An Edit-distance Model for the Approximate Matching of Timed Strings

Simon Dobrišek, Janez Žibert, Nikola Pavešić *Member, IEEE*, and France Mihelič *Member, IEEE*

Abstract—An edit-distance model that can be used for the approximate matching of contiguous and non-contiguous timed strings is presented. The model extends the concept of the weighted string-edit distance by introducing timed edit operations and by making the edit costs time dependent. Special attention is paid to the timed null symbols that are associated with the timed insertions and deletions. The usefulness of the presented model is demonstrated on the classification of phone-recognition errors using the TIMIT speech database.

Index Terms—Pattern matching, similarity measures, edit distance, classifier evaluation, speech recognition.

1 Introduction

Strings are commonly defined as finite sequences of symbols taken from a finite alphabet. In the field of pattern recognition, string patterns are not restricted just to such strings. Symbols that compose string patterns are often associated with their real-world manifestation. Different kinds of signals that are modeled as functions of time (and/or space) represent such manifestations. In spoken language modeling, for instance, sequences of spoken words are associated with segments of a speech signal, or at least with two time values, representing the time period when the words were most probably uttered by a speaker [1]. In the area of real-time computations, the concept of the timed words of a timed language that are accepted by a timed automaton is introduced [2], [3]. Additional examples can be found in many other areas, e.g., telecommunications, temporal data mining, event-management systems and system monitoring.

It is often necessary to compare string patterns and to measure the extent to which they differ. Comparison is the basis for pattern recognition, error classification, error correction, and for determining relationships [4], [5]. In this paper we investigate the pattern-matching problem for finite sequences of symbols, where each symbol is associated with two time values that represent the time period when the corresponding symbol is available. Such sequences are called timed strings. We focus on an approximate matching of timed strings that is based on edit distance. Our main motivation for developing the model is to improve the classification of speech-recognition errors.

The rest of the paper is organized as follows: Sec. 2 presents related work. In Sec. 3, non-empty and empty timed strings are defined and discussed. A timed edit-distance model that can be used for the approximate matching of timed strings is presented in Sec. 4. Possible applications of the presented model are briefly discussed in Sec. 5. In Sec. 6 we focus on a demonstration of using the model as a scoring algorithm for the classification of phone-recognition errors. The last section concludes with a summary and plans for future work.

• *The authors are with the Faculty of Electrical Engineering, University of Ljubljana, SI-1000 Ljubljana, Slovenia.*
E-mail: see http://luks.fe.uni-lj.si/en/staff

2 Related Work

The basic concept of string comparison based on edit distance was proposed a long time ago in coding theory [6]. This concept is often referred to as the Levenshtein distance. Efficient algorithms for computing the Levenshtein distance were also proposed [7], [8], and various similar versions of the algorithm and their reviews can be found in the literature [9], [10], [11]. The Levenshtein distance was extended by weighting the basic edit operations using edit costs that are symbol dependent [12]. This more general concept is usually referred to as the generalized Levenshtein distance or the weighted string-edit distance.

The concept of the string-edit distance is related to the concept of discrete time warping [13]. In the speech recognition community this concept is called dynamic time warping (DTW) [14]. The notion of discrete time warping can be formalized as a linking that connects the time scales of the two sequences. The linking concept is similar to the trace concept used with comparisons that are based on the string-edit distance. The differences between linking and trace reflect the differences between expansion-compression and insertion-deletion [13].

The timed strings discussed in this paper can be used for a representation of event sequences, and the presented model can be used for measurements of the similarities between such sequences. Several other models were developed that can also be used for the same purpose and are generally used for the retrieval of similar time sequences [15]. The model proposed by Mannila and Ronkainen [16] is related to the model presented in this paper. The main difference is that their model considers sequences of events with single time stamps, rather than time intervals. Their model is also based on a different set of basic edit operations (moves instead of substitutions) and uses different symbol- and time-distance functions for the edit costs.

Mongeau and Sankoff [17] showed how concepts from the theory of sequence comparison can be adapted to measure the similarity or dissimilarity between two musical scores. Their model is related to the one proposed in this paper as it also considers time information. The main difference is that their model considers only a simple difference of time lengths, while our model considers a more general time-distance function. Our model also considers different timed null units and supports non-contiguous timed strings as well.

Another group of important possible applications of the presented model is the evaluation of automatic speech recognizers [1]. Our research interests center around such systems, and some known assessment problems [18], [19] actually motivated the development of the proposed model. In this context, the presented model is related to an algorithm that performs the so-called time-mediated alignment of speech transcriptions and is implemented in the NIST Scoring Toolkit (SCTK) [20]. The main difference between our model and this algorithm is that our model considers insertions and deletions differently, and supports a more general edit-cost function. In this paper we focus on this ASR-related application and the usefulness of the proposed model is demonstrated on the classification of phone-recognition errors.

3 TIMED STRINGS

Assume that the symbols taken from some alphabet are available in finite time intervals. Such events can be represented as symbols associated with two time values: the start and end times of the time interval when the corresponding symbol is available. Let us call such an association between a symbol and two time values a *timed symbol*. Let $\mathbf{a} = (a, s, e)$ denote a timed symbol, where a is a symbol taken from a finite alphabet Σ and $s, e \in \mathbb{R}$ are the start and end times of the time interval when the symbol a is available. We would normally assume non-negative durations of the time intervals, i.e., $e \geq s$.

