orthonormal transform without modification.

Initially, each weblog sequence $x = \{x_0, x_2, \dots, x_{N-1}\}$ will be represented in the transformed domain by a vector X . In the case of the Fourier transform, X is:

$$X(f_{n/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \quad n = 0, 1 \dots N-1$$

Each compressed sequence X will be described by a set of c coefficients that hold the largest energy. The remaining (low energy) coefficients can be discarded. The cardinality of retained coefficients c can either be a fixed number per sequence (if there are specific space constraints that need to be adhered to), or it can be as many so that $p\%$ of the original sequence energy is retained. Notice that for each sequence we will store a possibly diverse set of coefficients.

In order to record also some information about the discarded coefficients, we will also retain the total energy

of the discarded coefficients: $e_X = \|X^-\|^2$, which is the sum of squares of the omitted coefficients. This quantity represents a measure of the error in the compressed sequence representation. This additional information we will allow us to provide a tighter bound on the distance approximation.

Notation: The vector describing the positions of the high-energy coefficients in X is denoted as p^+ , while the positions of the remaining ones as p^- (that is $p^+, p^- \subset [1, \dots, N]$). So, what we need to store is the vector $X(p^+)$ or equivalently X^+ . In order to simplify notation, if Q is a search query in the transformed domain over all sequences $X^{(i)}$, then $Q(p^+)$ (or Q^+) describes a sequence holding the equivalent coefficients as the vector $X(p^+)$. Similarly, $Q(p^-) \equiv Q^-$ is the vector holding the analogous elements of $X(p^-) \equiv X^-$.

Example: Suppose that a sequence x in the database is transformed using Fourier decomposition and $X = \{(1 + 2i), (2 + 2i), (1 + i), (5 + i)\}$. The magnitude vector of X is: $\|X\| = \{2.23, 2.82, 1.41, 5.09\}$, so if we retain the 2 high-energy coefficients, $p^+ = \{2, 4\}$, then, for a sequence $Q = \{(2 + 2i), (1 + i), (3 + i), (1 + 2i)\}$, $X(p^+) = \{(2 + 2i), (5 + i)\}$ and $Q(p^+) = \{(1 + i), (1 + 2i)\}$.

3.3 Searching compressed weblogs. Considering the above, we have all the elements for describing our problem setting. Given an uncompressed query q , we need to find the closest sequences x in the database based on the Euclidean distance (L_2 -Norm). Parseval's theorem dictates that the Euclidean distance is the same whether computed in the time or in the frequency domain. The preservation of energy holds for any orthonormal transform (wavelets, PCA, etc), so the following derivations are applicable on all those data transforms. The distance can be decomposed as follows:

$$\begin{aligned} D(x, q)^2 &= D(X, Q)^2 \quad (\text{Parseval}) \\ (3.1) \quad &= D(X^+, Q^+)^2 + D(X^-, Q^-)^2 \\ &= \|X^+ - Q^+\|^2 + \|X^- - Q^-\|^2 \end{aligned}$$

Since, X^- is unknown, the exact value of $\|X - Q\|^2$ cannot be calculated. However, the computation of the first part of the distance is trivial since we have all the required data. For the second part we are missing the term X^- , the discarded coefficients. Because we have compressed each sequence X using the best coefficients, we know that the magnitude of each of the coefficients in X^- is less than the smallest magnitude in X^+ . We use $\text{minPower} = \|X_{\text{min}}^+\|$ to denote the magnitude of the smallest coefficient in X^+ .

We can estimate the range of values within which $\|X^- - Q^-\|^2$ lies, by expressing it as an optimization problem, specifically as two optimization sub-problems. As a maximization problem when considering the upper-bound distance, and as a minimization problem when attempting

to establish the lower-bound distance:

$$\begin{aligned} \|X^+ - Q^+\|^2 + \min_{X^-} \|X^- - Q^-\|^2 &\leq \|X - Q\|^2 \text{ and} \\ \|X - Q\|^2 &\leq \|X^+ - Q^+\|^2 + \max_{X^-} \|X^- - Q^-\|^2. \end{aligned}$$

Since X^+ and Q^+ are known, we need to solve the following optimization problems:

$$(3.2) \quad \max_{X^-} \|X^- - Q^-\|^2 \text{ such that}$$

$$(3.3) \quad \|X^-\|^2 = e_X$$

$$(3.4) \quad \|X_i^-\| \leq \text{minPower},$$

and

$$(3.5) \quad \min_{X^-} \|X^- - Q^-\|^2 \text{ such that}$$

$$(3.6) \quad \|X^-\|^2 = e_X$$

$$(3.7) \quad \|X_i^-\| \leq \text{minPower},$$

where X_i^- is the i th component of X^- .

The algorithm that we provide is *optimal*, that is, the bounds that we compute are the tightest possible to the original distance, given that p of the high energy coefficients are stored. To the best of our knowledge, this is the first work that offers such bounds. First, we provide an intuition regarding our solution to the problem, initially on 2-dimensions and then on n -dimensions.

