Melody Recognition with Learned Edit Distances*

Amaury Habrard[1], José Manuel Iñesta[2], David Rizo[2], and Marc Sebban[3]

[1] Laboratoire d'Informatique Fondamentale, Université de Provence,
13453 Marseille cedex 13, France
[2] Dept. Lenguajes y Sistemas Informáticos, Universidad de Alicante,
E-03080 Alicante, Spain
[3] Laboratoire Hubert Curien, Université de Saint-Etienne,
42000 Saint-Etienne, France

Abstract. In a music recognition task, the classification of a new melody is often achieved by looking for the closest piece in a set of already known prototypes. The definition of a relevant similarity measure becomes then a crucial point. So far, the edit distance approach with a-priori fixed operation costs has been one of the most used to accomplish the task. In this paper, the application of a probabilistic learning model to both string and tree edit distances is proposed and is compared to a genetic algorithm cost fitting approach. The results show that both learning models outperform fixed-costs systems, and that the probabilistic approach is able to describe consistently the underlying melodic similarity model.

Keywords: Edit distance learning, music similarity, genetic algorithms, probabilistic models.

1 Introduction

In a music recognition task, the classification of a new melody is often achieved by looking for the closest piece in a set of already known prototypes. The definition of a relevant similarity measure becomes then a crucial point. Several approaches for the musical similarity computation can be found in the literature, such as geometric approaches [9], language models [6] or edit distance based-methods [2,7]. This last category has encountered a great success during the past few years. Actually, musical pieces can be naturally represented by symbolic representations such as strings or trees containing the sequence of notes in the melody. In this context, the edit distance is a performing similarity measure because it is able to directly work on the structure of the data and thus the melody structure itself.

Let us recall that the basic edit distance between two strings corresponds to the minimum number of primitive edit operations allowing us to transform one

* This work was funded by the French *ANR Marmota project*[1,3], the Spanish *PROSEMUS project* (TIN2006-14932-C02)[2], the research programme *Consolider Ingenio 2010* (MIPRCV, CSD2007-00018)[2], and the *Pascal Network of Excellence.*[1,2,3]

N. da Vitora Lobo et al. (Eds.): SSPR&SPR 2008, LNCS 5342, pp. 86–96, 2008.

input string into an output one. Three edit operations are usually allowed: the deletion of a symbol in the input string, the insertion of a symbol in the output one, or the substitution of a symbol in the input string by a letter in the output string. To take into account the relative importance of each edit operation, one usually assigns a cost to each of them. Therefore, the computation of the edit distance boils down to looking for the less costly transformation. String edit distance in music recognition has already been applied in former works (e.g. [3]) and it has been extended to the context of tree-structured data [7] where a tree representation of the melodies is utilized to compare the instances with tree edit distances, using fixed edit operation costs. In fact, whatever the edit distance we use, the problem of assigning relevant costs to achieve a performing music recognition task remains the crucial point. To tackle this problem, two solutions can be considered. First, one can use background knowledge and human expertise to fix them. For instance, Mongeau and Sankoff [2] compared melodies by using an extended string edit distance (with consolidation and fragmentation edit operations), where the edit costs are tuned according to musicological knowledge. The second strategy consists to automatically tune or learn the edit costs. One of the main motivations for learning the edit costs from a training set is to define models able to better estimate similarities between structured objects resulting in a better accuracy while performing pattern recognition tasks (as also done by approaches which learn Mahalanobis distances in the context of numerical vectors). In other words, such learned edit costs capture background knowledge that enable an improvement of recognition performances.

For example, in the music recognition field, Gómez et al. [3] posed the problem of edit cost tuning in string edit distances given a training set through an exhaustive space searching. Another approach was used in [4] by running a genetic algorithm to fit the edit costs. However, as it will be confirmed in our experiments, this method does not seem to provide an interpretable model from a music recognition standpoint, performing as a blackbox. To overcome this drawback, we suggest in this paper the use of recent probabilistic learning algorithms of edit distance that have not been exploited in music recognition so far, but that have already proved their efficiency in other pattern recognition tasks [14,16,17]. This type of models has another interesting advantage. It also allows the representation of understandable knowledge. For example, in music analysis, one can wonder, in variations of some musical tunes, what are the most common changes that makes the tune still recognizable. Our hypothesis is that those changes will have small edit costs, in opposition to high valued edit operations.