Let then Γ be the set of all the possible finite-duration timed symbols, i.e.,

$$\Gamma = \{ \ (a, s, e) \mid a \in \Sigma \ ; \ s, e \in \mathbb{R}, \ e \geq s \ \} \ . \tag{1}$$

The timed strings that we consider are time-ordered sequences of timed symbols, where the time-order is defined as the non-decreasing order of the middle times of the symbols in a sequence. Our timed strings are required to be ordered sequences of timed symbols, and thus all the middle times of the symbols in a sequence are required to be different from each other. The middle times are used so that both the start and end times of the symbols influence the time order.

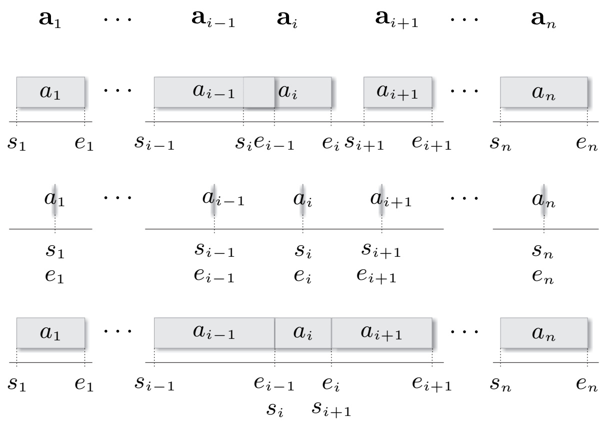

$\mathbf{a}_1$ $\cdots$ $\mathbf{a}_{i-1}$ $\mathbf{a}_i$ $\mathbf{a}_{i+1}$ $\cdots$ $\mathbf{a}_n$
a_1 a_{i-1} a_i a_{i+1} a_n
s_1 e_1 s_{i-1} s_i e_{i-1} e_i s_{i+1} e_{i+1} s_n e_n
a_1 a_{i-1} a_i a_{i+1} a_n
s_1 e_1 s_{i-1} e_{i-1} s_i e_i s_{i+1} e_{i+1} s_n e_n
a_1 a_{i-1} a_i a_{i+1} a_n
s_1 e_1 s_{i-1} e_{i-1} s_i e_i s_{i+1} e_{i+1} s_n e_n

Fig. 1: A graphical representation of three timed strings of length n. The top string is a generic one, where time gaps/overlaps are allowed between the subsequent timed symbols. The middle string comprises only zero-duration timed symbols. In the bottom string, the end time of $\mathbf{a}_i$ is equal to the start time of $\mathbf{a}_{i+1}$ for all $i = 1, \cdots, n-1$. Such timed strings are called *contiguous*.

Let $\mathbf{a}^n = \mathbf{a}_1 \cdots \mathbf{a}_i \cdots \mathbf{a}_n$ denote such a timed string, where n is the number of timed symbols $\mathbf{a}_i = (a_i, s_i, e_i) \in \Gamma$, and let $m_i \in \mathbb{R}$ denote the middle time of a timed symbol $\mathbf{a}_i$, i.e., $m_i = \frac{1}{2}(s_i + e_i)$.

The set of all the possible timed strings of length $N \in \mathbb{N}$ is then defined as

$$\begin{aligned} \Gamma^N = \{ \ & \mathbf{a}_1 \cdots \mathbf{a}_i \cdots \mathbf{a}_N \quad \mid \\ & \mathbf{a}_i \in \Gamma, \ i = 1, \cdots, N \ ; \\ & m_i \neq m_j \ , \ i, j = 1, \cdots, N \ , \ i \neq j \ \ \} \ . \end{aligned} \tag{2}$$

The set of all the possible finite-length timed strings is then defined as

$$\Gamma^+ = \bigcup_{N \in \mathbb{N}} \Gamma^N \ . \tag{3}$$

Note that the notation used does not distinguish between a timed string of length one and a single timed symbol, i.e., $\mathbf{a}^1 = \mathbf{a}_1$.

A graphical representation of three timed strings is shown in Fig. 1. The middle and bottom strings are special cases, where additional conditions are satisfied on their timed symbols.

3.1 An empty timed string

The concept of the edit distance between two timed strings requires the definition of an empty timed string. Let $\boldsymbol{\epsilon}$ denote an empty timed string that is defined as the identity element with the timed-string concatenation operation, i.e.,

$$\boldsymbol{\epsilon} \ \mathbf{a}^n = \mathbf{a}^n \boldsymbol{\epsilon} = \mathbf{a}^n \quad \text{for all} \quad \mathbf{a}^n \in \Gamma^+ \ .$$

At first glance, one could assume that there should be only one such empty timed string, and, that there is no need for it to be time dependent. This assumption is actually used in the algorithm for comparisons of musical sequences [17]. The same assumption can also be identified in the time-mediated alignment algorithm that is implemented as part of the NIST SCTK toolkit [20].