4 Optimal Distance Bounds

4.1 Algorithm Intuition on 2D. We demonstrate the optimal solution with a simple example. For this example we assume that \vec{X} and \vec{Q} are 2-dimensional real vectors. We first find the optimal upper bound and later the optimal lower bound. For the optimal upper bound calculation, $\|Q^+ - X^+\|$ is known and we want to find

$$(4.8) \quad \max_{X^-} \|X^- - Q^-\|^2$$

such that $e_X = \sqrt{(X_1^-)^2 + (X_2^-)^2}$ and $\|X_i^-\| \leq \text{minPower}, i = 1, 2$.

Intuitively, given the query Q^- , the vector which will maximize $\|Q^- - X^-\|^2$ should be on the opposite direction of Q^- , i.e., $X^- = -\alpha Q^-$ for some $\alpha > 0$, as seen in Figure 5(a). Let's also plot on the same figure the two constraints that we have:

1) Notice that the constraint on the total available energy e_X geometrically translates into a circle on the 2D plane (Figure 5(b)). Therefore, the unknown vector X^- should always lie within this circle, otherwise the energy constraints will be violated.

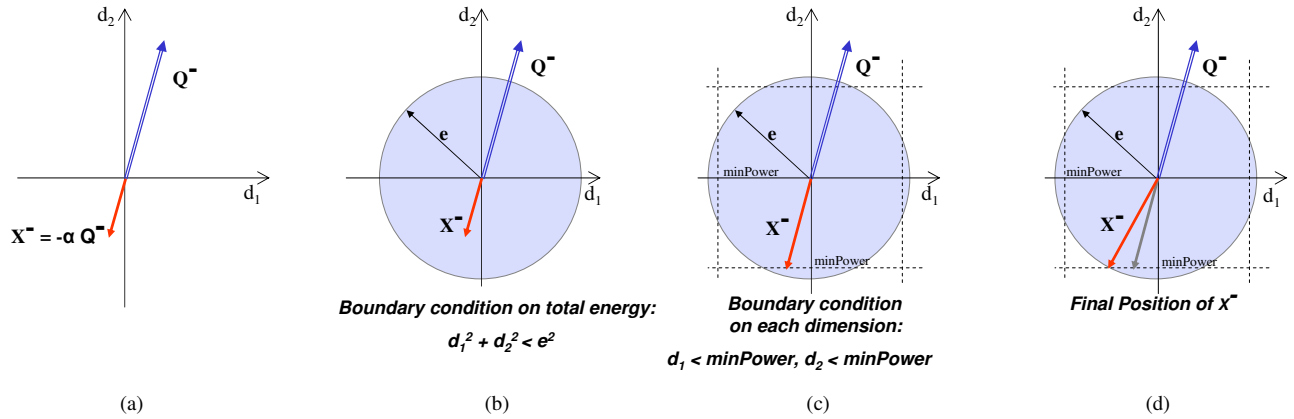


Figure 5. Illustration of the intuition behind our algorithm on 2-dimensions.

2) The constraint on the coefficients of X^- , requiring that each cannot exceed $minPower$, translates into additional constraints indicated by the dotted vertical and horizontal lines at position $minPower$ on the two dimensions/axes, d_1 and d_2 (Figure 5(c)).

The algorithm proceeds as follows; we begin to scale X^- in the opposite direction of the known Q^- by increasing α , so as to maximize the distance $\|Q^- - X^-\|^2$. Now, one of two things may happen. Either we hit on the $minPower$ boundary on one of the axes or we pass the circle indicating the total energy (whichever is violated first). As indicated in Figure 5(c), suppose that we encounter first the boundary condition on one of the axes, e.g., on axis d_2 . Then we keep the corresponding dimension fixed at $\|X_2^-\| = minPower$, i.e. $X_2^- = -minPower$, and only scale the vector X^- on the remaining dimensions until we use all the remaining energy or until we encounter another boundary condition. So, now we start increasing X^- along the d_1 dimension. We can only scale it up to a certain point, because we are faced with the total energy boundary (the circle). At that point, the search stops because all conditions of the maximization problem have been satisfied.

In a similar fashion, if we want to find the *lower bound*, we have to solve:

$$\min_{X^-} \|X^- - Q^-\|^2$$

such that $e_X = \|X^-\|$ and $\|X_i^-\| \leq minPower$, $i = 1, 2$. However, intuitively, given the query Q^- , the vector which will minimize $\|Q^- - X^-\|^2$ should be on the same direction of Q^- , i.e., $X^- = \alpha Q^-$ for some $\alpha > 0$. Since, the boundary conditions are symmetric, if we proceed as the maximization problem, we observe that the vector $-X^{-,*}$ yields the minimizer solution where $X^{-,*}$ is the solution to the maximization problem.

