Parameter-Free Extended Edit Distance

Muhammad Marwan Muhammad Fuad

Forskningsparken 3, Institutt for kjemi, NorStruct
Universitetet i Tromsø
NO-9037 Tromsø, Norway
mfu008@post.uit.no

Abstract: The edit distance is the most famous distance to compute the similarity between two strings of characters. The main drawback of the edit distance is that it is based on local procedures which reflect only a local view of similarity. To remedy this problem we presented in a previous work the extended edit distance, which adds a global view of similarity between two strings. However, the extended edit distance includes a parameter whose computation requires a long training time. In this paper we present a new extension of the edit distance which is parameter-free. We compare the performance of the new extension to that of the extended edit distance and we show how they both perform very similarly.

Keywords: Edit Distance, Extended Edit Distance, Parameter-Free Extended Edit Distance.

1 Introduction

Let U be a universe of objects. The similarity search problem is the process of finding and retrieving the objects in U that are similar to a given object q; the query. This problem comes in two flavors; *exact search*, i.e. a query q is given, and the algorithm retrieves the data objects in U that exactly match q, and the other is *approximate search* which is motivated by the fact that many exact similarity search methods are time-consuming, that in some cases the response time becomes unacceptable. Besides, in many applications, the overhead time necessary to achieve exact search is not worth the importance of the results obtained.

There are several types of queries, the most famous of which is *range queries,* which can be defined as: given a query q and a radius r, which represents a *threshold, tolerance*, or *selectivity*. The range query problem can be specified as retrieving all the data objects in U that are within a distance r of q. This can be represented as:

$$Range(q,r) = \{u \in U; d(q,u) \leq r\} \tag{1}$$

Another very important type of queries is the *k-nearest neighbor*. In this kind of queries we look for the most similar, i.e. the closest, object in the database to a given query. In the general case we look for the k most similar objects. Unlike the case with

range queries, the response set here is never empty. Moreover, its size is defined beforehand by the user. Formally, this problem can be defined as:

$$kNN(q) = \{X \subseteq U, |X| = k \quad \wedge \forall u \in X, v \in U - X : d(u,q) \leq d(v,q)\} \tag{2}$$

There are still other types of queries such as the *k- reverse nearest neighbor* and *similarity join*.

At the heart of the similarity search problem is the question of how this similarity can be depicted. One of the models that have been presented to tackle this problem is the *metric model*, which is based on the *distance metric*.

In this paper we present a new distance metric applied to sequential data. This new distance is an extension of the well-known *edit distance*. The particularity of the new distance compared with other extensions of the edit distance that we presented before is that it does not include any parameters, whose computing can be very time consuming, thus the new distance can be applied immediately.

The rest of the paper is organized as follows; the necessary background is presented in Section 2, and the new distance is presented in Section 3 with an analysis of its complexity in Section 4, we validate the new distance in Section 5, and related remarks are presented in Section 6. We conclude in Section 7.

2 Background

Let U be a collection of objects. A function d $d : U \times U \rightarrow \mathsf{R}^+ \bigcup \{0\}$ is called a distance metric if the following holds:

(p1) $d(x,y) \geq 0$ (non-negativity)

(p2) $d(x,y) = d(y,x)$ (symmetry)

(p3) $x = y \Leftrightarrow d(x,y) = 0$ (identity)

(p4) $d(x,z) \leq d(x,y) + d(y,z)$ (triangle inequality)

$\forall x, y, z \in U$. We call (U, d) a metric space. □

Of the distance metric properties, the triangle inequality is the key property for pruning the search space when processing queries [10].

Search in metric spaces has many advantages, the most famous of which is that a single indexing structure can be applied to several kinds of queries and data types that are so different in nature. This is mainly important in establishing unifying models for the search problem that are independent of the data type. This makes metric spaces a solid structure that is able to deal with several data types [12].

In metric spaces the only operation that can be performed on data objects is computing the distance between any two objects, which enables us to determine the relative location of the data objects to one another. This is different from the case of

vector spaces; a special case of metric spaces, where data objects have k real-valued coordinates which makes it possible to perform operations that can not be performed in general metric spaces, like addition or subtraction, for instance. Vectors have certain geometric properties that can be exploited to construct indexing structures, but these properties can not be extended to general metric spaces [1].