On the other hand, an empty timed string can be seen rather as a time interval when no symbol is available, or, in other words, when nothing happened. An empty timed string can thus be defined as a special timed string that comprises a timed null symbol. The proposed edit-distance model is based on this assumption and the results of the experiments presented later in the paper indicate that this may be an improvement over the use of the single time-independent empty string.

Let then ε be the null symbol that is associated with two time values. A timed null symbol is then denoted by

$$\boldsymbol{\epsilon} = (\varepsilon, s, e) \ , \tag{4}$$

where $s, e \in \mathbb{R}$ are the start and end times of the time interval when no symbol is available. Note that the time e can be less than time s. As shown later, this generalizes the proposed model at the point where timed null symbols need to be inserted between two overlapped non-null symbols.

As the notation used does not distinguish between a single timed null symbol and a timed string that comprises only one such unit, the symbol $\boldsymbol{\epsilon}$ also denotes an empty timed string. The set of all the possible empty timed strings is then defined as

$$\Upsilon = \{(\varepsilon, s, e) \mid s, e \in \mathbb{R} \ \} \ . \tag{5}$$

The timed null symbols play an important role in the proposed edit-distance model. The basic edit operations that are called timed insertions and timed deletions are just substitutions of timed symbols, assumed to be deleted or inserted, and timed null symbols that are positioned between two subsequent timed symbols in the opposite string. If several subsequent deletions or insertions are required for the minimum-cost transformation of one timed string into the other, then several timed null symbols could be considered to be positioned between two subsequent timed symbols in the opposite string.

Let us denote by $\boldsymbol{\epsilon}^{r_i} = \boldsymbol{\epsilon}_{i,1} \cdots \boldsymbol{\epsilon}_{i,r_i}$ a sequence of timed null symbols that is positioned between the two timed symbols $\mathbf{a}_i$ and $\mathbf{a}_{i+1}$ of a timed string $\mathbf{a}^n$, where r_i denotes the length of the sequence $\boldsymbol{\epsilon}^{r_i}$. Let then $\boldsymbol{\epsilon}^{r_0}$ denote the sequence of timed null symbols that is positioned before the first timed symbol $\mathbf{a}_1$, and $\boldsymbol{\epsilon}^{r_n}$ the sequence that is positioned after the last timed symbol $\mathbf{a}_n$.

According to the notation used, the sequence of timed null symbols $\boldsymbol{\epsilon}^{r_i}$ can also be seen as the concatenation of the subsequent empty timed strings. Following this, a timed string $\mathbf{a}_1 \cdots \mathbf{a}_i \cdots \mathbf{a}_n$ can also be defined as the alternating concatenation of the timed strings of length one and the concatenated sequences of empty timed strings inserted between them, i.e.,

$$\mathbf{a}_1 \cdots \mathbf{a}_i \cdots \mathbf{a}_n \ = \ \boldsymbol{\epsilon}^{r_0}\mathbf{a}_1\boldsymbol{\epsilon}^{r_1}\mathbf{a}_2 \cdots \boldsymbol{\epsilon}^{r_{i-1}}\mathbf{a}_i\boldsymbol{\epsilon}^{r_i} \cdots \boldsymbol{\epsilon}^{r_{n-1}}\mathbf{a}_n\boldsymbol{\epsilon}^{r_n} \tag{6}$$

In general, it does not matter which particular start and end times of the timed null symbols in the above sequences are actually considered, as long as the time order of the string is satisfied. However, one would naturally expect that the timed null symbols in $\boldsymbol{\epsilon}^{r_i}$ somehow cover the time gap/overlap between the subsequent $\mathbf{a}_i$ and $\mathbf{a}_{i+1}$. Furthermore, for simplicity, we can also assume that the timed null symbols in $\boldsymbol{\epsilon}^{r_i}$ are all just replications of the same timed null symbol, say $\boldsymbol{\epsilon}_i$, that covers the whole time gap/overlap between $\mathbf{a}_i$ and $\mathbf{a}_{i+1}$, i.e., $\boldsymbol{\epsilon}_{i,1} = \boldsymbol{\epsilon}_{i,2} = \cdots = \boldsymbol{\epsilon}_{i,r_i} = \boldsymbol{\epsilon}_i$. The start time of the replicated $\boldsymbol{\epsilon}_i$ is then equal to the end time of $\mathbf{a}_i$ and its end time is then equal to the start time of $\mathbf{a}_{i+1}$. The start and end times of $\boldsymbol{\epsilon}_0$ can then both equal the start time of the first timed symbol $\boldsymbol{a}_1$. Similarly, the start and end times of $\boldsymbol{\epsilon}_n$ can then both equal the end time of the last timed symbol $\boldsymbol{a}_n$.

