ON STOCHASTIC TREE DISTANCES AND THEIR TRAINING VIA EXPECTATION-MAXIMISATION

Martin Emms
School of Computer Science and Statistics, Trinity College, Dublin, Ireland
mtemms@tcd.ie

Keywords: Tree-distance, Expecation-Maximisation.

Abstract: Continuing a line of work initiated in (Boyer et al., 2007), the generalisation of stochastic string distance to a stochastic tree distance is considered. We point out some hitherto overlooked necessary modifications to the Zhang/Shasha tree-distance algorithm for all-paths and viterbi variants of this stochastic tree distance. A strategy towards an EM cost-adaptation algorithm for the all-paths distance which was suggested by (Boyer et al., 2007) is shown to overlook necessary ancestry preservation constraints, and an alternative EM cost-adaptation algorithm for the Viterbi variant is proposed. Experiments are reported on in which a distance-weighted kNN categorisation algorithm is applied to a corpus of categorised tree structures. We show that a 67.7% base-line using standard unit-costs can be improved to 72.5% by the EM cost adaptation algorithm.

1 INTRODUCTION

The classification of tree structures into categories is necessary in many settings. In natural language processing an example is furnished by Question-Answering systems, which frequently have a Question-Categorisation sub-component, whose purpose is to assign the question to one of a number of predefined semantic categories (section 5 gives some details). Often one would like to obtain such a classifier by a data-driven machine-learning approach, rather than by hand-crafting one. The *distance*-based approach to such a classifier is to have a pre-categorised *example set* and to compute a category for a test item based on its distances to examples in the example set, such as via k-NN.

With items to be categorised represented as trees, a crucial component in such a classifer is the measure used to compare trees. The tree-distance first proposed by (Tai, 1979) is a well motivated candidate measure (see later for further details). This measure can be seen as composing a relation between trees out of several kinds of atomic operations (match, swap, delete, insert) the costs of which are dependent on the labels of the nodes involved. The performance of such a distance-based classifier is therefore very dependent on the settings of these atomic costs.

In the work presented below probabilistic variants of the standard tree-distance are considered and then Expectation-Maximisation techniques are considered as potential means to adapt atomic costs given a corpus of training tree pairs.

Section 2 recalls the standard definitions of string- and tree-distance. Section 3 goes on to first recall the stochastic string-distance as proposed by (Ristad and Yianilos, 1998) and then defines stochastic variants of tree-distance, which we term *All-scripts* and *Viterbi-script stochastic Tai distances* $\Delta^a_s(S,T)$ and $\Delta^v_s(S,T)$. The standard algorithm of (Zhang and Shasha, 1989) for the tree-distance is then recalled and some necessary modifications to allow correct and efficient computation of the All-scripts and Viterbi variants are described. Section 4 concerns how one might adapt atomic costs from a training corpus of same-category neighbours via Expectation-Maximisation. The string-case is recalled and then a brute-force, exponentially expensive method for adapting the parameters of the *All-scripts* distance is outlined. Whilst for the linear case, a particular factorisation is applicable which permits efficient implementation of the EM all-scripts method, we show by means of a counter-example that for the tree case this kind of factorisation is not applicable, although it has been suggested in (Boyer et al., 2007) that it is. This leads to a proposed method for adapting the parameters of the *Viterbi-script* distance, something which is feasible. Section 5 then reports some experimental outcomes obtained with this EM procedure for adapting the Viterbi-script distance.

Emms M. (2012).
ON STOCHASTIC TREE DISTANCES AND THEIR TRAINING VIA EXPECTATION-MAXIMISATION.
In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, pages 144-153
DOI: 10.5220/0003864901440153

2 NON-STOCHASTIC SEQUENCE AND TREE DISTANCES

Let us begin by formulating some definition relation to the familiar notion of string-distance. The formulations follow closely those of (Ristad and Yianilos, 1998) and are chosen to allow an easy transition to the stochastic case.

Let Σ be an alphabet. Let the set of edit operation identifiers, $EdOp$, be defined by

$$EdOp = ((\Sigma \cup \{\lambda\}) \times (\Sigma \cup \{\lambda\})) \setminus \langle \lambda, \lambda \rangle$$

and let an *edit-script* be a sequence $e_1 \ldots e_n\#$, $n \geq 0$, with each $e_i \in EdOp$ and with # as a special end-of-script marker. Given an edit-script $\mathcal{A}$, it can be projected into a 'source' string $src(\mathcal{A}) \in \Sigma^*$, by contenating the left elements of the contained operations, and likewise into a 'target' string $trg(\mathcal{A}) \in \Sigma^*$:

$$\begin{array}{ll} src(\#) = \varepsilon & trg(\#) = \varepsilon \\ src(\langle \lambda, y \rangle \mathcal{A}) = src(\mathcal{A}) & trg(\langle x, \lambda \rangle \mathcal{A}) = trg(\mathcal{A}) \\ src(\langle x, y \rangle \mathcal{A}) = x\; src(\mathcal{A}) & trg(\langle x, y \rangle \mathcal{A}) = y\; trg(\mathcal{A}) \end{array}$$