A *string* is an ordered set of an alphabet Σ . Strings appear in a variety of domains in computer science and bioinformatics. The main distance used to compare two strings is the edit distance [11], also called the *Levenshtein distance* [3], which is defined as the minimum number of delete, insert, and substitute operations needed to transform string *S* into string *R*.

Formally, the edit distance is defined as follows: Let Σ be a finite alphabet, and let Σ^* be the set of strings on Σ. Given two strings $S = s_1 s_2 ....s_n$ and $R = r_1 r_2 ....r_m$ defined on Σ^*. An *elementary edit operation* is defined as a pair: $(a,b) \neq (\lambda,\lambda)$, where a and b are strings of lengths 0 and 1, respectively. The elementary edit operation is usually denoted $a \to b$ and the three elementary edit operations are $a \to \lambda$ (deletion) $\lambda \to b$ (insertion) and $a \to b$ (substitution). Those three operations can be weighted by a weighting function γ which assigns a nonnegative value to each of these operations. This function can be extended to edit transformations $T = T_1 T_2 ...T_m$.

The edit distance between S and R can then be defined as:

$$ED\ (S,\ R) = \{\gamma\ (T) |\ T \text{ is an edit transformation of } S \text{ into } R\ \} \qquad (3)$$

ED is the main distance measure used to compare two strings and it is widely used in many applications. Fig. 1 shows the edit distance between the two strings $S_1 = \{M, A, R, W, A, N\}$ and $S_2 = \{F, U, A, D\}$

ED has a few drawbacks; the first is that it is a measure of local similarities in which matches between substrings are highly dependent on their positions in the strings. In fact, the edit distance is based on local procedures both in the way it is defined and also in the algorithms used to compute it.

		M	A	R	W	A	N
	0	1	2	3	4	5	6
F	1	1	2	3	4	5	6
U	2	2	2	3	4	5	6
A	3	3	2	3	4	4	5
D	4	4	3	3	4	5	5

Fig. 1. The edit distance between two strings.

Another drawback is that ED does not consider the length of the two strings.

Several modifications have been proposed to improve ED. One of them is the *extended edit distance* [7] [8], which adds a global level of similarity to that of ED.

The Extended Edit Distance: The extended edit distance (EED) is defined as follows:

Let Σ be a finite alphabet, and let Σ^* be the set of strings on Σ. Let $f_a^{(S)}$, $f_a^{(R)}$ be the frequency of the character *a* in *S* and *R*, respectively, where *S*, *R* are two strings in Σ^*. EED is defined as;

$$EED(S,R) = ED(S,R) + \lambda\left[|S| + |R| - 2\sum_{a\in\Sigma} min\left(f_a^{(S)}, f_a^{(R)}\right)\right] \tag{4}$$

Where $|S|, |T|$ are the lengths of the two strings *S*, *R* respectively, and where $\lambda \geq 0$ ($\lambda \in \mathsf{R}^+$). We call λ the co-occurrence frequency factor.

EED is based on the principle that the ED distance does not take into account whether the change operation used a character that is more "familiar" to the two strings or not, because ED considers a local level of similarity only, while EED adds to this local level of similarity a global one, which makes EED more intuitive as we showed in [7] [8].

We also presented other modifications of the edit distance based on the frequencies of *n*-grams [5], [6], [9] but they all include parameters whose computing can be very time consuming. In this paper we try to introduce a new extension of the edit distance which is parameter-free, thus can be applied directly.

3 The Parameter-Free Extended Edit Distance

3.1 The Number of Discrete Characters

Given two strings S, T. The number of distinct characters *NDC* is defined as:

$$NDC(S,T) = \left|\{ch(S)\} \cup \{ch(T)\}\right| \tag{5}$$

where *ch* ( ) is the set of characters that a string contains.

