A Novel Similarity Measure for Sequence Data

Mohammad. H. Pandi*, Omid Kashefi* and BehrouzMinaei*

Abstract—A variety of different metrics has been introduced to measure the similarity of two given sequences. These widely used metrics are ranging from spell correctors and categorizers to new sequence mining applications. Different metrics consider different aspects of sequences, but the essence of any sequence is extracted from the ordering of its elements. In this paper, we propose a novel sequence similarity measure that is based on all ordered pairs of one sequence and where a Hasse diagram is built in the other sequence. In contrast with existing approaches, the idea behind the proposed sequence similarity metric is to extract all ordering features to capture sequence properties. We designed a clustering problem to evaluate our sequence similarity metric. Experimental results showed the superiority of our proposed sequence similarity metric in maximizing the purity of clustering compared to metrics such as d2, Smith-Waterman, Levenshtein, and Needleman-Wunsch. The limitation of those methods originates from some neglected sequence features, which are considered in our proposed sequence similarity metric.

Keywords—Sequence Data, Similarity Measure, Sequence Mining

1. INTRODUCTION

A sequence is an ordered list of objects; in a more formal definition a sequence of length m over the element T is an ordered list such as when $S = s_1 ... s_m$ where each s_i is a member of T and is called an element of S [1]. Sequences of data play important roles in scientific, medical, security, business, and other fields [1]. Biological sequences such as DNA and protein sequences, event sequences such as business transactions and web surfing logs, and letter strings such as words and documents are in the form of sequence data [2]. Measuring sequence similarity is an integral part of many applications for things such as identifying the characteristic (motif) patterns of families of DNA, RNA or protein sequences, comparing sequence families associated with different species/diseases in biology, classification and clustering users and customers in business marketing, and approximate matching and document retrieval in the domain of text analysis [1].

The problem of measuring the similarity of sequences has been studied for years. Sequence features could be considered from different perspectives. The main difference of various similarity measures is originated from a different set of features that are considered. Sequence similarity measures could be character (alignment) based, feature based, information theoretic based, or conditional probability distribution based [1].

Sequence similarity metrics are usually introduced for specific domains and are focused on sequence features of those domains [3]. Unfortunately, the main feature of sequence data that is

Manuscript received January 5, 2011; first revision April 19, 2011; accepted June 21, 2011.
Corresponding Author: Mohammad. H. Pandi
* School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran (pandi@comp.iust.ac.ir, kashefi@{ieee.org, iust.ac.ir}, b_minaei@iust.ac.ir)

the order of sequence's symbols are less considered. As an example, the Smith-Waterman metric [4], as a common local alignment method, detects and emphasizes segments of a sequence, which is simply items appearing right after each other. However, the order of distant symbols (i.e., symbols having one or more symbols appear in between) are not captured and emphasized.

Therefore, in this paper we proposed a new sequence similarity metric that focuses on the symbol-ordering feature of sequences. Our proposed metric extracts all the symbols' orders of a sequence and compares them with the symbol ordering of another sequence by utilizing a specific weighting strategy. We evaluate the effectiveness of proposed sequence similarity metric in a clustering problem compared to some of the most common similarity metrics such as d2 [5], Smith-Waterman [4], Levenshtein [6], and Needleman-Wunsch [7]. Evaluation results show the superiority of our proposed metric compared to other evaluated metrics in the clustering of users from their web usage log.

The remainder of this paper is organized as follow: Section 2 surveys the literature and presents some background on sequence similarity problems; this section also includes some notes on the most common similarity metrics too. Our proposed sequence similarity metric is described in Section 3 accompanied with analysis of the time complexity of it. In Section 4 we evaluate our proposed sequence similarity metric by comparing it with some popular similarity metrics through a clustering problem. Finally, Section 5 concludes the paper and suggests some possible feature tasks for improving the work.

2. RELATED WORKS

There are a variety of methods used in different applications for finding the similarity between two or more sequences. In this section, we give a brief introduction to existing metrics and denote how our proposed sequence similarity metric is related to them.