4 Timed String-Edit Distance

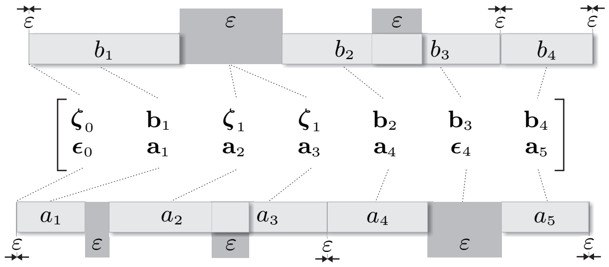
The edit distance for the timed strings is defined by a pair $(\{\Gamma \cup \Upsilon\}, c)$, where $c : E \rightarrow \mathbb{R}_0^+$ is a timed edit-cost function that assigns non-negative real numbers to the timed edit operations in $E = E_s \cup E_d \cup E_i$, where $E_s = \Gamma \times \Gamma$ is a set of timed substitution operations, $E_d = \Gamma \times \Upsilon$ is a set of timed deletion operations, and $E_i = \Upsilon \times \Gamma$ is a set of timed insertion operations.

Each such pair $(\{\Gamma \cup \Upsilon\}, c)$ induces a function $d : \{\Gamma^+ \cup \Upsilon\} \times \{\Gamma^+ \cup \Upsilon\} \rightarrow \mathbb{R}_0^+$ that maps a pair of timed strings into a non-negative real number. The function d is then defined as the minimum sum of the edit costs of the edit operations that, step by step, transform one timed string into another.

4.0.1 An Algorithm for Computing the Timed String-Edit Distance

An algorithm for computing the function d is derived from the recursion that is defined with the simple trace distance [5], [21]. The main difference is that our basic edit operations are time dependent, and that we do not consider just a single timed null symbol for all the timed deletions and insertions. Furthermore, if several subsequent timed deletions or insertions are required for the minimum-cost transformation of one timed string into another, then several timed null symbols can be considered to be positioned between the two subsequent timed non-null symbols in the opposite string.

Our edit-distance model is based on the assumption that these timed non-null symbols are just replications of the same timed null symbol. The computation of the function d can then be simplified to the recursion that is directly derived from the recursion that is defined with the simple trace distance. Note, however, that other rules for the determination of the timed null symbols in $\boldsymbol{\epsilon}^{r_i}$ are possible. For instance, the time gap/overlap between $\mathbf{a}_i$ and $\mathbf{a}_{i+1}$ could be split between different timed null symbols in $\boldsymbol{\epsilon}^{r_i}$. This would require a further extension of



ε
b_1
b_2
b_3
b_4
$\boldsymbol{\zeta}_0$ $\mathbf{b}_1$ $\boldsymbol{\zeta}_1$ $\boldsymbol{\zeta}_1$ $\mathbf{b}_2$ $\mathbf{b}_3$ $\mathbf{b}_4$
$\boldsymbol{\epsilon}_0$ $\mathbf{a}_1$ $\mathbf{a}_2$ $\mathbf{a}_3$ $\mathbf{a}_4$ $\boldsymbol{\epsilon}_4$ $\mathbf{a}_5$
a_1
a_2
a_3
a_4
a_5

Fig. 2: A graphical presentation of the optimum alignment between the two timed strings $\mathbf{a}^5$ into $\mathbf{b}^4$. Note how the two timed strings are initially synchronized and how the timed null symbol $\boldsymbol{\zeta}_1$ is replicated.

the recursion below, which is beyond the scope of this paper.

Let us define the recursion on the two timed strings $\mathbf{a}^n \in \Gamma^n$ and $\mathbf{b}^m \in \Gamma^m$. The recursion is defined as follows:

$$\begin{aligned} \mathrm{d}_{0,0} &= c(\boldsymbol{\epsilon}_0\,,\,\boldsymbol{\zeta}_0) \\ \mathrm{d}_{\mathrm{i},0} &= \mathrm{d}_{\mathrm{i}-1,0} \;+\; c(\mathbf{a}_i\,,\,\boldsymbol{\zeta}_0) \\ \mathrm{d}_{0,\mathrm{j}} &= \mathrm{d}_{0,\mathrm{j}-1} \;+\; c(\boldsymbol{\epsilon}_0\,,\,\mathbf{b}_j) \\ \mathrm{d}_{\mathrm{i},\mathrm{j}} &= \min\left\{\begin{array}{lcl} \mathrm{d}_{\mathrm{i}-1,\mathrm{j}-1} & + & c(\mathbf{a}_i\,,\,\mathbf{b}_j)\,, \\ \mathrm{d}_{\mathrm{i}-1,\mathrm{j}} & + & c(\mathbf{a}_i\,,\,\boldsymbol{\zeta}_j)\,, \\ \mathrm{d}_{\mathrm{i},\mathrm{j}-1} & + & c(\boldsymbol{\epsilon}_i\,,\,\mathbf{b}_j) \end{array}\right\}, \end{aligned} \tag{7}$$

where $\mathrm{d}_{\mathrm{i},\mathrm{j}} = d(\mathbf{a}^i, \mathbf{b}^j)$ for $i = 0, \cdots, n$ and $j = 0, \cdots, m$ denotes the distance between the two timed substrings $\mathbf{a}^i$ and $\mathbf{b}^j$ of the two timed strings $\mathbf{a}^n$ and $\mathbf{b}^m$. The symbol $\boldsymbol{\zeta}_j$ denotes the timed null symbol that is positioned and replicated as necessary between $\mathbf{b}_j$ and $\mathbf{b}_{j+1}$ in the same way as the timed null symbol $\boldsymbol{\epsilon}_i$ is positioned and replicated between $\mathbf{a}_i$ and $\mathbf{a}_{i+1}$.