One solution for both learning efficiently edit costs and keeping the understandability of learned weights is to use a probabilistic framework. In this context, rather than fixing edit costs, one aims to learn a matrix of edit probabilities which provides higher meaning than absolute weights. These edit probabilities can be estimated by learning the parameters of *memoryless* joint transducers as initially introduced by Ristad and Yianilos [17]. However, this joint model has the known disadvantages of generative approaches [1]. To overcome this drawback, Oncina and Sebban [16] have proposed to learn a discriminative

memoryless transducer where conditional probabilities are directly estimated. This approach has shown its efficiency in handwritten digit recognition [16] and has been recently extended to tree-structured data [10,14]. Note that non memoryless models have been proposed during the past few years [11,13,15]. While these approaches allow us to model more complex phenomena, their understandability is more complicated because the knowledge is captured in several matrices of finite state machines. This explains why we will only deal in our following experiments with memoryless models.

In this paper, our objective is to apply the very efficient learning algorithms presented in [10,16], respectively on string and tree-based representations of musical pieces [7]. The quality of the system is assessed in terms of the two aforementioned qualities: classification accuracy and understandability of the induced model. The performance improvement is measured using the classical edit distances with a-priori fixed costs as baseline. The ability to extract significant information in the form of edit cost matrices is compared to a version of the classical edit distance algorithm where the costs have been fitted using a genetic algorithm.

The rest of this paper is organized as follows: The framework of edit distance learning is introduced in Section 2. Section 3 is devoted to the presentation of the musical corpus, the experimental setup and the analysis of the results. Finally, we conclude in Section 4.

2 Edit Distance and Learning

2.1 Notations and Classical Edit Distance

For sake of simplicity, we mainly focus on the string case but the notions presented here can be extended to trees. Strings are defined over a finite alphabet Σ and the empty symbol is denoted by $\lambda \notin \Sigma$. In this paper, we use letters in lowercase for symbols of $\Sigma \cup \{\lambda\}$ while letters in uppercase correspond to strings built from Σ.

The classical edit distance between two strings, also called the Levenshtein distance, is the minimal cost to transform by edit operations one input string into an output one. Three kinds of basic edit operations can be used in the transformation process: The insertion of a letter, the deletion of a letter and the substitution of a letter by another one. To each operation a so-called *edit cost* is assigned. Suppose, for example, that strings are built from an alphabet of two letters a and b. The costs for computing an edit distance can be represented by a 3×3 matrix c where the rows describe the input alphabet and the columns the output one. Therefore, $c(a, b)$ denotes the cost of applying the edit operation (a, b) where a is an input letter while b belongs to the output alphabet; If $a = \lambda$, the operation denotes an insertion; If $b = \lambda$ the operation is a deletion. Note that the operation (λ, λ) is not allowed.

Definition 1. *An edit script* $e = e_1 \cdots e_n$ *is a sequence of edit operations* $e_i = (a_i, b_i) \in (\Sigma \cup \{\lambda\}) \times (\Sigma \cup \{\lambda\})$ *allowing the transformation of a string* X *into*

a string Y. The cost of an edit script $\pi(e)$ is the sum of the costs of the edit operations involved in the script: $\pi(e) = \sum_{i=1}^{n} c(e_i)$.

Let $S(X, Y)$ be the set of all the scripts that enable the emission of Y given X, the edit distance between X and Y is defined by: $d(X, Y) = min_{e \in S(X,Y)} \pi(e)$.

The computation of such an edit distance can be computed in quadratic time using dynamic programming techniques.

Note that many extensions have been presented in the context of tree-structured data [18,19]. We will consider in this paper the algorithm of Selkow [18] which takes into account three basic edit operations: The substitution of a node by another one, the deletion of an entire subtree and the insertion of an entire subtree. The interested reader will find more details in [18].

2.2 Learning in a Stochastic Framework

Learning the parameters of an edit distance requires the use of an inductive principle. A possible approach is to use the maximum likelihood paradigm as used in [10,16,17]. In this context, the algorithms do not learn a true metric between strings but rather an edit similarity estimated from the probability of a transformation between two strings. To overcome the bias of generative models such as [17], a relevant strategy consists to directly learn a conditional model [16]. Therefore, the similarity between two strings X and Y is assessed from the estimation of $p(Y|X)$ corresponding to the probability of generating by edit operations an output Y given an input X.