We note, that we don't have to solve the optimization problem twice, but only once, since the two problems are identical.

```

1 // Q is uncompressed
2 // X is compressed + eX is the error
3 // of omitted components
4 //
5 function [LB, UB] = optimal(Q, [X, eX])
6 {
7     p+ = find(X.coeffs); // best coefficients of X
8     minPower = min(abs(X(p+)));
9
10    // distance of known coefficients
11    distSq = sum( abs( X(p+) - Q(p+) ).^2);
12
13    // distance from the remaining coefficients
14    p- = setdiff(1:N, p+); // discarded coefficients
15
16    eY = sum(abs(Q(p-)).^2); // eX is given
17
18    directions = Q ./ norm(Q); // extract direction
19    Xnew = directions .* norm(X); // rescale it
20
21    violations = find(Xnew > minPower);
22
23    // iterate until no energy is left
24    while (~isempty(violations))
25    {
26        Xnew(violations) = minPower; // fix dimensions
27
28        remainingCoeff= find(Xnew < minPower);
29        delta = sqrt(eX - |Xnew| - |remainingCoeff|) *
30            (minPower^2)/sum(directions(dest).^2);
31        Xnew(remainingCoeff) = delta * directions(remainingCoeff);
32
33        violations = find(Xnew > minPower);
34    }
35
36    LB = distSq + sum((Q - Xnew).^2); // updated LB
37    UB = distSq + sum((Q + Xnew).^2); // updated UB
38
39    LB = sqrt(LB);
40    UB = sqrt(UB);
41
42 }

```

Figure 6. An implementation of the solution to the optimization problem.

4.2 Algorithm on n-Dimensions. We now show how the algorithm operates in n -dimensions to allow better exposition of our ideas. We depict the maximization problem.

Figure 7(a) shows the known vector Q^- and the (unknown yet) vector X^- which we attempt to estimate. On the right side of the Figure we also depict a bucket indicat-

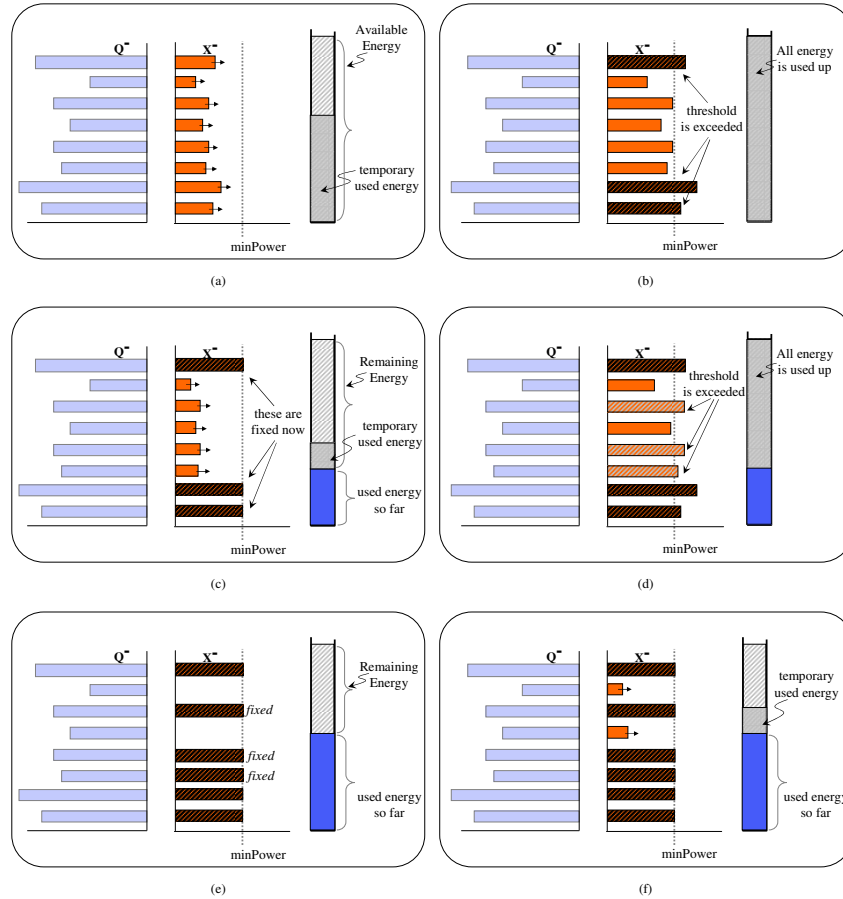


Figure 7. Various steps of the optimization (water-filling) algorithm in n -dimensional space. The unknown vector X^- is rescaled (opposite Q^-) until all energy is used up (a),(b). Coefficients exceeding the given constraints are fixed (c), and the process is repeated for the remaining coefficients until all energy is used up (d),(e),(f).

ing the available energy that we can allocate on X^- . In the previous example, we mentioned that vector X^- needs to be on the opposite direction of vector Q^- , which translates to creating a rescaled vector of Q^- along that direction. X^- is rescaled until all the energy is used up (Figure 7(b)). If certain coefficients exceed the $minPower$ constraint, they are truncated/fixed to $minPower$ (Figure 7(c)). The energy that is allotted for the coefficients that are now fixed is subtracted from the total available energy (Figure 7(c)). For the remaining coefficients we repeat the same process, as shown in Figures 7(d),(e) and (f), until all the available energy is used, or all the unknown coefficients are approximated (fixed). In Figure 6 we provide a pseudocode of the algorithm that we just described.