The *yield* of edit-script $\mathcal{A}$ can be defined as pair of strings $\langle src(\mathcal{A}), trg(\mathcal{A}) \rangle$. If $(s,t) = yield(\mathcal{A})$, each $e_i \in \mathcal{A}$ is interpretable as an edit operation in a process of transforming s to t: deletion (a, λ), insertion (λ, b), or match/substitution (a, b). Let $E(s,t)$ be all scripts which yield (s,t). The multiple scripts in $E(s,t)$ describe alternative ways to transform s into t.

If costs are defined for such edit-scripts, a string-distance between s and t can be defined as the cost of the least-cost script in $E(s,t)$.

Alternatively one can consider the *partial*, *1-to-1*, *order-respecting* mappings from s to t. Costs can be defined for such mappings and a distance measure defined via minimising these costs. Script-based and mapping-based definitions are equivalent (Wagner and Fischer, 1974): fundamentally an edit-script is viewable as a particular serialisation of a mapping.

For ordered, labelled trees, the analogue to standard string distance was first considered by (Tai, 1979). We develop the definitions relevant to this below, starting first with a *mapping*-based definition. The equivalent *script*-based definition follows this.

A *Tai* mapping is a *partial*, *1-to-1* mapping σ from the nodes of source tree S to a target tree T which respects *left-to-right order* and *ancestry*[1]. For the purpose of assigning a score to such a mapping it is convenient to identify three sets:

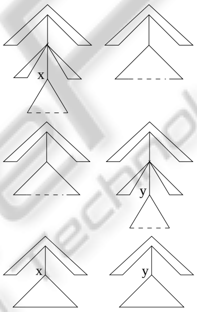
$\mathcal{M}$	the $(i,j) \in \sigma$: the 'matches' and 'swaps'
$\mathcal{D}$	the $i \in S$ s.t. $\forall j \in T, (i,j) \notin \sigma$: the 'deletions'
$\mathcal{I}$	the $j \in T$ s.t. $\forall i \in S, (i,j) \notin \sigma$: the 'insertions'

Where Σ is the label alphabets of source and target trees, let $\gamma(i)$ be the label of node i. Let $\mathcal{C}$ be a *cost table* of dimensions $|\Sigma|+1 \times |\Sigma|+1$. The cost of a mapping is the sum over the atomic costs defined by[2]

for $(i,j) \in \mathcal{M}$	cost is $\mathcal{C}[\gamma(i)][\gamma(j)]$
for $i \in \mathcal{D}$	cost is $\mathcal{C}[\gamma(i)][0]$
for $j \in \mathcal{I}$	cost is $\mathcal{C}[0][\gamma(j)]$

The so-called *unit-cost* matrix, $\mathcal{C}_{01}$ has 0 on the diagonal and 1 everywhere else. For a given cost matrix $\mathcal{C}$, the *Tai*- or *tree-distance* $\Delta(S,T)$ is defined as the cost of the least-costly Tai mapping σ between S and T.

There is an alternative, more procedural definition route, via *tree*-edit operations analogous to string-edit operations. The table below depicts the three edit operations:



operation	*script element*
$(m\; \vec{l}\; (\mathbf{x}\; \vec{d})\; \vec{r}) \rightarrow (m\; \vec{l}\; \vec{d}\; \vec{r})$	(x, λ)
$(m\; \vec{l}\; \vec{d}\; \vec{r}) \rightarrow (m\; \vec{l}\; (\mathbf{y}\; \vec{d})\; \vec{r})$	(λ, y)
$(m\; \vec{l}\; (\mathbf{x}\; \vec{d})\; \vec{r}) \rightarrow (m\; \vec{l}\; (\mathbf{y}\; \vec{d})\; \vec{r})$	(x, y)

Thus deletion involves making the daughters of some node x into the daughters of that node's parent m, insertion involves taking some of the daughers of a node m and making them instead the daughters of a new daughter y of m, and swapping/matching involves simply replacing some node x with a node y at the same position.