2.1 Edit-Based Approaches

Edit distance approaches are well-studied and widely used in textual string sequences. Edit distance approaches focus on the lexical level and consider the insertion and deletion of a letter, the substitution of two letters, and the transposition of two adjacent letters as a unit of difference between two string sequences [6]. Extensions of edit distance approaches devise to exploit some domain specific issues similar to keyboard layout [8] and phonetics [9, 10]. Levenshtein distance [6] is one of the most popular sequence similarity metric, which is widely used as the baseline of other edit-based similarity metrics. Mitton [11] proposed a combination of an edit distance and letter match method. Considering the length of sequences, Yujin et al. [12] proposed a normalized Levenshtein distance metric [6]. Bilenko et al. [13] presented a learnable edit distance metric that is based on a support vector machine (SVM).

2.2 Statistical Approaches

Statistical (and probabilistic) approaches are other widely used approaches in applications such as document retrieval and record linkage. Frequency-based edit distance [14], Soft TFIDF [3] and Jaccard similarity [15] could be categorized as statistical approaches. Jiong et al. [2] present a similarity metric based on conditional probability distribution to cluster the sequence

data based on structural features. N-gram is another widely used statistical approach. This approach takes the order of symbols into consideration and is exploited in a variety of methods such as d2 [5]. However, it only considers *N* consecutive symbols but not all of existing orders in a sequence. Hodge et al. [16] proposed a combination of an n-gram model, hamming distance [17], and phonetic coding as a framework to recommend alternative spellings for a misspelled sequence.

Approximate matching [18], record linkage, and duplicate detection [19] are other applications of sequence similarity measure. Statistical metrics such as Monge-Elkan [20], Jaro [21, 22] and Jaro-Winkler [23] are used in record linkage communities. Bilenko et al. [13] present a framework for improving duplicate detection using trainable measures of textual similarity. Zobel et al. [24] propose a method to match name entities with the same phonetics but different spellings. In information retrieval domains, another statistical approach called the language modeling approach [25] is used to model a document as a sequence of terms generated by a probabilistic model. When receiving a query, the similarity of that query with each document could be measured by calculating the probability of generating that query using each document generator model. Another widely used measure in IR is, cosine similarity, which maps the text string into a vector by a statistical weighting scheme [26]. It then measures the similarity of two vectors by finding the cosine of the angle between them.

2.3 Alignment Approaches

Alignment approach is another approach used, especially in biological sequence mining. Needleman-Wunsch [7] was one of the first global alignment algorithms used as a biological sequence similarity measure. Later, based on a dynamic programming approach, local alignment algorithms such as Smith-Waterman [4], BLAST [27], FASTA [28], and YASS [29] were introduced. The studies also include new implementation strategies to improve performance issues [30]. Using programs such as FASTA and BLAST, one can compare unknown Bio-sequences such as a protein sequence, with a set of known sequences to detect functional similarities.

Some of the metrics mentioned above are similar to our proposed sequence similarity metric to some extent. For instance, the N-gram approach can employ a sequence similarity metric as it considers the *N* consecutive symbols, but some of the existing orders in a sequence are neglected in this approach. Alignment approach can be more related to our proposed metric from the perspective of exploiting symbol orders. However, alignment approaches emphasize the same symbols of two sequences only in the same distance from each other. The nature of these approaches is more similar to matching symbols of two sequences but not extracting all ordering features from the sequences, which is the key idea of our proposed sequence similarity metric. The following section explains our proposed sequence similarity metric in detail.

3. OUR PROPOSED SEQUENCE SIMILARITY METRIC

The existing sequence similarity metrics mentioned earlier do not extract all ordering features in a sequence. Suppose $S_1 = abcd$, if we consider S_1 as a presentation of a totally ordered relation in the set of $\{a, b, c, d\}$, then the following features can be extracted in Table 1.