Let $\delta(b|a)$ be the conditional probability to change the symbol a into an output letter b (note that $\delta(b|a)$ is in some way equivalent to the cost $c(a, b)$ in the stochastic framework). To learn a conditional distribution over the edit operations, the δ function must fulfill the following condition of a statistical distribution:

$$\forall a \in \Sigma, \sum_{b \in \Sigma \cup \{\lambda\}} (\delta(b|a) + \delta(b|\lambda)) = 1.$$

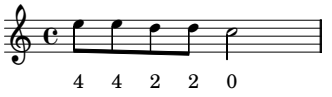
Let us now extend the notion of edit script in the probabilistic framework.

Definition 2. *A stochastic edit script $e = e_1 \cdots e_n$ is a sequence of edit operations $e_i = (b_i|a_i)$ allowing the transformation of a string X into a string Y. The probability $\pi_s(e)$ of such a script corresponds to the product of the probabilities of the edit operations involved in the script:* $\pi_s(e) = \prod_{i=1}^{n} \delta(e_i)$.

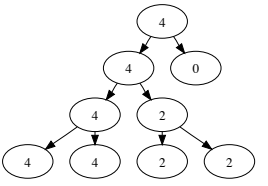
Definition 3. *The probability $p(Y|X)$ of generating a string Y given an input string X is the sum of the probabilities of all the edit scripts transforming X into Y:* $p(Y|X) = \sum_{e \in S(Y|X)} \pi_s(e)$.

One can remark that in this stochastic framework, we consider all the edit scripts to estimate $p(Y|X)$. Given $p(Y|X)$, one can compute an edit similarity between X and Y.

Definition 4. *The edit similarity between two strings X and Y, conditionally to X, is defined as:* $sim(X, Y|X) = -\log p(Y|X)$.



(a) Original melody. The figures below the score indicate the Σ_p (interval from tonic).



(b) Corresponding tree.

Fig. 1. Original tune

The learning of the parameters (*i.e.* the δ matrix) is done by an adaptation of the well known *expectation-maximization* algorithm [12] that aims to learn the hidden parameters of a probabilistic model by maximization the likelihood of a learning sample composed of pairs of $(input, output)$ strings.

3 Experiments and Analysis in Melody Recognition

3.1 Representation of Music and Database

In our series of experiments, we use both string and tree representations of music pieces [7]. For representing a monodic melody s as a string, symbols σ_i from a pitch description alphabet Σ_p are used: $s \in \Sigma_p^*, s = \sigma_1\sigma_2...\sigma_{|s|}$. In this paper, the interval from the tonic of the song is utilized as pitch descriptor (Fig. 1a): $\Sigma_p = \{p \mid 0 \leq p \leq 11\}$. In the tree representation (Fig. 1b), the node labels are symbols from the same alphabet Σ_p. Initially, only leaf nodes are labelled. Then, a melodic analysis [8] is used to decide which notes are more important in order to implement a bottom-up propagation process until all the inner nodes are labelled.

In our experiments, we used a corpus consisting of a set of 420 monophonic 8-12 bar incipits of 20 worldwide well known tunes of different musical genres [1]. For each song, a canonic version was created by writing the score in a musical notation application and exported to MIDI and MP3 format. The MP3 files were given to three amateur and two professional musicians who listened to the songs (mainly to identify the part of the tune to be played) and played with MIDI controllers the same tune several times with different embellishments and capturing performance errors. This way, for each of the 20 original scores, 20 different variations have been built.

3.2 Fitting the Edit Costs by Genetic Algorithms

One of the objectives of the paper is to compare the probabilistic edit distance learning methods to other existing cost fitting approaches that have already been

[1] The MIDI data is available upon request to the authors.

used in music recognition. In this context, a conventional genetic algorithm, as introduced in [4], has been used. The objective of this algorithm is to find the best insertion, deletion and substitution costs to be used in the classical edit distance. In our series of experiments, these costs can be represented by a matrix of size $13 \times 13 - 1 = 168$ (the operation (λ, λ) is not allowed) that represents the cost of the substitution of each of the 12 pitches for any other pitch, plus the insertion and the deletion of each pitch. Individuals containing the cost matrix in the form of vector are represented as chromosomes with 168 genes, one for each cost to be optimized, where the genes are real-valued in the range [0,2].