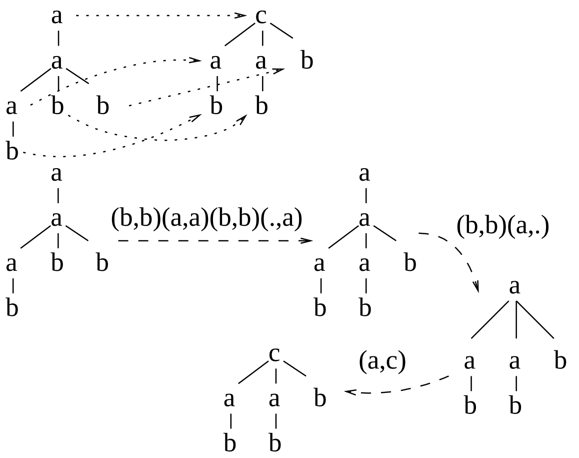
The right-hand column shows the script element which is used to record the use of particular edit operation. Using the same table of cost $\mathcal{C}$ as was used for costing a mapping, a cost can be assigned to the script describing the operations to transform a tree S into a tree T, and a script-based definition of distance then given via minimising this cost.

The mapping- and script-based definitions are equivalent (Tai, 1979; Kuboyama, 2007), with a script serving as a serialised representation of a mapping. (Zhang and Shasha, 1989) provided an efficient algorithm for its calculation.

[1] So if (i_1, j_1) and (i_2, j_2) are in the mapping, then (T1) $left(i_1, i_2)$ iff $left(j_1, j_2)$ and (T2) $anc(i_1, i_2)$ iff $anc(j_1, j_2)$

[2] See also (Emms and Franco-Penya, 2011) in these proceedings.

To illustrate, below is shown first a mapping between two trees, and second the sequence of edit-operations corresponding to it, with some of the intermediate stages as these operations are applied; with unit-costs the distance is 3[3]:



It is easy to see that for strings encoded as linear, vertical trees, the string-distance and tree-distance coincide. We will use tree-distance and Tai-distance interchangeably, though the literature contains several other, non-equivalent notions bearing the name *tree-distance*. Based, as it is, simply on the notion of mappings respecting the two defining dimensions of trees, the Tai distance seems a particularly compelling notion.

3 STOCHASTIC SEQUENCE AND TREE DISTANCES

(Ristad and Yianilos, 1998) introduced a probabilistic perspective on string distance, defining a model which assigns a probability to every possible edit-script. Edit-script compoments $e_i \in EdOp \cup \{\#\}$ are pictured as generated in succession, independently of each other. There is an emission probability p on edit-script components, such that $\sum_{e \in EdOp \cup \{\#\}} p(e) = 1$, and a script's probability is defined by

$$P(e_1 \ldots e_n) = \prod_i p(e_i)$$

For a given string pair (s,t), as before $E(s,t)$ denotes all the edit-scripts which have (s,t) as their yield. They then define the *all-paths stochastic edit distance*, $\Pi^a(s,t)$, as the sum of the probabilities of all scripts $e \in E(s,t)$, whilst the *viterbi* version $\Pi^v(s,t)$ is the probability of the most probable one.

It is natural to consider to what extent the probabilistic perspective adopted for string-distance by Ristad and Yianilos can be applied to tree-distance. The simplest possibility is to use exactly the same model of edit-script probability, which leads to the notions:

Definition 1 (All-scripts stochastic Tai similarity/distance)**.** *The all-scripts stochastic Tai similarity, $\Theta^A_s(S,T)$, is the sum of the probabilities of all edit-scripts which represent a Tai-mapping from S to T. The all-scripts stochastic Tai distance, $\Delta^A_s(S,T)$, is its negated logarithm, ie.*

$$2^{-\Delta^A_s(S,T)} = \Theta^A_s(S,T)$$

Definition 2 (Viterbi-script stochastic Tai similarity/distance)**.** *The Viterbi-script stochastic Tai similarity, $\Theta^V_s(S,T)$, is the probability of the most probable edit-script which represents a Tai-mapping from S to T. The Viterbi-script stochastic Tai distance, $\Delta^V_s(S,T)$, is its negated logarithm, ie.*

$$2^{-\Delta^V_s(S,T)} = \Theta^V_s(S,T)$$

For Θ^A_s and Θ^V_s the probabilities on each possible component of an edit script, $EdOp \cup \{\#\}$, must be defined. In a similar fashion to the non-stochastic case, let this be defined by a table C^Θ of dimensions $|\Sigma|+1 \times |\Sigma|+1$ such that:

$$\begin{array}{ll} \text{for } \langle x,y \rangle \in \Sigma \times \Sigma & p(\langle x,y \rangle) = C^\Theta(x,y) \\ \text{for } x \in \Sigma & p(\langle x,\lambda \rangle) = C^\Theta(\gamma(i),0) \\ \text{for } y \in \Sigma & p(\langle \lambda,y \rangle) = C^\Theta(0,\gamma(j)) \\ & p(\#) = C^\Theta(0,0) \end{array}$$

For convenience $C^\Theta(0,0)$ is interpreted as $p(\#)$. The sum over all the entries in this table should be 1. It is clear that an equivalent cost-table C^Δ can be defined, containing the negated logs of the C^Θ entries, and that $\Delta^V_s(S,T)$ can be equivalently defined by an additive scoring of the scripts using the entries in C^Δ. Therefore $\Delta^V_s(S,T)$ coincides with the standard notion of tree-distance[4] if the cost-table is restricted to be the image of a possible probability-table under the negated logarithm mapping. We will call such tables stochastically valid cost tables. Again it is easy to see that with sequences encoded as vertical trees, these notions coincide with those defined on sequences by Ristad and Yianilos.