The first feature in Table 1 expresses the existence of symbol *b* after symbol *a*, where the position index of two symbols in S_1 is different by 1. The representativeness of a feature in this

Table 1. Extracted features from S_1

Feature	Distance
a ≤ b	1
b ≤ c	1
c ≤ d	1
a ≤ c	2
b ≤ d	2
a ≤ d	3

extraction scheme has an inverse relationship with the distance of that feature. To make it clearer, consider the sequence S_2 ="*fundamental operational structure of a computer system*." Table 2 shows the features extracted from S_2.

To calculate the similarity of sequence S_2, which is called "referring sequence to sequence and S_1, which is called a "referred sequence," we use two special symbols *"("* and *")"* to represent the start and end of sequences. With the first, we keep common symbols between referred sequence and referring sequence, and remove the other symbols. Reformatting two sequences may create some holes between symbols. We replace these holes with integer numbers indicating the distance between two consecutive symbols. Suppose S_1 = *"efahecfeagyhbffaebyxcgediy"* and S_2 = *"jmmandopbmqnaoqbjdnnamdocq"*. Extracting common symbols, adding *"("* and*")"* to the beginning and the end of a sequence, and specifying distances between remaining symbols makes S_1 = *(3a3c3a4b3a2b3c3d3)* and S_2 = *(4a2d3b4a3b2d3a2d2c2)*.

In the next step, the distance information of a referred sequence is extracted. This step results in a table (see Table 3 for S_1) containing the shortest distance between any two given symbols in the referred sequence. Figure 1 contains the pseudo code of extracting distance information by using the dynamic programming approach.

Table 3 shows the distance information for S_1. For example the number in location *(b, d)* shows the nearest distance of *d* appearing after *b*, which is 6 in S_1. A location will be assigned to -1 when no case with that condition is found. The next step is to build a valid Hasse diagram [31] from the referring sequence where edges are added in respect of the referred sequence. At the beginning, each symbol of the referring sequence is added as a node of a Hasse diagram. Figure 2 (a) shows this step for S_2.

Table 2. Extracted features from S_2

Feature	Difference
fundamental ≤ operational	1
operational ≤ structure	1
structure ≤ computer	1
computer ≤ system	1
fundamental ≤ structure	2
operational ≤ computer	2
structure ≤ system	2
fundamental ≤ computer	3
operational ≤ system	3
fundamental ≤ system	4

```
1:    Procedure Extract Distance Information: Input(S, SYMBOLS): Output(Matrix M)
2:    csi ← 0    //current symbol index
3:    Seen: a list of symbols indices in SYMBOLS
4:    Initialize a square matrix with symbols.size rows to -1
5:    while (csi < Symbols.size )
6:    {
7:    i ← csi
8:    j ← index of next symbol of S in SYMBOLS
9:    weight ← next character of sequence    //contains a length
10:   M[i,j].first ← weight
11:   seen ← seen + i
12:   if (M[csi,j].second = -1) then M[i,j].second ← weight
13:   else M[i,j].second ← Min(M[i,j].second, weight)
14:   if (i - 1 > -1)
15:   for (k : 0 → SYMBOLS.size)
16:   {
17:   if (k is in seen & k ≠ i) then M[k,j].first ← wight + M[k,i-1].first
18:   if (M[k,j].second = -1) then M[k,j].second ← M[k,j].first
19:   }
20:   else M[k,j].second ← Min(M[k,j].second , M[k,j].first)
21:   csi ← j+1
22:   }
23:   return M
```

Fig. 1. Pseudo code for extracting the distance information

Table 3. Distance information of S_1

	a	c	b	d	)
(	3	6	13	24	27
a	6	3	2	8	11
c	3	15	7	3	6
b	3	3	5	6	9
d	-1	-1	-1	-1	3

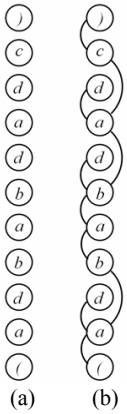

Fig. 2. Building the Hasse Diagram, (a) Diagram at the beginning, (b) Completed diagram

Table 4. Candidate edges sorted by step length

Step Length	Edges
1	(a, ad, db, ba, ab, bd, da, ad, dc, c)
2	(d, ab, da, bb, ad, ba, dd, ac, d)
3	(b, aa, db, bd, aa, bd, dc, a)
4	(a, ab, dd, ba, ad, bc, d)
5	(b, ad, da, bd, ac, b)
6	(d, aa, dd, bc, a)
7	(a, ad, dc, b)
8	(d, ac, d)
9	(c, a)
10	()

Table 4 shows all possible edges sorted by the distance between nodes in a referring sequence, which is indicated by step length. These ordered edges are used as candidate edges to be added to the empty diagram in the next step.