The fitness of each chromosome is evaluated experimentally by training and testing a classifier using the edit costs contained in the chromosome, where better classifier performance yields higher fitness. The fitness function has been computed as the averaged *precision-at-n* for all the melodies in the training set. The *precision-at-n* corresponds to the number of relevant hits among the n closest melodies, n being the number of correct prototypes in the learning set. The system has been implemented with an open-source robust GA software named JGAP [5]. We used the JGAP default settings that include mutation rate and selection method. Note that the algorithm does not stop until all the specified evolutions are finished. In the experiments, the number of evolutions has been empirically set to 100 from the observation of the fitness function evolution curve.

3.3 Melody Classification Accuracy

The main goal of our first series of experiments is to identify a melody from the set of all the different variations played by the musicians. For a melody, the edit distances to all the other prototypes are computed and the 1-NN rule was applied. A 3-fold cross-validation scheme has been followed to perform the experiments. This evaluation process has been achieved from both string and tree representations of the data.

The average error rates are shown in Fig.2. These results show that both the stochastic and the genetic methods outperform the results of the classical edit distance with insertion and deletion costs fixed to 1, and substitution fixed to 2, in the case of strings. The improvement is even much more significant for trees, while comparing with a standard algorithm with all the costs set to 1. We can note that the genetic algorithm seems to be a little bit more efficient than the stochastic method from the only accuracy point of view.

However, to improve the music recognition systems, we have to develop methods that are able not only to improve the classification accuracy of standard edit distance algorithms, but also to provide understandable knowledge explaining the distortions of the melodies. This is the aim of our second series of experiments.

3.4 Analysis of the Learned Edit Matrices

To analyze the understandability of the inferred models, we compared the learned cost matrices obtained with both genetic and probabilistic approaches. In order

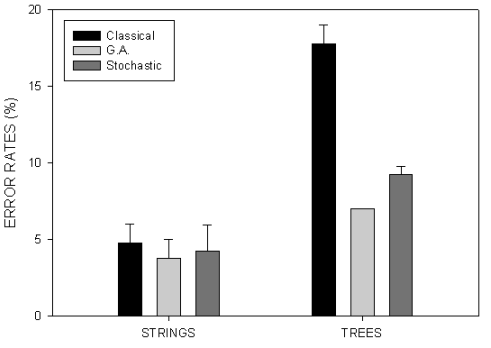
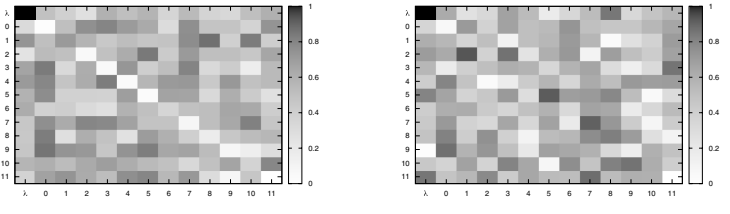


Fig. 2. Error rates from both string and tree representations

to represent graphically these matrices, they have been averaged and normalized over the three folds. To allow the comparison, the conditional probabilities of the stochastic model have been translated into costs using negative logarithms. Our aim here, is to see if the learned edit costs are able to model the changes and mistakes that can be found in the corpus. By analyzing it, the mistakes correspond in general to note elisions, grace notes insertions, the substitution of the intended note by another at a distance of a semitone or a tone, and note duration and onset changes.

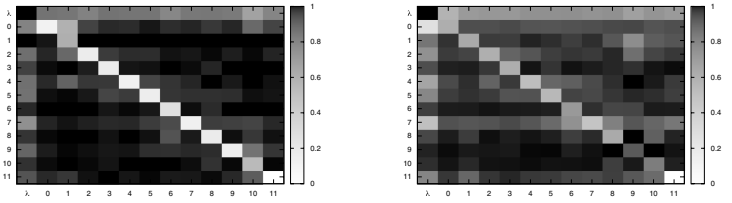
The matrices learned with the genetic approach are shown in Fig. 3. From the lighter gray diagonal in the plot in Fig. 3a it can be drawn that the smallest costs are found in the substitution of a pitch for itself as it could be expected. No other pattern can be found that explains the most common differences between the original songs with its variations. In the case of the learned costs for trees (Fig. 3b), nor even the diagonal can be identified. From these results, it can be stated that the genetic approach to fit costs is useful to improve the success rates but not to learn a readable model of the melodic similarity phenomena.