The significance of *NDC* is related to the change operation; one of the three atomic operations that the edit distance is based on. Instead of predefining a cost function for all the change operations between any two characters in the alphabet, *NDC* can make the distance, by itself, detect if the change operations use characters that are familiar or unfamiliar to the two strings concerned.

3.2 The Proposed Distance

The objective of our work is to introduce a new distance that adds new features to the edit distance to make it detect global similarity, all this is done by keeping the new distance metric, which is the same objective as that of EED, but unlike EED, our new distance is parameter-free, which makes it more generic.

Definition: The Parameter-Free Extended Edit Distance (PFEED) is defined as:

$$PFEED\ (S,T) = ED(S,T) + \left(1 - \frac{2\sum_{i} min\left(f_i^{(S)}, f_i^{(T)}\right)}{|S| + |T|} \right) \tag{6}$$

Where $|S|, |T|$ are the lengths of the two strings S,T, respectively.

3.3 Theorem

PFEED is a metric distance □

3. 3. 1. Proof **:** Before we prove the theorem, we can easily notice that:

$$\left(1 - \frac{2\sum_{i} min(f_i^{(S)}, f_i^{(T)})}{|S| + |T|} \right) \geq 0 \qquad \forall S,T \tag{7}$$

In order to prove the theorem we have to prove that:

i) $PFEED(S,T) = 0 \Leftrightarrow S = T$

i. a) $PFEED(S,T) = 0 \Rightarrow S = T$

-Proof:

If $PFEED(S,T) = 0$, and taking into account (7), we get the following relations:

$$\left(1 - \frac{2\sum_{i} min(f_i^{(S)}, f_i^{(T)})}{|S| + |T|} \right) = 0 \tag{8}$$

$$ED(S,T) = 0 \tag{9}$$

From (9), and since ED is metric we get: S=T .

i. b) $S = T \Rightarrow PFEED(S,T) = 0$ (obvious).

From i. a) and i. b) we get $PFEED(S,T) = 0 \Leftrightarrow S = T$

ii) $PFEED(S,T) = PFEED(T,S)$ (obvious).

iii) $PFEED(S,T) \le PFEED(S,R) + PEEED(R,T)$

-Proof: $\forall S, T, R$, we have:

$$ED(S,T) \le ED(S,R) + ED(R,T) \tag{10}$$

(Valid since ED is metric).

We also have:

$$\left(1 - \frac{2\sum_i \min(f_i^{(S)}, f_i^{(T)})}{|S|+|T|}\right) \le \left(1 - \frac{2\sum_i \min(f_i^{(S)}, f_i^{(R)})}{|S|+|R|}\right) + \left(1 - \frac{2\sum_i \min(f_i^{(R)}, f_i^{(T)})}{|R|+|T|}\right) \tag{11}$$

(See the appendix for a proof of (11))

Adding (10), (11) side to side we get: $PFEED(S,T) \le PFEED(S,R) + PFEED(R,T)$.

From i), ii), and iii) we conclude that the theorem is valid.

4 Complexity Analysis

The time complexity of PFEED is $O(m \times n)$, where *m* is the length of the first string and *n* is the length of the second string, or $O(n^2)$ if the two strings are of the same lengths. This is the same complexity as that of ED and EED.

5 Experiments

The objective of our experiments is to compare PFEED with EED and show that they have similar performance but one, PFEED, can be applied directly while the other needs a long training time.

A *time series TS* is an ordered collection of observations at different time points. Time series data mining handles several tasks such as classification, clustering, similarity search, motif discovery, anomaly detection, and others. Time series are

high-dimensional data so they are usually processed by using representation methods that are used to extract features from these data and project them on lower-dimensional spaces.

The *Symbolic Aggregate approXimation* method (SAX) [4] is one of the most important representation methods of time series. SAX is applied as follows:

1-The time series are normalized.

2-The dimensionality of the time series is reduced using a dimensionality reduction technique called *piecewise aggregate approximation* (PAA).

3-The PAA representation of the time series is discretized by determining the number and location of the breakpoints. Their locations are determined using Gaussian lookup tables. The interval between two successive breakpoints is assigned to a symbol of the alphabet, and each segment of PAA that lies within that interval is discretized by that symbol.