Lemma: *The configuration described above is a **water-filling** solution and it is guaranteed to provide the tightest possible distance bounds.*

Proof (sketch): Our setting can be formulated as the following set of optimization problems:

$$\begin{aligned} & \min_{X^-} \|X^- - Q^-\|^2 \text{ (or } \max_{X^-} \|X^- - Q^-\|^2 \text{) such that} \\ & \|X^-\| = e_X \\ & \|X_k^-\| \leq X_{min}, \quad k = 1, \dots, m \end{aligned}$$

where the minimization problem discovers the lower bound and the maximization the upper bound. By expansion we get that:

$$\|X^- - Q^-\|^2 = \|X^-\|^2 - 2X^- \cdot Q^- + \|Q^-\|^2$$

therefore we need to maximize (or minimize) only the unknown middle inner product term, $(X^- \cdot Q^-)$. We will only explain how to solve the maximization problem, since the minimization is identical. First, for compactness, we define $r_k = \|X_k^-\|$ and $s_k = \|Q_k^-\|$ for $k = 1, \dots, m$ and the angles between X_k^- and Q_k^- as θ_k . Now, we can rewrite the maximization as:

$$\max_{r_1, \dots, r_m, \theta_1, \dots, \theta_m} e_X^2 - \sum_{k=1}^m r_k s_k \cos(\theta_k) + \|Q^-\|^2$$

such that

$$\sum_{k=1}^m r_k^2 = e_X^2$$

$$0 \leq r_k \leq r, k = 1, \dots, m.$$

by selecting each $\theta_k = \pi$, $\cos(\pi) = -1$, to provide the largest difference, the maximization equation is formed as:

$$\max_{r_1, \dots, r_m} e_X^2 + \sum_{k=1}^m r_k s_k + \|Q^-\|^2$$

To solve the new re-parametrized maximization problem, we form the corresponding Lagrangian.

$$L(\lambda, \beta_1, \dots, \beta_m) =$$

$$(e_X^2 + \sum_{k=1}^m r_k s_k + \|Q^-\|^2) + \lambda(e_X^2 - \sum_{k=1}^m r_k^2) + \sum_{k=1}^m \beta_k(r - r_k),$$

where $\beta_k \geq 0$, $k = 1, \dots, m$. Taking the derivatives of the Lagrangian with respect to unknown variables, λ , β_k , r_k , yields the following Kuhn-Tucker conditions, for $k = 1, \dots, m$

$$(4.9) \quad s_k - 2\lambda r_k - \beta_k = 0,$$

$$(4.10) \quad \beta_k(r - r_k) = 0,$$

$$(4.11) \quad \sum_{k=1}^m r_k^2 = e_X^2$$

where $\beta_k \geq 0$, $k = 1, \dots, m$. Observe that Equation (4.9) can also be written as $r_k = \frac{s_k - \beta_k}{2\lambda}$, $k = 1, \dots, m$, i.e., each r_k is directly proportional to s_k except a bias term β_k . Therefore, the unknown vector X^- needs to be a rescaled version of the known vector Q^- , which directly corresponds to the solution provided by our main algorithm. Hence, the proposed solution satisfies the necessary Kuhn-Tucker conditions, and yields the desired optimal minimizer/maximizer solution.

5 Experiments

We evaluate various parameters of our algorithm; the convergence rate, the tightness of the estimated bounds, and the additional pruning power that is achieved when using the presented optimal algorithm. As our testbed we use search engine logs spanning a period of 3 years (3×365 points per sequence), which we trim down to 1024 points in order to simplify calculations and exposition of ideas. The analyzed data were gathered from a major search engine. The sequences were studentized (mean value was subtracted and sequences normalized by the std), so as to remove any scale

bias. In this way we are reverting the distance into a measurement of correlation and can discover more flexible patterns. Finally, the sequences were compressed using the high energy Fourier coefficients.

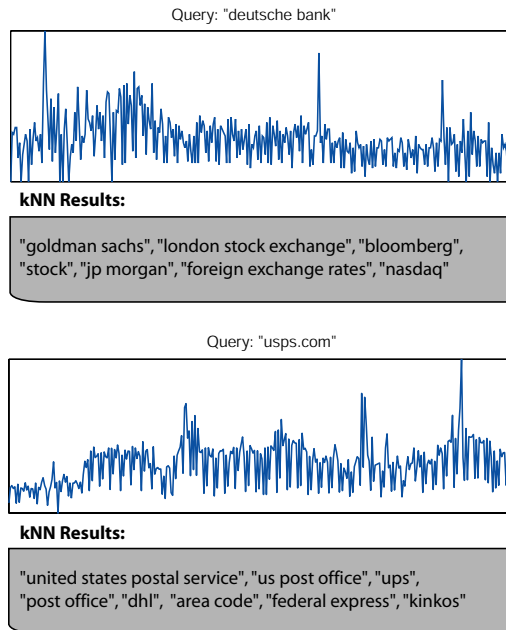


Figure 8. kNN search results on the compressed weblog repositories

Before evaluating any performance aspects of our approach, we depict some of the Nearest-Neighbor (NN) matches that resulted from the search procedure for various queries. Due to space restrictions, in Fig. 8 we illustrate NN-matches for only 2 queries: *deutsche bank*, and *usps.com*. We can observe that the returned matches hold a semantic affinity to the posed query. For example, the outcome of the query *deutsche bank* resulted in keywords relating to financial companies and stocks. In general, search on the query logs returns highly interpretable and useful matches, something that was also attested in other relevant publications [9, 5, 7, 15].

