Pattern Recognition 44 (2011) 1172–1182

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A sum-over-paths extension of edit distances accounting for all sequence alignments

Silvia García-Díez [a,*], François Fouss [b], Masashi Shimbo [c], Marco Saerens [a]

[a] Université de Louvain, ISYS, LSM, Louvain-la-Neuve, Belgium
[b] Facultés Universitaires Catholiques de Mons, Management Science Department, LSM, Belgium
[c] Graduate School of Information Science, Nara Institute of Science and Technology, Japan

ARTICLE INFO

Article history:
Received 7 October 2009
Received in revised form
29 November 2010
Accepted 30 November 2010
Available online 3 December 2010

Keywords:
Edit distance
Longest common subsequence
Sequence comparison
Approximate string matching
Dynamic programming
Viterbi algorithm
Shortest path

ABSTRACT

This paper introduces a simple Sum-over-Paths (SoP) formulation of string edit distances accounting for all possible alignments between two sequences, and extends related previous work from bioinformatics to the case of graphs with cycles. Each alignment $\wp$, with a total cost $C(\wp)$, is assigned a probability of occurrence $P(\wp)=\exp[-\theta C(\wp)]/\mathcal{Z}$ where $\mathcal{Z}$ is a normalization factor. Therefore, good alignments (having a low cost) are favored over bad alignments (having a high cost). The expected cost $\sum_{\wp\in\mathcal{P}} C(\wp)\exp[-\theta C(\wp)]/\mathcal{Z}$ computed over all possible alignments $\wp\in\mathcal{P}$ defines the SoP edit distance. When $\theta\to\infty$, only the best alignments matter and the measure reduces to the standard edit distance. The rationale behind this definition is the following: for some applications, two sequences sharing many good alignments should be considered as more similar than two sequences having only one single good, optimal, alignment in common. In other words, sub-optimal alignments could also be taken into account. Forward/backward recurrences allowing to efficiently compute the expected cost are developed. Virtually any Viterbi-like sequence comparison algorithm computed on a lattice can be generalized in the same way; for instance, a SoP longest common subsequence is also developed. Pattern classification tasks performed on five data sets show that the new measures usually outperform the standard ones and, in any case, never perform significantly worse, at the expense of tuning the parameter θ.

1. Introduction

1.1. General introduction

Approximate string matching (ASM) algorithms find applications in various areas, such as pattern recognition [57], computer science [12,54], bioinformatics [17,29,51], and computational linguistics [27]. Numerous extensions have been proposed, allowing either to improve the performance of the basic techniques or to extend their applicability to new problems for which the basic techniques are not suited.

The aim of this work is to extend, in a straightforward way, the basic ASM algorithms in the following way. Most of the ASM algorithms return the score provided by the best, optimal, alignment between two sequences. By optimal, we mean the alignment corresponding to the least total cost, which can be computed by the well-known Viterbi algorithm [20]. Now, instead of only relying on the best alignments, we propose to average the total cost over all the possible alignments. Indeed, as noticed in [17], relevant information is also contained in the sub-optimal paths. For some applications, two sequences sharing many good alignments should be considered as more similar than two sequences having only one single good, optimal, alignment in common. To this end, we adopt a Sum-over-Paths (SoP) formalism, considering that each alignment corresponds to a path on the dynamic programming lattice. This formalism is directly inspired by [49], which was itself inspired by the work of Akamatsu in transportation sciences [3]. The same idea has already appeared in bioinformatics for acyclic lattices [40,70] (see Section 1.2). The SoP generalizes this idea to arbitrary graphs.

The basic idea is to consider the set of all possible alignments and to weight their contribution to the total cost according to their respective quality. Each alignment is weighted by a probability mass penalizing bad alignments, so that the contribution of better alignments is more important. Concretely, the alignment choice is *weighted* by imposing a pre-defined entropy level. The probability distribution on the set of alignments is defined as the one minimizing the total expected cost for the predefined entropy. It turns out that this optimal probability distribution is a Boltzmann distribution on the set of alignments (see Section 2.3 or [49]), depending on a parameter θ—the inverse temperature which is inversely related to the entropy – controlling the trade-off between the contribution of good and bad alignments. The SoP edit distance

* Corresponding author.
E-mail addresses: silvia.garciadiez@uclouvain.be (S. García-Díez), francois.fouss@fucam.ac.be (F. Fouss), shimbo@is.naist.jp (M. Shimbo), marco.saerens@uclouvain.be (M. Saerens).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.11.020

is then defined as the total expected cost computed over all possible alignments, the expectation being taken according to the Boltzmann distribution.

It is shown that this total expected cost can be easily computed within this framework. The resulting algorithm is similar to the forward/backward algorithm used in training hidden Markov models: the recurrence relations allowing to compute the different quantities in an efficient way are readily derived. In particular, our algorithm reduces to the standard forward/backward algorithm when the θ parameter is equal to 1. On the other hand, when $\theta \to \infty$, it corresponds to the Viterbi algorithm and thus reduces to the basic algorithm computing only the best alignments. Two standard edit distance algorithms are extended within this framework: the Levenshtein edit distance (LED) and the longest common subsequence (LCS); they will respectively be called the SoP edit distance (SoP ED) and the SoP common subsequence similarity (SoP CS).

1.2. Related work

As discussed in [49], the idea of randomizing the policy was introduced by [1,2] in the context of reinforcement learning and was inspired by the entropy rate of an ergodic Markov chain defined in information theory (see, e.g., [13]). In this previous work, the sum of the local entropies on all nodes was fixed and the optimal policy for this fixed level of entropy was computed through a value-iteration-like algorithm.