The last step of SAX is using the following similarity measure:

$$MINDIST\left(\hat{S},\hat{R}\right)\equiv\sqrt{\frac{n}{N}}\sqrt{\sum_{i=1}^{N}\left(dist\left(\hat{s}_i,\ \hat{r}_i\right)\right)^2} \tag{12}$$

Where n is the length of the original time series, N is the length of the strings. $\hat{S}$ and $\hat{R}$ are the symbolic representations of the two time series S and R, respectively, and where the function $dist(\ )$ is implemented by using the appropriate lookup table. After reaching this last step, SAX converts the resulting strings into numeric values so that the MINDIST can be calculated.

Since EED is applied to strings of characters, in [7] and [8] EED was tested on symbolically represented time series using SAX. In this paper we proceed in the same manner and compare PFEED on symbolically represented time series using SAX as a representation method.

EED has two main drawbacks; the first is that the parameter λ does not have any semantics, so its choice is completely heuristic. Besides in all the applications in which EED should be applied directly, i.e. there is no training phase, it becomes difficult to choose and justify the numeric value assigned to the parameter λ to calculate EED. The second drawback is that for each dataset we have to train the training datasets for 5 times at least for parameter λ ($\lambda = 0,...,1$ step=0.25, and sometimes we have to go beyond $\lambda = 1$), which takes a long time.

We conducted the experiments on PFEED using the same datasets on which EED was tested. These datasets are available at UCR [2]. We used the same protocol used with EED; we used SAX to get a symbolic representation of the time series, and then we replaced MINDIST with PFEED (or EED, when testing EED). We also used the same compression ratio that was used to test EED (i.e. 1:4) and the same range of alphabet size [3,10]. The experiments are a time series classification task based on the

Table 1: A comparison of the classification error between PFEED and EED on different datasets.

Dataset	Distance	
	PFEED	EED
Synthetic Control	0.030 α* =7	0.037 α =7, λ=0
Gun-Point	0.067 α =4	0.060 α =4, λ =0.25
CBF	0.010 α =6	0.026 α =3, λ =0.27
Face (all)	0.323 α =5	0.324 α =7=, λ =0
OSULeaf	0.310 α =5	0.293 α =5, λ =0.75
50words	0.270 α =7	0.266 α =7, λ =0
Adiac	0.650 α =9	0.642 α =9, λ =0.5
Yoga	0.155 α =8	0.155 α =7, λ =0

(*: α is the alphabet size)

first nearest-neighbor (*1*-NN) rule using leaving-one-out cross validation. This means that every time series is compared to the other time series in the dataset. If the *1*-NN does not belong to the same class, the error counter is incremented by 1.

We varied the alphabet size on the training set to get the optimal value of the alphabet size; i.e. the value that minimizes the error rate, and then we utilized this optimal value of the alphabet size on the testing sets. We obtained the results shown in Table. 1.

We see from Table 1 that the performance of PFEED is very similar to that of EED, yet PFEED can be applied directly as it requires no training to get the value of λ, which is not the case with EED.

6 Remarks

1-In the experiments we conducted we had to use time series of equal lengths for comparison reasons only, since SAX can be applied only to strings of equal lengths. But PFEED and EED can both be applied to strings of different lengths

2- We did not conduct experiments for alphabet size=2 because SAX is not applicable in this case, but PFEED and EED.

7 Conclusion

In this paper we presented a new extension of the edit distance; the parameter-free extended edit distance (PFEED) and we compared it to another extension; the extended edit distance (EED). The experiments we conducted show that PFEED gives similar results on a classification task, yet the new distance does not include any parameters, thus can be applied directly, which is not the case with EED.

References

1 Chavez, E., Navarro, G., Baeza-Yates, R. A., and Marroquin, J. L.: Searching in Metric Spaces. ACM Computing Surveys (2001).

2 Keogh, E., Zhu, Q., Hu, B., Hao. Y., Xi, X., Wei, L. & Ratanamahatana, The UCR Time Series Classification/Clustering Homepage: www.cs.ucr.edu/~eamonn/time_series_data/ C. A. (2011).