The recursion starts with a pair of possibly different empty timed strings $\mathbf{a}^0 = \boldsymbol{\epsilon}_0$ and $\mathbf{b}^0 = \boldsymbol{\zeta}_0$. The initial cost $c(\boldsymbol{\epsilon}_0\,,\,\boldsymbol{\zeta}_0)$ is thus not necessarily zero. This cost can be seen as a cost for the initial time synchronization of the two time strings being transformed to each other. The recursion is then calculated for all $i = 1, \cdots, n$ and $j = 1, \cdots, m$ and the final $\mathrm{d}_{\mathrm{n},\mathrm{m}} = d(\mathbf{a}^n, \mathbf{b}^m)$ is the timed string-edit distance between $\mathbf{a}^n, \mathbf{b}^m$.

The minimum-cost sequence of edit operations that transforms $\mathbf{a}^n$ into $\mathbf{b}^m$ can be identified at the end of the recursion by starting at $\mathrm{d}_{\mathrm{n},\mathrm{m}}$ and tracing the local minima back to $\mathrm{d}_{0,0}$.

The minimum-cost sequence of edit operations is often presented as the optimum alignment of two strings [4]. Fig. 2 shows the optimum timed alignment of the two timed strings $\mathbf{a}^5$ into $\mathbf{b}^4$. Note how the timed null symbols are considered in the alignment.

4.1 Timed Edit-Cost Function

The crucial parts of the recursion in Eqs. (7) are the costs of the edit operations, defined by the edit-cost function c. The costs of the edit operations are normally related to some dissimilarity measure between the two units being compared, where a greater dissimilarity implies a higher cost value.

The timed symbols are composed of two distinctive parts, the symbol part and the time part. This holds for timed non-null symbols taken from Γ and also for timed null symbols taken from Υ. We investigated the timed edit-cost function c that is simply split into a sum of two parts: the part that is proportional to some symbol-dissimilarity measure between their symbols and the part that is proportional to some time-distance measure between two time intervals.

Let $\mathbf{x} = (x, s^{(x)}, e^{(x)})$ and $\mathbf{y} = (y, s^{(y)}, e^{(y)})$ denote two timed symbols that form a pair $(\mathbf{x}, \mathbf{y})$ taken from the set of timed edit operations E. Let then c_s denote a symbol-dissimilarity measure and c_t a time-distance measure. Both measures can have different ranges, and thus an additional weight factor has to be introduced that balances their contribution to the joint cost. The timed edit-cost function that we investigated is a weighted sum of both measures, i.e.,

$$c(\mathbf{x}, \mathbf{y}) \;=\; \varrho\, c_s(x, y) + (1-\varrho)\, c_t\big((s^{(x)}, e^{(x)}), (s^{(y)}, e^{(y)})\big)\,, \tag{8}$$

where $\varrho \in [0, 1]$ is the weight factor. The functions c_s and c_t and the weight factor ϱ are the model parameters yet to be determined.

4.1.1 Symbol-dissimilarity measure

For the symbol-dissimilarity measure $c_s(x, y)$, where $x, y \in \Sigma \bigcup \{\varepsilon\}$, we can use any of the known edit-cost functions defined with the edit-distance models for common strings. The most obvious choice is the unit edit-cost function that assigns one to two different symbols and zero to two equal symbols. This is, of course, not the only possible choice that can be used in practice. The symbols in Σ usually have some meaning, and a symbol-dissimilarity measure is chosen accordingly. Nevertheless, it is obvious that $c_s(x, y)$ depends on the application of the model.

4.1.2 Time-distance measure

The time part of $\mathbf{x}$ and $\mathbf{y}$ is always a pair of time stamps, no matter whether timed null or non-null symbols are considered. Pairs of time stamps are actually taken from a subspace of $\mathbb{R} \times \mathbb{R}$, and thus $c_t((s^{(x)}, e^{(x)}), (s^{(y)}, e^{(y)}))$ can be any of the known distance metrics that are defined on $\mathbb{R} \times \mathbb{R}$, among them are the Manhattan, Euclidian, Chebyshev and other well-known metrics. For example, if the Manhattan metrics is used, then c_t is defined as follows:

$$c_t\big((s^{(x)}, e^{(x)}), (s^{(y)}, e^{(y)})\big) \;=\; |s^{(x)} - s^{(y)}| + |e^{(x)} - e^{(y)}|\,. \tag{9}$$

Which time-distance measure performs best in practice depends again on the meaning of the symbols that are actually considered and also on how the model is evaluated.

4.2 Measuring the Distance between Timed Strings

The recursion in Eqs. (7) can be recognized as an implementation of the trace distance [21]. It is not hard to see that the calculated d is a metric on timed strings if c is a metric on its domain. We investigated only such timed edit-cost functions c that are metric on $\{\Gamma \cup \Upsilon\} \times \{\Gamma \cup \Upsilon\}$. This means that the calculated function d was also always a metric on timed strings. As c is just a weighted sum of c_s and c_t, it is also not hard to see that c is a metric on its domain if c_s and c_t are metrics on their domains.