In [49], instead of fixing the local entropy defined at each node as in [1,2], the global entropy spread over the whole network is fixed. While this difference seems a priori insignificant, it appears that constraining the global spread entropy of the network is more natural and easier to compute. Clearly, the nodes that need a large spread are difficult to determine in advance, and the model has to distribute the entropy by optimizing it globally all over the network. It was shown in [49] that the optimal randomized policy can be found by solving a simple system of linear equations. In the same paper, the authors showed that when the graph is acyclic—and therefore a lattice—the expected cost as well as the policy can be computed efficiently from two simple recurrence relations. This fact is exploited in this paper in order to define a randomized edit distance.

Apart from this previous work [1,2,49], and some others in game theory (see for instance [42]) or Markov games [34], very few optimal randomized strategies have been exploited in the context of shortest-path problems. There is, however, one exception: Akamatsu's model [3], who designed a randomized policy for routing traffic in transportation networks. In transportation science, randomized strategies are called *stochastic traffic assignments* and, within this context, Akamatsu's model is the model of reference. This work, as well as [49], are inspired by Akamatsu's model.

Let us also mention some papers that are related to path randomization, and therefore to the present work. The entropy of the paths (or trajectories) connecting an initial and an absorbing destination node of an absorbing Markov chain was studied in [18]. In this paper, the authors provide formulae allowing to compute the entropy needed to reach the destination node. In [56] a one-parameter family of algorithms that recover both the procedure for finding shortest paths as well as the iterative algorithm for computing the average first-passage time in a Markov chain is introduced. However, having a heuristic foundation, it is not based on the optimization of a well-defined cost function. In another context, Todorov [60] studied a family of Markov decision problems that are linearly solvable, that is, for which a solution can be computed by solving a matrix eigenvector problem. In order to make this possible, Todorov assumes a special form of control for the transition probabilities, which recasts the problem of finding the policy into an eigenvector problem. In [9] a Markov chain that has the fastest mixing properties is designed, while [55] discuss its continuous-time counterpart. In a completely different framework, uninformed random walks, based on maximizing the long-term entropy [15,61] have recently been proposed as an alternative to the standard PageRank algorithm. Finally, notice that some authors tackled the problem of designing ergodic (non-absorbing) Markov or semi-Markov chains in a maximum-entropy framework (see for instance [21,22] and the references therein).

Within the context of computing edit distances between sequences, there have been successful attempts to account for all possible alignments (see, e.g., [10,17,28,31,35,48,52,68], among others). The sequence comparison model introduced in [17] is quite popular in bioinformatics. It is based on a hidden Markov model (HMM) of sequence pairs generation, called the pair HMM. The model is trained by maximum likelihood on a sequences sample and, once is trained, it provides the likelihood of observing any two sequences, each alignment being weighted by its probability. The SoP edit distance introduced in this paper shares therefore some similarities with the pair HMM, but is much simpler since it does not require any transition-probability estimation, i.e., it is model-free, although by modifying the edition costs or the similarity measure we could adapt it to any model or specific task. It is, however, also possible to fix a priori the transition probabilities of the pair HMM. In that case, the SoP edit distance is equivalent to the pair HMM with a suitable choice of the editing costs, and parameter θ equal to 1. However, two important differences between the proposed SoP techniques and pair HMMs is that (i) it weights the contribution of the different alignments, according to their respective total costs, and (ii) it depends on a parameter allowing to regulate the degree of exploration. It therefore encompasses both the Viterbi and the Baum–Welch algorithms as special cases. It will be observed in the experimental section that the best performance obtained by the SoP techniques is often achieved for θ parameter values greater than 1. Finally, notice that a valid kernel derived from a pair HMM was proposed in [26,68].

The string kernels introduced in [31,33,35,48,50,52] compute a score depending on all the possible common subsequences of the two compared sequences. The main idea behind these sequence-comparison techniques is to reward common subsequences, contiguous or not, depending on the technique. However, when the size of the alphabet is very low, there is a huge quantity of such short common subsequences and very few long common subsequences, so that the obtained score does not reflect accurately the proper similarity between the two sequences. Indeed, a much larger weight should be put on long common subsequences; this is exactly what the SoP edit distance does. We therefore expect string kernels to perform well when the alphabet is quite large (for instance in the context of text mining—in this case—there are very few long common subsequences; see for example [11]), and worse in the case of reduced alphabets—this is indeed observed in our experiments.

Yet another approach computing an edit score along all possible alignments is the stochastic edit distance, and related methods [6]. The underlying model is based on a noisy channel. An input string $\mathbf{x}$—the reference string or code—is distorted by a noisy channel, therefore producing an output string $\mathbf{y}$ that is a noisy transform of $\mathbf{x}$. The noisy channel is often modeled as a Markov model or a transducer [6,41]. For these models, the probability of generating string $\mathbf{y}$ from $\mathbf{x}$, $P(\mathbf{y}|\mathbf{x})$, can be computed thanks to a forward recursion formula involving all possible ways of generating $\mathbf{y}$, and therefore all possible paths through the lattice. The scores $-\log P(\mathbf{y}|\mathbf{x})$ or $-\log P(\mathbf{x}|\mathbf{y})$ can be considered as dissimilarity measures between $\mathbf{x}$ and $\mathbf{y}$. This stochastic edit distance model is refined in [47,4,41], where the local distances between the symbols are estimated from sample data. A paragraph explaining the relationships between stochastic edit distances and the sum-over-paths approach appears at the end of Section 2.4. However, these