3 Levenshtein, V. I.. Binary Codes Capable of Correcting Spurious Insertions and Deletions Of Ones. Problems of Information Transmission, 1:8-17. Kluwer Academic Publishers, (1965).

4 Lin, J., Keogh, E., Lonardi, S., Chiu, B. Y.: A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. DMKD 2003: 2-11(2003).

5 Muhammad Fuad, M.M.: ABC-SG: A New Artificial Bee Colony Algorithm-Based Distance of Sequential Data Using Sigma Grams. The Tenth Australasian Data Mining Conference - AusDM 2012, Sydney, Australia, 5-7 December, 2012. Published in the CRPIT Series-Volume 134 (2012).

6 Muhammad Fuad, M.M.: Towards Normalizing the Edit Distance Using a Genetic Algorithms–Based Scheme. The 8th International Conference on Advanced Data Mining and Applications -ADMA2012, 15-18 December 2012, Nanjing, China. Published by Springer-Verlag in Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence, Volume 7713 (2012).

7 Muhammad Fuad, M.M, Marteau , P.F. : Extending the Edit Distance Using Frequencies of Common Characters. 19th International Conference on Database and Expert Systems Applications - DEXA'08, Lecture Notes in Computer Science, 2008, Volume 5181/2008, 1-5 September,2008, Turin, Italy (2008).

8 Muhammad Fuad, M.M., Marteau, P.F.: The Extended Edit Distance Metric. Sixth International Workshop on Content-Based Multimedia Indexing (CBMI 2008), London, UK, 18-20th June (2008).

9 Muhammad Fuad, M.M., Marteau, P.F.: The Multi-resolution Extended Edit Distance. Third International ICST Conference on Scalable Information Systems, Infoscale 2008 - Vico Equense, Italy, June 4-6 ACM Digital Library (2008).

10 Samet, H. : Foundations of Multidimensional and Metric Data Structures. Elsevier (2006).

11 Wagner, R., A., Fischer, M. J.: The String-to-String Correction Problem. Journal of the Association for Computing Machinery, Vol. 21, No. I, January 1974 (1974).

12 Zezula et al.: Similarity Search - The Metric Space Approach, Springer (2005).

Appendix

We present a brief proof of the theorem presented in Section 3.3

Lemma

Let Σ be a finite alphabet, $f_a^{(S)}$ be the frequency of the character a in S, where $S \in \Sigma^*$. Then $\forall S_1, S_2, S_3$ we have:

$$\left(1 - \frac{2\sum_i \min(f_i^{(S_1)}, f_i^{(S_2)})}{|S_1| + |S_2|}\right) \leq \left(1 - \frac{2\sum_i \min(f_i^{(S_1)}, f_i^{(S_3)})}{|S_1| + |S_3|}\right) + \left(1 - \frac{2\sum_i \min(f_i^{(S_3)}, f_i^{(S_2)})}{|S_3| + |S_2|}\right) \quad \text{(A1)}$$

For all n, where n is the number of characters used to represent the strings

Proof

i) $n = 1$, this is a trivial case, where the strings are represented with one character .

Given three strings S_1, S_2, S_3 represented by the same character a .
Let $f_a^{(S_1)}, f_a^{(S_2)}, f_a^{(S_3)}$ be the frequency of a in S_1, S_2, S_3, respectively. We have six configurations in this case:

1) $f_a^{(S_1)} \leq f_a^{(S_2)} \leq f_a^{(S_3)}$
2) $f_a^{(S_1)} \leq f_a^{(S_3)} \leq f_a^{(S_2)}$
3) $f_a^{(S_2)} \leq f_a^{(S_1)} \leq f_a^{(S_3)}$
4) $f_a^{(S_2)} \leq f_a^{(S_3)} \leq f_a^{(S_1)}$
5) $f_a^{(S_3)} \leq f_a^{(S_1)} \leq f_a^{(S_2)}$
6) $f_a^{(S_3)} \leq f_a^{(S_2)} \leq f_a^{(S_1)}$

We will prove that relation (A1) holds in these six configurations.