Through an iterative process, edges are added and ordered by step length to the empty diagram until it is connected. This process checks whether the candidate edge turns the diagram into an invalid Hass diagram. If an edge violates the Hass diagram properties, the edge is ignored and the next candidate edge is considered. The candidate edge is also checked to make sure that its respective index in the distance information of the referred sequence is not -1. In the procedure of adding edges, we try not to add further orders on a node that is already in the connected component (i.e., we just make sure that the current diagram being created is a valid Hasse diagram and do not care about making each node as comparable as possible). Figure 2 (b) shows the completed Hass diagram of S_2 in regards to S_1.

The weight of each edge is calculated as shown in Equation 1, where f is the step length of the edge and d is the distance difference of symbol x from y in referring and referred sequences.

$$Weight(x, y) = \frac{1}{f(1+d)} \tag{1}$$

The similarity between a referring sequence with a referred sequence is simply calculated as the summation of all weights as shown in Equation 2.

$$S_u(S_1, S_2) = \sum_{\forall x,y} Weight(x, y) \tag{2}$$

Table 5 shows the weight of each edge for the Hass diagram of Figure 2 (b).

Generally, the similarity of two sequences S_1 and S_2 is calculated as the average of similarity score when S_1 is referred sequence and S_2 is referring sequence, and similarity score when S_2 is referred sequence and S_1 is referring sequence. Equation 3 shows the similarity measure of two sequences. In our example, the similarity measure of S_1 and S_2 is *(2.72+ 3.40)/2 = 3.06*.

$$Sim(S_1, S_2) = \frac{S_u(S_1, S_2) + S_u(S_2, S_1)}{2} \tag{3}$$

Table 5. Edges weighting details for S_1 and S_2

Edge	f	d	Weight
(a	1	|4-3| = 1	0.5
ad	1	|2-8| = 6	0.14
ba	1	|4-3| = 1	0.5
ab	1	|3-2| = 1	0.5
bd	1	|2-6| = 4	0.2
ad	1	|2-8| = 6	0.14
c)	1	|2-6| = 4	0.2
ab	2	|5-2| = 3	0.12
ba	2	|5-3| = 2	0.16
ac	2	|4-3| = 1	0.25

3.1 Time Complexity

In this section we analyze the time complexity of our proposed sequence similarity metric, but it is necessary to denote that we do not focus on the time complexity of the metric and we do not evaluate the time complexity and performance of the proposed metric in this paper. The running time of the proposed metric is the summation of the steps of the metric as shown in Equation 4, where T_{FC} is the runtime for reformatting sequences, T_{DI} is the runtime for the constructing distance information table, and T_{HD} is the runtime for building a Hasse diagram.

$$T_{overall} = T_{FC} + T_{DI} + T_{HD} \tag{4}$$

Suppose m is the referring sequence length, n is the referred sequence length, and s is its number of symbols. The reformatting procedure needs to iterate over the referred sequence and in each step checks if the referring sequence contains the current symbol of the referred sequence. Therefore, the combination of a two linear procedure gives $O(nm)$ the time complexity order of T_{FC}.

The procedure shown in Figure 1, calculating the distance information iterates over the referred sequence and in each step updates some table cells. The number of updated cells in each step does not exceed the number of symbols. Therefore, the time complexity of T_{DI} is represented as $O(ns)$.