5 Applications

The presented edit-distance model can be used whenever timed strings need to be compared. In the field of event-management systems, for instance, finite sequences of events are often represented as timed strings, and similarities between such sequences can then be measured using this model. One could also use it to perform approximate queries from the database of contiguous or non-contiguous event sequences.

Speech utterances are commonly represented by sequences of spoken language units. Such sequences are called speech transcriptions and are not necessarily just sequences of plain symbols. They can also be timed strings, where two time values represent the time interval within which the corresponding spoken language unit is most probably uttered by a speaker. Timed transcriptions of speech are mostly considered to be contiguous. However, non-contiguous timed transcriptions could also be used wherever speech overlaps are possible, for instance, in timed transcriptions of broadcast speech and spoken dialogues.

The presented model can be used for the classification of automatic speech recognition (ASR) errors. This is done by comparing the reference and automatically obtained hypothesis transcriptions of speech using the appropriate edit-distance model. The edit operations on the different symbols in the minimum-cost sequence that transforms one transcription into the other are then classified as recognition errors in terms of substitutions, deletions and insertions. The error classifications obtained using the proposed model consider time information in speech transcriptions, when one is available, and this may improve the diagnostic value of examining the error statistics. The usual approaches that are commonly used for this purpose do not consider time information and this is a known problem in the ASR community [19].

One of the approaches to ASR on platforms with limited computing resources is the use of vector-quantization techniques that convert spoken commands into index strings. In this context, the presented model could also be used for the comparisons of such index strings that can be converted into timed strings [22].

In this paper we demonstrate the usefulness of the presented model on the classification of ASR errors. We conducted many experiments using the model as a scoring algorithm for the classification of the phone-recognition errors from the TIMIT speech database [23]. The main purpose of these experiments is to demonstrate how the timed edit-cost function c strongly depends on the application of the model. The proposed model was also compared to the edit-distance models that are implemented as part of the two well-known speech recognition scoring toolkits: the NIST SCTK [20] and CUED HTK [24].

6 Experiments

For the experiments we built a simple phone recognizer using the CUED HTK toolkit [24]. The conventional HMM-based recognizer was deliberately designed to be simple and to produce a relatively high rate of recognition errors. This is because our goal here is not to improve speech recognition accuracy, but to demonstrate the differences in the recognition scores obtained by different edit-distance models and their variants. The recognizer was trained from the TIMIT speech database and the hypothesis timed phone transcriptions were generated for all the utterances in this database.

The presented model can then be used to classify phone-recognition errors by comparing the hypothesis and reference timed phonetic transcriptions. All the edit operations on the timed phone units that differ in terms of their symbols are then considered as phone-recognition errors. All the substitutions of the two timed phone units that have the same phone symbol would normally be considered as recognition matches, although the two units might differ in terms of their start and/or end times.

Different edit-distance models and their variants can produce different classifications of phone-recognition errors, even though the same pairs of phone transcriptions are compared. Unfortunately, there is no easy or standard way to determine whether a given phone-recognition error classification is more correct than some other. However, at least some indication of which error classification is more correct than the other can be drawn from the analysis of the frequencies of phone-recognition errors that are collected in a so-called confusion matrix [18]. Namely, a joint probability distribution that can be estimated from such a matrix tells us how likely it is that the particular phonetic units are substituted, inserted or deleted during the recognition process.

Considering the nature of the ASR process, we can make certain assumptions about the phone-recognition error distribution [25], [26]. First, substitutions between the phone units within the same phone classes are more likely to happen than substitutions between phone units from different phone classes, e.g., it is more likely that a vowel is substituted with another vowel than with a plosive. Furthermore, we would normally expect that the majority of all phone-recognition errors are not phone deletions or insertions. Finally, we would also expect that the total number of phone-recognition errors is

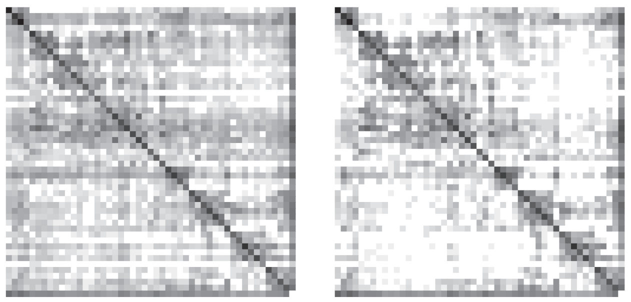

$CSR = 38.40\%,\ MUI = 2.62$ $CSR = 44.65\%,\ MUI = 2.92$

Fig. 3: Graphical representations of the two joint-probability distributions that were estimated from the frequencies of the phone-recognition errors classified using two different edit-distance models and from the same pairs of the TIMIT reference and hypothesis phonetic transcriptions. The different levels of gray correspond to the joint-probability values. The rows and columns of the two matrices correspond to the 48 phonetic units that are usually considered in the TIMIT phonetic transcriptions. The phonetic units are ordered by their classification into the usual phone classes (vowels, sonorants, plosives, etc). The right-most column and the bottom row of both matrices correspond to the null symbol, i.e., phone deletions and insertions.

not much larger than the minimum possible number of edit operations that transforms the hypothesis phonetic transcriptions into the reference ones. We propose four comparison measures that are derived from the above assumptions and that may provide certain indications about which phone-recognition error classification is more correct.