1) $f_a^{(S_1)} \leq f_a^{(S_2)} \leq f_a^{(S_3)}$

In this case we have:

$$min\left(f_a^{(S_1)}, f_a^{(S_2)}\right) = f_a^{(S_1)}, min\left(f_a^{(S_1)}, f_a^{(S_3)}\right) = f_a^{(S_1)}, min\left(f_a^{(S_2)}, f_a^{(S_3)}\right) = f_a^{(S_2)}$$

By substituting the above values in (A1) we get:

$$1 - \frac{2\,min\left(f_a^{(S_1)}, f_a^{(S_2)}\right)}{f_a^{(S_1)} + f_a^{(S_2)}} \leq 1 - \frac{2\,min\left(f_a^{(S_1)}, f_a^{(S_3)}\right)}{f_a^{(S_1)} + f_a^{(S_3)}} + 1 - \frac{2\,min\left(f_a^{(S_3)}, f_a^{(S_2)}\right)}{f_a^{(S_3)} + f_a^{(S_2)}}$$

$$1-\frac{2f_a^{(S_1)}}{f_a^{(S_1)}+f_a^{(S_2)}}\le 1-\frac{2f_a^{(S_1)}}{f_a^{(S_1)}+f_a^{(S_3)}}+1-\frac{2f_a^{(S_2)}}{f_a^{(S_3)}+f_a^{(S_2)}}$$

$$\frac{2f_a^{(S_1)}}{f_a^{(S_1)}+f_a^{(S_2)}}\ge \frac{2f_a^{(S_1)}}{f_a^{(S_1)}+f_a^{(S_3)}}+\frac{2f_a^{(S_2)}}{f_a^{(S_3)}+f_a^{(S_2)}}-1$$

If we substitute $f_a^{(S_2)}, f_a^{(S_1)}, f_a^{(S_2)}$ with $f_a^{(S_1)}, f_a^{(S_3)}, f_a^{(S_3)}$, respectively in the denominators of the last relation it still holds according to the stipulation of this configuration. We get:

$$\frac{2f_a^{(S_1)}}{f_a^{(S_1)}+f_a^{(S_1)}}\ge \frac{2f_a^{(S_1)}}{f_a^{(S_3)}+f_a^{(S_3)}}+\frac{2f_a^{(S_2)}}{f_a^{(S_3)}+f_a^{(S_3)}}-1$$

$$\frac{2f_a^{(S_1)}}{2f_a^{(S_1)}}\ge \frac{2f_a^{(S_1)}}{2f_a^{(S_3)}}+\frac{2f_a^{(S_2)}}{2f_a^{(S_3)}}-1$$

$$1\ge \frac{f_a^{(S_1)}+f_a^{(S_2)}}{f_a^{(S_3)}}-1$$

$$2f_a^{(S_3)}\ge f_a^{(S_1)}+f_a^{(S_2)}$$

This is valid according to the stipulation of this configuration.

The proofs of cases 2), 3), 4), 5) and 6) are similar to that of case 1).

From 1)-6) we conclude that the lemma is valid for $n=1$

ii) $n>1$

This is a generalization of the case where $n=1$.

$\forall i\in n$, then

$$\left(1-\frac{2\,min\left(f_i^{(S_1)},f_i^{(S_2)}\right)}{|S_1|+|S_2|}\right)\le\left(1-\frac{2\,min\left(f_i^{(S_1)},f_i^{(S_3)}\right)}{|S_1|+|S_3|}\right)+\left(1-\frac{2\,min\left(f_i^{(S_3)},f_i^{(S_2)}\right)}{|S_3|+|S_2|}\right)$$

holds, according to the first case i)

By summing over n we get

$$\left(1-\frac{2\sum_i min\left(f_i^{(S_1)},f_i^{(S_2)}\right)}{|S_1|+|S_2|}\right)\leq\left(1-\frac{2\sum_i min\left(f_i^{(S_1)},f_i^{(S_3)}\right)}{|S_1|+|S_3|}\right)+\left(1-\frac{2\sum_i min\left(f_i^{(S_3)},f_i^{(S_2)}\right)}{|S_3|+|S_2|}\right)$$