The learned edit probabilities in the form of edit costs have been plotted in Fig. 4. In the case of strings (Fig. 4a), both the diagonal with the identity substitution cost and the insertion row (top) and deletion column (left-most) are



(a) String edit costs. (b) Tree edit costs.

Fig. 3. Learned edit costs by a genetic algorithm

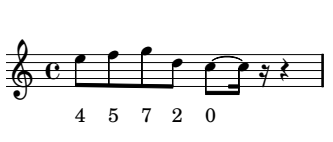


(a) String edit costs. (b) Tree edit costs.

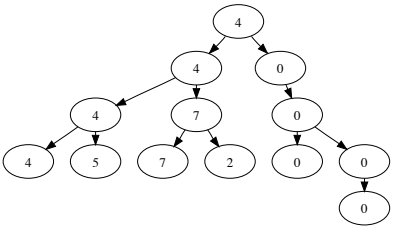
Fig. 4. Learned edit costs by a stochastic algorithm

evident. These insertions and deletion reflect changes in note onset and durations. Moreover, the values close to the diagonal are also given higher probabilities, showing that the stochastic system has the ability of capturing changes of semitones made in the playing of the songs as pointed above in Sec. 3.1. The tree learned costs in Fig. 4b contain more or less the same scheme even if the diagonal is less dark. This behavior can be easily explained by the restriction imposed by Selkow's learning algorithm [10] used in the tree case: It allows only the insertions and deletions of whole subtrees which implies that deletion and insertion operations cover a larger part of the probability density of edit operations.

Based on the previous observations, it is much more difficult to understand the mapping from the original version to the variation using the matrices generated by the genetic algorithm. The reason why the genetic algorithm is unable to produce an understandable set of costs in opposite to the probabilistic approach seems to be due to the fact that in the classical edit distance computations, only the best path in the dynamic programming matrix is retained, and the genetic algorithm just tries to minimize the cost of one path for each prototype in the training set. This is only a local minimum in the parameter space. On the other hand, being the stochastic distance a sum of probabilities of all possible paths, all those probabilities must remain high (*i.e.* with low costs), being more likely to assess the importance of each operation. Thus, this method is more capable to capture relevant knowledge even with little corpora.



(a) Variation of tune in Fig.1a.



(b) Corresponding tree.

Fig. 5. Variation of tune in Fig.1

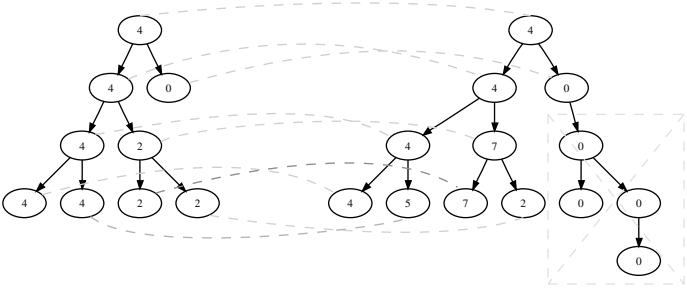


Fig. 6. Edit operations between the original tune and one of its variations. The deleted nodes are inside a rectangle. Gray levels of edit operations follow values in the costs matrix in Fig. 3b.

To illustrate the adequacy of the proposed approach to the melody similarity problem, two different variations of the same bar of the same song are shown in Figs. 1 and 5. The most probable edit script is indicated in Fig. 6 with dashed lines. The gray level of these dashed lines is set according to the gray levels in the matrix of Fig. 3b. Note that the structure of the tree is preserved, the substitution costs and the deletion of the nodes with a label "0" (tonic note) are the most probable option given the cost matrix. The string edit script, that can be deduced easily from the strings indicated below the scores in Figs. 1a and 5a, follow the same substitution and deletion pattern. Although not all the examples in the dataset are so easy to describe, in general the same process as the one we have just exposed can be followed.

4 Conclusions and Future Work

The ability for machine learning approaches to provide interpretable models is a desirable quality not frequently found. In this paper a recent algorithm of edit distance learning has been applied to the melodic similarity problem. It has been compared to both edit distance systems with a-priori fixed costs and to a genetic algorithm that fits the edit weights through the exploration of the cost space. It has been shown that both proposed methods to learn the edit costs, genetic and probabilistic schemes, are better than using initially fixed costs.