6.1 Comparison measures

Our first measure, denoted as CSR, is defined as the ratio between the number of within-phone-class substitutions and the total number of all the phone-recognition errors. According to our first ASR-related assumption above, the higher is the value of CSR the closer to our expectations is the phone-recognition error distribution, from which CSR is estimated. Fig. 3 shows a graphical representation of two different phone-recognition error distributions. From Fig. 3 it can be seen that the contrast between the within-phone-class substitutions, on one side, and the cross-phone-class substitutions, on the other, is higher on the right-hand distribution than on the left-hand one. This may indicate that the right-hand distribution is probably closer to our ASR-related expectations than the left-hand one, and this should be reflected in the value of CSR. And indeed, the value of the CSR estimated from the right-hand distribution is considerably higher than the value estimated from the left-hand distribution.

From Fig. 3 it can also be seen that the errors in the right-hand distribution are generally distributed less uniformly over the pairs of phonetic units than is the case with the left-hand distribution. From information theory it is known that the mutual information between two random variables is related to the uniformity of their joint-probability distribution: the less uniform the distribution, the higher is the mutual information. Let MUI denote the mutual information that is defined from our joint-probability distributions. As expected, the value of MUI, given in bits per symbol, estimated from the right-hand distribution is again considerably higher than the value estimated from the left-hand distribution.

As CSR and MUI both have higher values for the right-hand distribution than for the left-hand one, we consider this as an indication that the edit-distance model that produced the right-hand distribution classifies phone-recognition errors more correctly than the one that produced the left-hand distribution. However, one could argue that at least MUI alone does not necessarily indicate how correct is the error classification. Namely, the value of MUI could also be high if all the phone-recognition errors are classified only as phone insertions and deletions, which is contradictory to our ASR-related assumptions. Our experiments show that this is actually not true; however, we decided to observe two additional measures that consider the ratio between phone substitutions, deletions and insertions and the total number of phone-recognition errors.

The first additional measure is the ratio between the number of phone substitutions and the total number of all the phone-recognition errors. Let this ratio be denoted as TSR. As mentioned, we expect that the majority of all the phone-recognition errors are not phone deletions or insertions. Accordingly, the higher is the value of TSR the closer to our expectations is the phone-recognition error distribution, from which TSR is estimated.

The second additional measure is a relative increase in the total number of phone-recognition errors when compared to the minimum possible number of edit operations that transforms the hypothesis phonetic transcriptions into the reference ones. This minimum is the sum of the Levenshtein distances between the hypothesis and reference phonetic transcriptions. Let this ratio be denoted as REI. According to our last-mentioned ASR-related assumption, the lower the value of REI, the closer to our expectations is the phone-recognition error distribution, from which REI is estimated.

6.2 Comparison results

We experimented with different variants of the proposed model by using it as a scoring algorithm for the classification of phone-recognition errors. We experimented with many different symbol-dissimilarity measures, time-distance measures and weight-factor values that are part of the timed edit-cost function. Observing all the four proposed comparison measures, we found that the best results are obtained if the fixed edit-cost function that assigns 0.9 to the symbol insertions/deletions and 1.0 to the symbol substitutions is

TABLE 1: The obtained values of CSR, MUI, TSR and REI for the five different edit-distance models.

Model	CSR [%]	MUI $[\frac{bits}{symbol}]$	TSR [%]	REI [%]
LEDM	38.40	2.62	70.72	0.00
HEDM	37.71	2.71	65.23	0.02
NEDM	37.07	2.71	63.77	0.42
NTMA	43.76	2.83	65.54	6.89
TEDM	44.65	2.92	66.15	0.07

used for the symbol-dissimilarity measure c_s, together with the Manhattan time-distance c_t, and the edit-cost weight ϱ that equals 0.5.

Our model with the selected timed edit-cost function, denoted as [TEDM], was then compared to four other edit-distance models that are regularly used as scoring algorithms for the classification of speech recognition errors. The models are the following:

LEDM: The common edit-distance model that computes the Levenshtein distance.

HEDM: The common edit-distance model implemented in the CUED HTK tool *HResults*.

NEDM: The common edit-distance model implemented in the NIST SCTK tool *sclite*.

NTMA: The time-mediated alignment algorithm implemented in the NIST SCTK tool *sclite*

The obtained values of the comparison measures for the five models/algorithms are shown in Tab. 1. From the table it can be seen that our model, denoted as TEDM, achieved the highest value of CSR and MUI, and also the value of TSR that is closest to the value obtained by the LEDM. The value of REI is also very low and according to our ASR-related assumptions, we believe that these results give a good indication that TEDM may be able to classify phone-recognition errors more correctly than the other models.

7 Conclusions and Future Work

In this paper, a unified mathematical framework for contiguous and/or noncontiguous timed strings comprising non-zero- and/or zero-duration time symbols is proposed. A generic timed edit-distance model that can be used for the approximate matching of different types of timed strings is also defined. One can choose the symbol dissimilarity and the time-distance measures that fit best to the problem at hand. Which particular measures and which edit-cost weight factor are the best for a certain type of timed strings need to be verified by experiments. The usefulness of the presented model is demonstrated on the classification of the phone-recognition errors that considers timed phonetic speech transcriptions.