5.1 Convergence Rate. The proposed water-filling algorithm iteratively rescales subsets of the unknown coefficients, in order to utilize the known total signal energy. A number of iterations are required until convergence. Here, we empirically demonstrate that the algorithm reaches the solution in very few iterations (typically 2 to 3), therefore performance of the algorithm is not adversely impacted. The experiment is conducted by computing 1000 distance calculations (lower and upper bounds) from a pool of randomly selected query logs. We repeat the experiment for various compressed representations, retaining from 8 to 64 coefficients per sequence, or in other words for compression rates of $\frac{128}{1}$ to $\frac{16}{1}$. The histograms of the number of iterations

are depicted in Figure 9. We observe that the algorithm converges very fast, typically in 1 to 4 iterations, with the majority of the cases being 2 – 3 iterations.

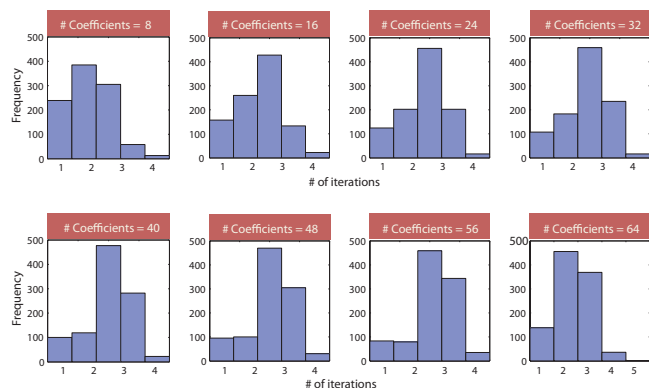


Figure 9. Number of iterations for convergence on the optimization algorithm. The algorithm converges very fast, typically in 2 – 3 iterations.

Notice, that most search operations are I/O bound, and the small additional cost that our algorithm incurs is only CPU-based. However, as we will show, our algorithm achieves much tighter distance bounds which ultimately leads to a great reduction on the uncompressed sequences that are fetched from the disk.

5.2 Bound Tightness. Here, we undertake a systematic comparison and evaluate the tightness of our bounds against previously proposed distance bounding techniques, which have appeared in the data mining literature. The strawmen approaches that we compare with are:

1. **First Coefficients:** Techniques that compute bounds on the distance using the first coefficients, inherently make the assumption that the underlying signal contains primarily low frequency components, such as the approaches proposed by Agrawal et al.[12] and Rafiei et al.[13]. These approaches perform well on random walk signals, such as stock market data, but in general do not adapt well for generic signals. Additionally, such approaches only estimate lower bounds on the distance function, therefore in general cannot match the pruning performance that the combination of lower/upper bounds can achieve.
2. **First Coefficients + error:** This approach augments the aforementioned methodology by recording also the reconstruction error (or remaining energy of the omitted coefficients), which improves upon the previous bounds. This work, presented by Wang et al.[14] additionally utilizes upper bounds, which the previous approaches did not consider.
3. **Best Coefficients + error:** Similar to the previous

approach, this technique exploits the coefficients with the highest energy plus the approximation error in order to bound the distance [7].

We compare the distance bounds returned by these approaches and we juxtapose them with the bounds returned by our methodology which utilizes the highest energy coefficients in conjunction with the optimal distance estimation. However, in order to compare all these approaches we need to properly allocate the storage space for each approach, so as not to favor any of the competing techniques. This is the topic of the next section.

Space Requirements: Notice that it is not meaningful to directly compare the above approaches using the same number of coefficients, because each technique may require a different amount of storage space. We need to compare all approaches under the same memory/space requirements.

The storage of the first c Fourier coefficients requires $2c$ doubles (or $2c * 8$ bytes). However, when utilizing the c best coefficients for each sequence, we also need to store their positions in the original DFT vector. That is, the compressed representation with the c largest coefficients is stored as pairs of $[position-coefficient]$.

For our experiments, the sequences are composed of 1024 points, which means that we need to store 512 positions, if we consider the symmetric property of the Fourier coefficients. 9 bits would be sufficient to describe any of the coefficient positions, however, since on disk we can write only multiples of bytes, recording each position requires 2 bytes. Therefore, each approach that utilizes the best coefficients allocates $16 + 2$ bytes per coefficient. In other words, if an approach storing the first coefficients uses c coefficients, then our method will use $\lfloor 16c/18 \rfloor = \lfloor c/1.125 \rfloor$ coefficients.