In the procedure of building a Hasse diagram, edges are added until the diagram is connected. The upper bound of this procedure is when the connectivity condition is never satisfied as it needs all possible edges checked. Therefore the time complexity of T_{HD} is represented as shown in Equation 5.

$$T_{HD} = (m-1) + (m-2) + \cdots + 1 \in O(m^2) \tag{5}$$

Therefore, the cumulative time complexity of the proposed metrics is the maximum of the time complexity of the mentioned procedures [31] as presented in Equation 6.

$$O(T_{overall}) = O(Max(T_{FC}, T_{DI}, T_{HD})) = O(Max(nm, ns, m^2) \tag{6}$$

In a typical case where some symbols are repeated in the referred sequence (i.e., $s<n$) and two compared sequences have almost equal length (i.e., $n \approx m$), the order of total runtime will be $O(mn)$.

4. EVALUATION

We evaluate the effectiveness of our proposed sequence similarity metric through clustering the sequences of users' web surfing sessions. Every web surfing session contains several consecutive web sites that were visited by each user. We build a probabilistic generative model inspired by Markov chains [32] to simulate a web session. In this model each website is represented by a symbol and a probability distribution is considered for generating the next symbol (i.e., the next website to be visited). Table 6 shows three probabilistic models: M_1, which tends to generate the *abcdef* sequence, means the web sites must be visited in the order of an *a, b, c, d, e,* and *f* sequence; M_2, which tends to generate the *cbafde* sequence; and M_3, which tends to generate the *edfbac* sequence. These three models are shown in Table 6. The symbol *"("*, indicates the starting of a web surfing session.

As an example, the first row of numbers indicates the probability distribution of surfing web sites *a, b, c, d, e,* and *f* after *"("*, as the first visited web site for M_1. The probability distribution function simulates the behavior of a typical web surfer in which the next website is likely to be related to the current website.

We generated 20 web sessions based on each probabilistic generative model presented in Table 6. To evaluate the effectiveness of our proposed metric, we clustered the 60 generated web sessions using a hierarchical agglomerative clustering approach [33] and studied how effective our proposed metric clusters that the sequences generated were by using each model. Table 7 shows the web session sequences generated using M_1, M_2, and M_3.

Strategies for hierarchical clustering are divided into two basic paradigms: agglomerative (bottom-up) and divisive (top-down). Agglomerative strategies start at the bottom and in each level, recursively merge a selected pair of clusters into a single cluster [33]. Each level of the hierarchy represents a particular grouping of the data into disjointed clusters of observations.

Hierarchical clustering approaches need a linkage strategy. Three widely used linkage strategies are single linkage, complete linkage, and group average linkage [26]. We used the group average linkage strategy in which the similarity of two clusters is the average similarity of each pair of them [26]. After that, using a sequence similarity metric, the similarity of these pairs is measured. We started clustering by considering each sequence as a cluster. Through a recursive process we merged the two most similar clusters until only one cluster containing all of the sequences was. We cut off the dendrogram [33] where three clusters were figured out.

We evaluated different sequence similarity metrics, including our proposed sequence similarity metric by comparing the clustering purity [34] results from each sequence similarity metric.

Table 6. Probabilistic generative model of web sessions

M_3			e	d	f	b	a	c
	M_2		c	b	a	f	d	e
		M_1	a	b	c	d	e	f
(	(	(	.5	.25	.16	.7	.2	0
e	c	a	0	.5	.25	.16	.7	.2
d	b	b	.2	0	.5	.25	.16	.7
f	a	c	.7	.2	0	.5	.25	.16
b	f	d	.16	.7	.2	0	.5	.25
a	d	e	.25	.16	.7	.2	0	.5
c	e	f	.5	.25	.16	.7	.2	0