Further extensions of the presented model are possible. As mentioned, other rules for the determination of the multiple timed null symbols that can appear between two subsequent timed non-null symbols could be used. This would require an extension of the local constraints in the recursion for computing the timed edit distance. Such an extension only has sense when we have to deal with non-contiguous timed strings.

References

[1] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.

[2] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[3] S. Verwer, M. de Weerdt, and C. Witteveen, "Identifying an automaton model for timed data," in *Proceedings of the Annual Machine Learning Conference of Belgium and the Netherlands*, Y. Saeys, E. Tsiporkova, B. D. Baets, and Y. van de Peer, Eds. Benelearn, 2006, pp. 57–64.

[4] J. B. Kruskal, "An overview of sequence comparison," in *Time Warps, String Edits and Macromolecules: The Theory and practice of Sequence Comparison*, D. Sankoff and J. Kruskal, Eds. CSLI Publications, 1999, pp. 11–44.

[5] A. Apostolico, "General pattern matching," in *Algorithms and Theory of Computation Handbook*, M. J. Atallah, Ed. Boca Raton: CRC Press, 1999, pp. 13–1–13–22.

[6] V. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics–Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[7] R. A. Wagner and M. J. Fisher, "The string-to-string correction problem," *Journal of the Association for Computing Machinery*, vol. 21, no. 1, pp. 168–173, 1974.

[8] W. Masek and M. Paterson, "A faster algorithm for computing string edit distances," *Journal of Computer System Sciences*, vol. 20, no. 1, pp. 18–31, 1980.

[9] A. V. Aho, "Algorithms for finding patterns in strings," in *Handbook of theoretical computer science*, J. Leeuwen, Ed. Elsevier Science Publishers, 1990, pp. 255–300.

[10] A. Marzal and E. Vidal, "Computation of normalized edit distance and applications," *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 926–932, 1993.

[11] H. Bunke and J. Csirik, "Parametric string edit distance and its application to pattern recognition," *IEEE Transations On Systems, Man, and Cybernetics*, vol. 25, no. 1, pp. 202–206, 1995.

[12] T. Okuda, E. Tanaka, and T. Kasai, "A method of correction of garbled words based on the levenshtein metric," *IEEE Transactions on Computers*, vol. 25, no. 2, pp. 172–177, 1976.

[13] J. B. Kruskal and M. Liberman, "The symmetric time warping problem: From continuous to discrete," in *Time Warps, String Edits and Macromolecules: The Theory and practice of Sequence Comparison*, D. Sankoff and J. Kruskal, Eds. CSLI Publications, 1999, pp. 125–161.

[14] C. S. Myers and L. R. Rabiner, "A comparative study of several dynamic time-warping algorithms for connected-word recognition," *The Bell System Technical Journal*, vol. 60, no. 5, pp. 1389–1409, 1981.

[15] M. L. Hetland, "A survey of recent methods for efficient retrieval of similar time sequences," in *Data Mining in Time Series Databases*, M. Last, A. Kandel, and H. Bunke, Eds. World Scientific, 2004, pp. 27–49.

[16] H. Mannila and P. Ronkainen, "Similarity of event sequences," in *TIME '97: Proceedings of the Fourth International Workshop on Temporal Representation and Reasoning*. Florida, USA: IEEE Computer Society, 1997, pp. 136–139.

[17] M. Mongeau and D. Sankoff, "Comparison of musical sequences," *Computers and the Humanities*, vol. 24, no. 3, pp. 161–175, 1990.

[18] D. Gibbon, R. Moore, and R. Winski, Eds., *Handbook of Standards and Resources for Spoken Language Systems*. Berlin: Mouton de Gruyter, Walter de Gruyter Publishers, 1997.

[19] G. Doddington, "Word alignment issues in asr scoring," in *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop*, Virgin Islands, USA, 2003, pp. 630–633.

[20] *The NIST Speech Recognition Scoring Toolkit (SCTK) Version 2.1.4*, National Institute of Standards and Technology (NIST), USA, 2007. [Online]. Available: http://www.nist.gov/speech/tools

[21] J. B. Kruskal and D. Sankoff, "An anthology of algorithms and concepts for sequence comparisons," in *Time Warps, String Edits and Macromolecules: The Theory and practice of Sequence Comparison*, D. Sankoff and J. Kruskal, Eds. CSLI Publications, 1999, pp. 256–310.

[22] Q. Liu, “Interactive and incremental learning via a multisensory mobile robot,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 2001.

[23] *The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus, [CDROM]*, National Institute of Standards and Technology (NIST), USA, 1990.

[24] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.4)*. Cambridge, UK: Cambridge University Engineering Department, 2006.

[25] A. Cutler, A. Weber, R. Smits, and N. Cooper, “Patterns of english phoneme confusions by native and non-native listeners,” *Journal of the Acoustical Society of America*, vol. 116, no. 6, pp. 3668–3678, 2004.

[26] S. M. Witt and S. J. Young, “Phone-level pronunciation scoring and assessment for interactive language learning,” *Speech Communication*, vol. 30, no. 2-3, pp. 95–108, 2000.