First Coeffs	c First Coeffs + Middle Coeff
First Coeffs + error	c First Coeffs + Error
Best Coeffs + error	$\lfloor c/1.125 \rfloor$ Best Coeffs + Error
Optimal	$\lfloor c/1.125 \rfloor$ Best Coeffs + Error

Table 1. Requirements for usage of same storage for each approach

For some distance measures we also use one additional double to record the error (sum of squares of the remaining coefficients). For the measures that don't use the approximation error we need to allocate one additional number and we choose this to be the middle coefficient of the full DFT vector, which is a real number (since we have real data with lengths power of two). If in some cases the middle coefficient happens to be one of the c best ones, then these sequences just use 1 less double than all other approaches. The following table summarizes how the same amount of space is allocated for each compressed sequence of every approach.

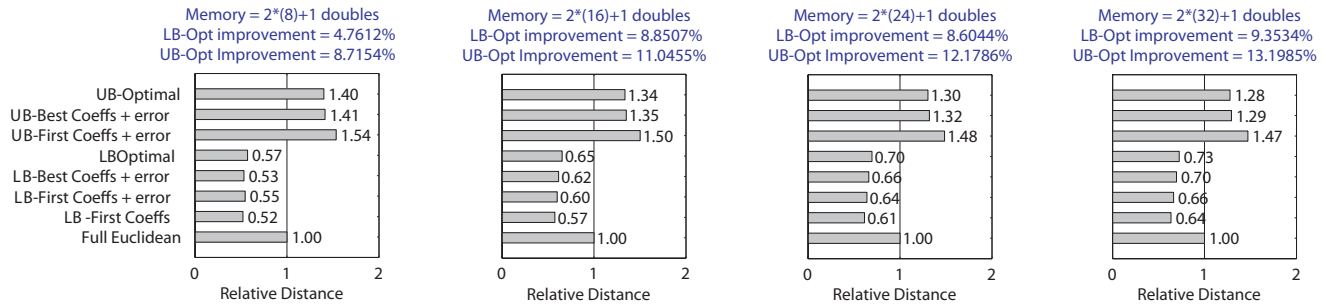


Figure 10. Comparison of lower/upper bounds returned by various techniques, across different compression rates. The Euclidean distance between the uncompressed sequences ('Full Euclidean') is indicated as '1' (one). Estimates closer to '1', suggest better distance bounds. We observe that the presented optimal algorithm exhibits the tightest possible bounds.

Therefore, when in the following figures we mention space usage of $[2*(32)+1]$ doubles, the number in parenthesis essentially denotes the coefficients used for the methods using the first coefficients (+ 1 for the middle coefficient or the error, respectively). For the same example, approaches using the best coefficients will use the 28 best coefficients but have the same space requirements.

Results: We plot the lower and upper bounds derived by each approach and we normalize the results against the exact euclidean distance. Numbers closer to 1 indicate tighter bounds. We observe that in all cases the optimal algorithm returns the best distance estimates compared to the other approaches, even though it uses fewer coefficients than some of the competing methodologies. On the title of each graph of Figure 10 we also indicate how much the optimal algorithm improves on the 'First Coeffs + error' approach. The best improvement is achieved when using 32 coefficients and the improvement reaches approximately 10% on the lower bounds and 13% on the upper bounds. As we will demonstrate in the following section, this reduction in the distance ambiguity can lead to very dramatic speedups in the overall search performance.

5.3 Pruning Power and Performance Improvement.

For this experiment we assemble a large pool of query weblogs consisting of 32000 temporal sequences. We pose 100 random queries that don't have exact matches in order to offer more realistic performance metrics. We search for the 1-Nearest-Neighbor of each query and we utilize both Lower and Upper bounds. For the 'First Coeffs' approach we utilize only the lower-bounds, since no upper-bounds can be computed.

We evaluate the performance of each technique based on the search procedure presented in 3.1, which prunes the search space and directs the search based on the lower/upper bounds derived from the compressed sequences. Ultimately, we measure the amount of uncompressed sequences that each technique retrieved from disk. This essentially reflects the most important bottleneck of a search performance, since

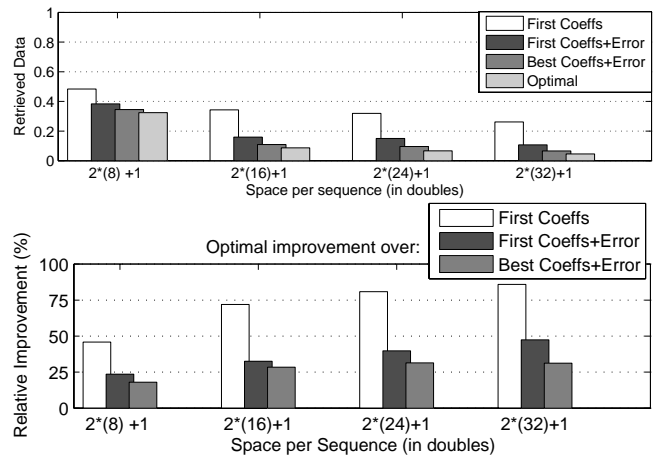


Figure 11. Top: Ratio of uncompressed sequences retrieved from disk (smaller numbers are better). Bottom: Improvement of Optimal against all other approaches (higher numbers are better).

it is an I/O bound process. Figure 11 shows how many uncompressed sequences were retrieved for each search approach, normalized by the total number of sequences. For clarity, at the bottom side of the Figure, the relative improvement that is achieved by the optimal algorithm is also shown. When using $[2*(32)+1]$ doubles per sequence we observe the largest improvement in performance; 80%, 50%, 30% compared to the 3 other distance bounding methodologies. Therefore, we can achieve excellent performance compared to previous state-of-the-art when utilizing the optimal distance bounds.