Table 7. Generated web sessions

Web Session Sequences			
M_1			
a e a b f a b c e f a	d e f a c a b d e f b	a b e f a b d f a b c	d e b c d f b f a e f
d b d e f c d b d e a	a c e a b c d e b c d	a b d e c e f a b e c	a e f a b e b e d e b
a b c d a b c d e b c	d f b c d a c f a b c	a b d e f b d f a b c	a b e f b d a d a b c
b a d a c d e f a b c	b d a b c f a b c e f	b c d e f b d f a b c	a b e d e a b c d e f
b e f a c d a b d e f	a b c d e f a b c f a	c f a d f a b d a b c	b e a f b e f a e f a
M_2			
c b a f d e b a f d e	f d e c b f e c b d a	a f a d a f d c f b a	a f c f d e f c a f d
d c a e c b d b d b a	c f d f d b d b a d c	b a f d e c b a d a f	b a f d e a f d c a f
f c b d b d b a d b f	c b e c d e b a f c a	c b a d e b a d c a e	f d e f e c b a f e c
b f d c a e b f c f d	b f d b a f e c f d e	f d c f d e c e b a e	c b f e b f d e c b a
a f d b a f d e a f c	c b a d e c b a d e b	f d c f e c b a d b a	c f e c b d c b a f a
M_3			
e b a f b f a e f a d	e d f a c e d b a c e	d b c d b a c d a e f	f c f b a c d f c e f
e f b c e d c e d f c	e d b a c f a c d f b	e b c e f a e a e f a	b a d f b d b f b c e
e d b a d f c e d a c	b a c b e f a d b f b	d f b a d f c d f c d	e d b d b e d b a f b
b a d f c f a c e d f	e d c f a f a c e d a	d b a e f e d b a d b	e f b e d a c e d b c
d a e b a c e d f b a	f c e d c e b c f b c	e d f b c f b e d f b	e d f a c d f e d a c

We calculated the purity as Equation 7, where C is the set of clusters, L is the set of categories (reference distribution), and N is the number of clustered items.

$$Purity = \sum_i \frac{|C_i|}{N} Max_j \frac{|C_i \cap L_j|}{|C_i|} \tag{7}$$

In comparison, we chose Needleman-Wunsch [7] and Smith-Waterman [4] metrics from the alignment approaches, as they are most related sequence similarity metrics to our proposed sequence similarity metric. In addition, Needleman-Wunsch and Smith-Waterman are the most alignment based sequence similarity metrics and used other sequence alignment approaches as the baseline. We also chose Levenshtein [6] as one of the most common and baseline edit-based similarity metrics. From among the many statistical and probabilistic approaches, N-gram-based approaches are more related to our work as they consider the order of symbols in sequences. Therefore, we compared our proposed metric with *d2* [5] as a common N-gram-based metric. We implement the *d2* metric with a window length of 5 and a gram length of 2.

Table 8 shows web session sequences clustering results by using our proposed sequence similarity metric as compared to other metric companions with the time complexity of each metric. The time complexity of all metrics are the same and are *O(mn)*, where *m* and *n* are the lengths of the referring and referred sequences. The higher purity rate of clustering using our proposed approach shows the superiority of the effectiveness of our proposed sequence similarity metric in contrast to other metrics.

The superior results of our proposed sequence similarity metric show that considering that all the symbols' orders of sequences may lead to a more effective similarity measurement. d2 and Smith-Waterman metrics present nearly the same effectiveness in measuring sequence similarity through clustering web session problem logs. The effectiveness of these two sequence similarity metrics is comparable to the effectiveness of our proposed sequence similarity metric. However, Levenshtein and Needleman-Wunsch perform significantly less efficient as sequence similarity metrics through clustering web sessions in contrast to our proposed metric of d2 and Smith-Waterman.

Table 8. Comparison of clustering web session results

	Our Proposed Sequence Similarity Metric			
	Cluster 1	**Cluster 2**	**Cluster 3**	**Time Complexity**
M_1	20	0	0	
M_2	0	18	2	O(mn)
M_3	3	0	17	
Purity		91%		
	d2 (W=5, ω=2)			
	Cluster 1	**Cluster 2**	**Cluster 3**	**Time Complexity**
M_1	19	1	0	
M_2	0	16	4	O(mn)
M_3	3	0	17	
Purity		86%		
	Smith-Waterman			
	Cluster 1	**Cluster 2**	**Cluster 3**	**Time Complexity**
M_1	18	1	1	
M_2	0	18	2	O(mn)
M_3	3	2	15	
Purity		85%		
	Levenshtein			
	Cluster 1	**Cluster 2**	**Cluster 3**	**Time Complexity**
M_1	10	3	7	
M_2	8	12	0	O(mn)
M_3	3	6	11	
Purity		55%		
	Needleman-Wunsch			
	Cluster 1	**Cluster 2**	**Cluster 3**	**Time Complexity**
M_1	13	3	4	
M_2	9	11	0	O(mn)
M_3	11	2	7	
Purity		51%		