Moreover, the probabilistic approach has been shown to be more adequate for explaining the variation or noise introduction process being able to produce a readable cost matrix coherent with the training corpus. With a bigger corpus maybe the genetic approach will generate more understandable cost matrices, but also the parameters used in stochastic distance learning approaches will be more precisely assessed.

In the future, we plan to consider more pitch representations such as contour, high definition contour, pitch classes, and intervals to evaluate their adequacy to the melody matching problem. Another perspective is to work on the more

frequent polyphonic music. Thus, one of the most interesting lines of work to be followed is the extension of the presented method to the polyphonic similarity computation. In this problem, the tree representation is the most suited to accomplish the task which brings us to focus on tree edit distance learning.

References

1. Bouchard, G., Triggs, B.: The trade-off between generative and discriminative classifiers. In: COMPSTAT 2004. Springer, Heidelberg (2004)
2. Mongeau, M., Sankoff, D.: Comparison of musical sequences. Computers and the Humanities 24, 161–175 (1990)
3. Gómez, C., Abad-Mota, S., Ruckhaus, E.: An Analysis of the Mongeau-Sankoff Algorithm for Music Information Retrieval. In: Proc. of the 8th International Conference on Music Information Retrieval (ISMIR), Vienna, Austria, pp. 109–110 (2007)
4. Grachten, M., Arcos, J., López de Mántaras, R.: Melody Retrieval using the Implication/Realization Model. In: Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR 2005 (2005)
5. Rotstan, N., Meffert, K.: JGAP: Java Genetic Algorithms Package (accessed June 7, 2008), `http://www.jgap.sf.net`
6. Doraisamy, S., Rüger, M.: Robust Polyphonic Music Retrieval with N-grams. J. Intell. Inf. Syst. 21, 53–70 (2003)
7. Rizo, D., Moreno-Seco, F., Iñesta, J.M.: Tree-structured representation of musical information. In: Perales, F.J., Campilho, A.C., Pérez, N., Sanfeliu, A. (eds.) IbPRIA 2003. Lecture Notes in Computer Science (LNAI), vol. 2652, pp. 838–846. Springer, Heidelberg (2003)
8. Illescas, P.R., Rizo, D., Iñesta, J.M.: Harmonic, melodic, and functional automatic analysis. In: Proc. International Computer Music Conference, pp. 165–168 (2007)
9. Aloupis, G., Fevens, T., Langerman, S., Matsui, T., Mesa, A., Nuñez, Y., Rappaport, D., Toussaint, G.: Algorithms for Computing Geometric Measures of Melodic Similarity. Computer Music Journal, Fall 2006 30(3), 67–76 (2006)
10. Bernard, M., Habrard, A., Sebban, M.: Learning stochastic tree edit distance. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS, vol. 4212, pp. 42–53. Springer, Heidelberg (2006)
11. Bernard, M., Janodet, J.-C., Sebban, M.: A discriminative model of stochastic edit distance in the form of a conditional transducer. In: Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E. (eds.) ICGI 2006. LNCS, vol. 4201, pp. 240–252. Springer, Heidelberg (2006)
12. Dempster, A., Laird, M., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. B(39), 1–38 (1977)
13. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological sequence analysis. Cambridge University Press, Cambridge (1998)
14. Boyer, L., Habrard, A., Sebban, M.: Learning metrics between tree structured data: Application to image recognition. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS, vol. 4701, pp. 54–66. Springer, Heidelberg (2007)

15. McCallum, A., Bellare, K., Pereira, P.: A conditional random field for discriminatively-trained finite-state string edit distance. In: Proc. of UAI 2005, pp. 388–400 (2005)
16. Oncina, J., Sebban, M.: Learning stochastic edit distance: application in handwritten character recognition. J. of Pattern Recognition 39(9), 1575–1587 (2006)
17. Ristad, S.V., Yianilos, P.N.: Learning string-edit distance. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(5), 522–532 (1998)
18. Selkow, S.M.: The tree-to-tree editing problem. Information Processing Letters 6(6), 184–186 (1977)
19. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal of Computing, 1245–1262 (1989)