In conclusion, with these experiments we have seen that the presented optimal distance estimation algorithm converges fast, provides the tightest possible distance bounds, and leads to significant benefits in the search performance.

6 Extending to other distance measures - Discussion

We have seen so far the significant improvement in Euclidean distance estimation that can be accomplished when utilizing the proposed optimal distance bounding technique. The Euclidean distance is probably the most prevalent of the

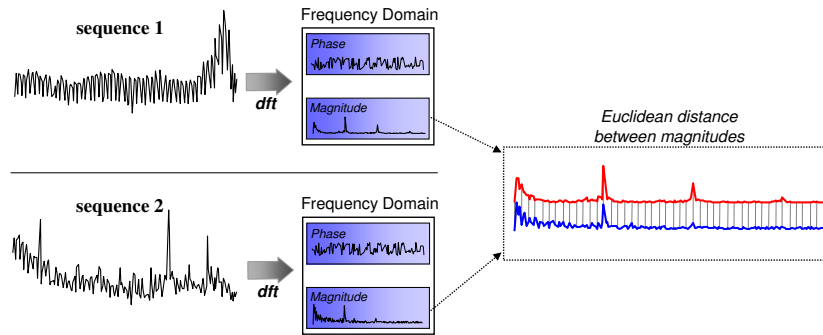


Figure 12. A Periodic Distance can discover pattern similarities under arbitrary time-shift. The Periodic Distance can be reverted into a Euclidean computation of the magnitude components in the frequency domain

distance measures in the mining bibliography [21]. Here we illustrate briefly how our methodology can be applied on a wider range of linear and even non-linear distance functions.

We examine the applicability of the proposed technique under two distance functions; a periodic distance function and the dynamic time warping (DTW). In the following sections we explain in more detail how our bounding technique can be used in conjunction with the aforementioned distance functions, allowing for an even tighter distance bound.

6.1 Bounding the periodic distance measure. Periodic distance measures such as the one presented in [15] can discover similar patterns under arbitrary time-shift. Periodic measures are very flexible, because they allow phase-invariant matching, and have been used with great success for a multitude of difficult mining tasks, including the rotation invariant matching of shapes [22, 23]. They can be used as an inexpensive substitute of DTW when speed is of essence. The periodic distance computes in its core a Euclidean distance in the frequency domain, therefore our bounding technique can be applied directly but in a transformed domain. Its computation is based on a Euclidean distance calculation just in the magnitude components of the Fourier coefficients of a time-series, as shown in Figure 12. Since the technique that we proposed lower bounds the Euclidean distance it can be used unaltered, by application on the magnitude components of the recorded Fourier coefficients. This will effectively compute the tightest bound on the periodic distance, when the original sequences are compressed using the high energy (Fourier or wavelet) coefficients.

6.2 Bounding the dynamic time warping. The Dynamic Time Warping (DTW) distance is a non-linear distance function. However, recent work has shown that the time-constrained version of the DTW can be bounded by a set of linear distance functions [16], and as an extension by our bounding technique. Let's assume that we denote the warping distance under constrained warping within δ time

instances between a query Q and another sequence A as $DTW_{\delta}(Q, A)$. The computation of the tight bound consists in forming the possible matching region (Minimum Bounding Envelope or MBE) and computing the distance between the resulting envelope and sequence A . Therefore, an inexpensive lower bound can be found by computing the euclidean distance between the $MBE(Q)$ and any other sequence A , or $D(MBE_{\delta}(Q), A)$. This is shown in Figure 13.

The bounding envelope of the query and the data sequences are rarely stored in their uncompressed format, but they are approximated by some orthonormal transform. This is where is new bound comes into place, by allowing again the very tight computation between the compressed sequence A (given any orthonormal compression technique) and the MBE of the query.

With the above examples we have only brushed upon the surface of how the proposed optimal bound can be used in conjunction with various distance measures. Other distance/similarity measures can also be expressed in a similar fashion and also similarly amenable for use under our framework. Another similarity function could be for example the correlation between sequences, which can easily be expressed as a function of the Euclidean distance. Finally, even though we have shown that other linear and non-linear distances can be reverted to a Euclidean distance computation, our technique can also be tailored to work under a *variety of linear functions*, since these can be easily formulated and solved under the same optimization umbrella.