5. CONCLUSION AND FUTURE WORKS

We proposed a new sequence similarity metric that considers all ordered pairs of two sequences by building a Hass diagram and utilizing a weighting scheme. We evaluated the effectiveness of our proposed metric through a clustering application, as one of many applications of sequence similarity metrics. As an effectiveness measure, we considered the purity of clustering and experimental results, which shows the superiority of our proposed sequence similarity metric in achieving the highest clustering purity out of other more common and popular sequence similarity metrics such as d2, Smith-Waterman, Levenshtein, and Needleman-Wunsch.

However, the proposed sequence similarity metric has the potential to be extended in different ways. Considering common substrings with different weighting strategies between two sequences is something to work on in the future. The sequence similarity metric can also be combined with alignment-based metrics to support applications in which location is an effective factor, similar to bio-sequences.

We considered sequences simply as consecutive symbols. However, it can be extended to item sets wherein a sequence is formed by consecutive items in which each item contains some elements [1].

The proposed similarity metric works based on string matching, which limits the metric to

discrete data and it can be modified to support contiguous data sequences too. Adding a normalization step to reduce the effect of sequence lengths could be considered as another extension to the current work.

REFERENCES

[1] G. Dong and J. Pei, *Sequence Data Mining*: Springer; 1 edition (August 9, 2007), 2007.

[2] Y. Jiong, *"CLUSEQ: Efficient and Effective Sequence Clustering,"* in *19th International Conference on Data Engineering*, Bangalore, India, 2003, pp.101-112.

[3] W. Cohen, *et al.*, *"A Comparison of String Metrics for Matching Names and Records,"* in *ACM International Conference on Knowledge Discovery and Data Mining (KDD) 09, Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[4] T. F. Smith and M. S. Waterman, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology,* Vol.147, 1981, pp.195-197.

[5] D. C. Torney, *et al.*, *"Computation of d2:A Measure of Sequence Dissimilarity," Computers and DNA,* 1990, pp.109-125.

[6] Levenshtein, *"Binary codes capable of correcting deletions, insertions, and reversals," Soviet Physics Doklady,* Vol.10, 1966, pp.707-10.

[7] S. B. Needleman and C. D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," *Journal of Molecular Biology,* Vol.48, 1970, pp.443-453.

[8] K. Min, *et al.*, *"Typographical and Orthographical Spelling Error Correction,"* 2008.

[9] Hodge and Austin, *"An Evaluation of Phonetic Spell Checkers "* Mechanisms of Radiation Eflects in Electronic Materials 2001.

[10] B. Bansal, *et al.*, *"Isolated-word Error Correction for Partially Phonemic Languages using Phonetic Cues,"* in *International Conference on Knowledge based Computer Systems (KBCS 2004)*, Hyderabad, India, 2004, pp.509-519.

[11] R. Mitton, "Ordering the suggestions of a spellchecker without using context," *Nat. Lang. Eng.,* Vol.15, 2009, pp.173-192.

[12] L. Yujian and L. Bo, "A Normalized Levenshtein Distance Metric," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol.29, June 2007 2007.

[13] M. Bilenko and R. J. Mooney, *"Adaptive Duplicate Detection Using Learnable String Similarity Measures,"* in *ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington, D.C., 2003.

[14] E. S. Ristad and P. N. Yianilos, "Learning String Edit Distance," *IEEE Transactions on Pattern Recognition and Machine Intelligence,* Vol.20, 1998, pp.522-532.

[15] P. Jaccard, "Étude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bulletin de la Société Vaudoise des Sciences Naturelles,* Vol.37, 1901, pp.547-579.