7 Conclusions

This work examined techniques that can boost the search performance on compressed time sequences, through introduction of optimally tight distance estimates. We examined the applicability of our technique on temporal query weblogs and we have shown that significant pruning in the search space can be achieved. This is the first work to present optimally tight lower and upper distance bounds, when working

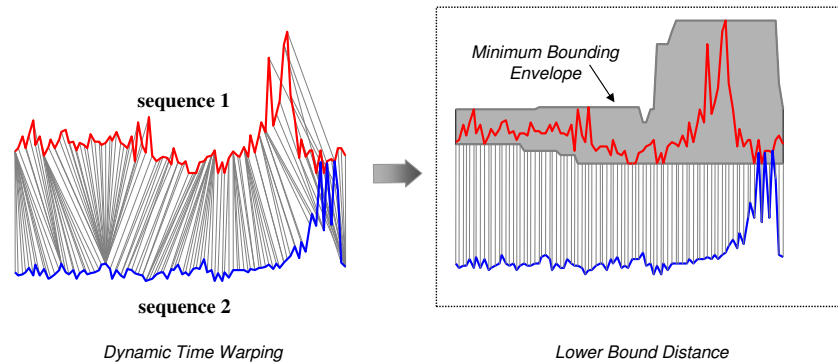


Figure 13. Time Warping distance and computation of lower bound using the Euclidean distance between the Bounding Envelope and any other sequence

directly on the compressed data sequence data.

Although the focus of this work has been on providing optimally tight bounds for the Euclidean distance, we have shown that other linear and non-linear distance functions can be bounded by the Euclidean distance in a transformed domain. Therefore, the outcome of this work can be useful for an extensive class of distance functions. As future work, we plan to evaluate the actual gains that are realized by the application of our technique on a diverse set of distance functions.

Acknowledgements: We would like to thank Claudio Lucchese for useful suggestions on the coding of the optimization problem.

References

- [1] S. Otsuka, M. Toyoda, J. Hirai, and M. Kitsuregawa, "Extracting User Behavior by Web Communities Technology on Global Web Logs," in *Proc. of DEXA*, 2004.
- [2] S. Otsuka and M. Kitsuregawa, "Clustering of Search Engine Keywords Using Access Logs," in *Proc. of DEXA*, 2006.
- [3] M. Sanderson and S. Dumais, "Examining Repetition in User Search Behavior," in *Advances in Information Retrieval*, 2007.
- [4] Q. Chen, M. Li, and M. Zhou, "Improving Query Spelling Correction Using Web Search Results," in *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language*, 2007.
- [5] E. Adar, D. Weld, B. Bershad, and S. Gribble, "Why we Search: Visualizing and Predicting User Behavior," in *Proc. of WWW*, 2007.
- [6] J. Kleinberg, "Bursty and hierarchical structure in streams," in *Proc. of 8th SIGKDD*, 2002.
- [7] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos, "Identification of Similarities, Periodicities & Bursts for Online Search Queries," in *Proc. of SIGMOD*, 2004.
- [8] Y. Sun, N. Liu, K. Xie, S. Yan, B. Zhang, and Z. Chen, "Causal Relation of Queries from Temporal Logs," in *Proc. of WWW*, 2007.
- [9] S. Chien and N. Immorlica, "Semantic similarity between search engine queries using temporal correlation," in *Proc. of WWW*, 2005.
- [10] B. Lie, R. Jones, and K. Klinkner, "Measuring the Meaning in Time Series Clustering of Text Search Queries," in *Proc. of CIKM*, 2005.
- [11] Q. Zhao, S. Hoi, T.-Y. Liu, S. S. Bhowmick, M. R. Lyu, and W.-Y. Ma, "Time-dependent semantic similarity measure of queries using historical click-through data," in *Proc. of WWW*, 2006, pp. 543–552.
- [12] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases," in *Proc. of FODO*, 1993.
- [13] D. Rafiei and A. Mendelzon, "Efficient retrieval of similar time sequences using dft," in *Proc. of FODO*, 1998.
- [14] C. Wang and X. S. Wang, "Multilevel filtering for high dimensional nearest neighbor search," in *ACM SIGMOD Workshop on Data Mining and Knowledge Discovery*, 2000.
- [15] M. Vlachos, P. Yu, and V. Castelli, "On Periodicity Detection and Structural Periodic Similarity," in *Proc. of SDM*, 2005.
- [16] E. Keogh, "Exact indexing of dynamic time warping," in *Proc. of VLDB*, 2002.
- [17] R. Weber, H.-J. Schek, and S. Blott, "A Quantitative Analysis and Performance Study for Similarity Search Methods in High-Dimensional Spaces," in *Proc. of the VLDB, NYC*, 1998, pp. 194–205.
- [18] B. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary Lp norms," in *Proc. of 26th International Conference on Very Large Databases*, 2000.
- [19] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," in *Proc. of ACM SIGMOD*, 2001, pp. 151–162.
- [20] Y. Cai and R. Ng, "Indexing spatio-temporal trajectories with chebyshev polynomials," in *Proc. of ACM SIGMOD*, 2004.
- [21] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," in *Proc. of KDD*, 2002.
- [22] M. Vlachos, Z. Vagena, P. S. Yu, and V. Athitsos, "Rotation invariant indexing of shapes and line drawings," in *Proc. of CIKM*, 2005, pp. 131–138.
- [23] E. J. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos, "Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures," in *Proc. of VLDB*, 2006, pp. 882–893.