[16] V. J. Hodge and J. Austin, *"A Novel Binary Spell Checker.,"* *in International Conference on Artificial Neural Networks (ICANN'2001)*, Vienna, Austria, 2001.

[17] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal,* Vol.29, 1950, pp.147-160.

[18] J. Zobel and P. Dart, "Finding approximate matches in large lexicons," *Softw. Pract. Exper.,* Vol.25, 1995, pp.331-345.

[19] A. Elmagarmid, *et al.*, "Duplicate Record Detection: A Survey," *IEEE Transactions on Knowledge and Data Engineering,* Vol.19, 2007, pp.1-16.

[20] A. E. Monge and C. P. Elkan, *"The field matching problem: Algorithms and applications," in Second International Conference on Knowledge Discovery and Data Mining, (KDD)*, 1996.

[21] M. A. Jaro, "Advances in record linkage methodology as applied to the 1985 census of Tampa Florida," *Journal of the American Statistical Society,* Vol.84, 1989, pp.414-20.

[22] M. A. Jaro, "Probabilistic linkage of large public health data file," *Statistics in Medicine,* Vol.14, 1995, pp.491-8.

[23] W. E. Winkler, *"String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage',"* in Section on Survey Research Methods, American Statistical Association, 1990, pp.472-477.
[24] J. Zobel and P. Dart, *"Phonetic String Matching: Lessons from Information Retrieval," in 19th annual international ACM SIGIR conference on Research and development in information retrieval* 1996.
[25] J. M. Ponte and W. B. Croft, *"A language modeling approach to information retrieval," in 21st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '98)*, ACM, New York, NY, USA, 1998, pp.275-281.
[26] C. D. Manning*, et al.*, *An Introduction to Information Retrieval*: Cambridge University Press; 1 edition (July 7, 2008), 2008.
[27] Altschul*, et al.*, "Basic local alignment search tool," *J Mol Biol,* Vol.215, 1990, pp.403-410.
[28] W. Pearson, "Rapid and sensitive sequence comparisonwith fastp and fasta," *Methods Enzymol,* Vol.183, 1990, pp.63-98.
[29] L. Noe and G. Kucherov, "YASS: enhancing the sensitivity of DNA similarity search.," *Nucleic Acids Research,* Vol.33(2), 2005, pp.540-543.
[30] W. G. Liu*, et al.*, *"Bio-Sequence Database Scanning on a GPU,"* 2006.
[31] K. H. Rosen, *"The Growth of Functions,"* in Discrete Mathematics and its Applications, 4th edition ed: McGraw-Hill, 1998, pp.80-90.
[32] D. J. Hartfiel, *Markov Set-Chains*: Springer-Verlag, 1998.
[33] T. Hastie*, et al.*, *"Hierarchical clustering,"* in The Elements of Statistical Learning, ed New York: Springer, 2009, pp.520-528.
[34] E. Amigó*, et al.*, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Information Retrieval,* Vol.12, 2008, pp.461-486.

Mohammad Hassan Pandi
Received his B.Sc. in Computer Engineering from the Sharif University of Technology, Tehran, Iran in 2007 and his M.Sc. in Artificial Intelligence from the Iran University of Science and Technology, Tehran, Iran in 2011. He is currently involved in the Iranian national search engine project. His research interests include data mining applications and ranking in information retrieval.

Omid Kashefi
Received his M. Sc. in Computer Software Engineering from the Iran University of Science and Technology, Tehran, Iran in 2008. He is a Lecturer and Research Associate in the School of Computer Engineering at the Iran University of Science and Technology. His major research interests include distributed systems, virtualization, system software and operating systems, and natural language processing.

BehrouzMinaei
Obtained his Ph.D. from the Computer Science & Engineering Department of Michigan State University, USA, in the field of Data Mining. He is currently an assistant professor in the Department of Computer Engineering at the Iran University of Science & Technology. He is also leader of the Data and Text Mining Research Group for the Noor Computer Research Center, Qom, Iran, where they develop large scale NLP and Text Mining projects for the Persian/Arabic languages.