[3] It is worth noting that for the equivalence between mapping-based costs and script-based costs, the scripts which correspond to mappings mention each source and target symbol exactly once. Thus the 'short' script segments shown in the picture are not representative of the scripts which correspond to a mapping

[4] $\Delta^V_s(S,T)$ will include a contribution from the negated log of $p(\#)$. As all pairs will share this contribution, any application ranking pairs can ignore this contribution.

For the *Viterb-script* distance $\Delta^V_s(S,T)$, the well known Zhang/Shasha algorithm is an implementation. The Viterbi-script similarity Θ^V_s can also be obtained by a variant replacing $+$ with $\times$. Implementing the *All-script* distance $\Delta^A_s(S,T)$ (or equivalent similarity $\Theta^A_s(S,T)$) turns out though to require one subtle change to the original Zhang/Shasha formulation. This is explained at further length below.

Figure 1 gives an algorithm for Δ^A_s and Δ^V_s. To discuss it first some definitions from (Zhang and Shasha, 1989) are required. The algorithm operates on the left-to-right post-order traversals of trees. If k is the index of a node of the tree, the *left-most leaf*, $l(k)$, is the index of the leaf reached by following the left-branch down. For a given leaf there is a highest node of which it is the left-most leaf and any such node is called a *key-root*. For any tree S, $KR(S)$ is the key-roots ordered by position in the post-order traversal. If i is the index of a node of S, $S[i]$ is the sub-tree of S rooted at i (i.e. all nodes n such that $l(i) \leq n \leq i$). Where i is any node of a tree S, for any i_s with $l(i) \leq i_s \leq i$, the prefix of $S[i]$ from $l(i)$ to i_s can be seen as a *forest* of subtrees of $S[i]$, denoted $For(l(i), i_s)$.

The description instantiates to two algorithms, with $x = V$ for Viterbi, and $x = A$ for All-Scripts. In both cases, it is a doubly nested loop ascending through the key-roots of S and T, in which for each pair of key-roots (i, j), a sub-routine $tree\_dist^x(i, j)$ is called. Values in a *tree*-table $\mathcal{T}$ are set during calls to $tree\_dist^x(i, j)$ and persist. Each call to $tree\_dist^x(i, j)$ operates on a sub-region[5] of the *forest*-table $\mathcal{F}$, from $l(i)-1, l(j)-1$ to i, j. The loop is designed so that $\mathcal{F}[i_s][j_t]$ is the *forest*-distance from $For(l(i), i_s)$ to $For(l(j), j_t)$. $\mathcal{F}$-entries do not persist between separate calls to $tree\_dist^x(i, j)$.

In the Viterbi case, TD^V, there is no inversion from neg-logs to probabilities, and the algorithm can be applied when C^Δ is an arbitrary table of atomic costs.

It is the design of case 2 that enforces that only Tai mappings are considered: when a forest distance $\mathcal{F}^V[i_s][j_t]$ is to be computed, the possibility that i_s is mapped to j_t is factored into a forest+tree combination $TM^V = \mathcal{F}^V[l(i_s)-1][l(j_t)-1] + \mathcal{T}^V[i_s][j_t]$, so that descendants of i_s can only possibly match with descendants of j_t and vice-versa.

```
input:$traversals\ S\ and\ T\ of\ two\ trees$
      $a\ cost\ table\ C^\Delta$

$compute\ KR(S),\ KR(T)$
$create\ table\ \ \mathcal{T}^x,\ size\ |\ S\ |\ \times\ |\ T\ |$
$create\ table\ \ \mathcal{F}^x,\ size\ |\ S\ |\ +\ 1\ \times\ |\ T\ |\ +\ 1$

$TD^x(S,T)$ {
 $for\ each\ i\ \in\ KR(S)\ in\ ascending\ order$ {
    $for\ each\ j\ \in\ KR(T)\ in\ ascending\ order$ {
      $execute\ tree\_dist^x(i,j)$}
   }
 return  $\mathcal{F}^x[|S|][|T|]$
}

$tree\_dist^x(i,j)$ {
 where $i_0 = l(i)-1, j_0 = l(j)-1$
  $\mathcal{F}^x[i_0][j_0] = 0$                          $initialize$
  for $i_s = l(i)$ to $i_s = i$ {
   $\mathcal{F}^x[i_s][j_0] = \mathcal{F}^x[i_s-1][j_0] + C^\Delta(\gamma(i_s),0)$ }
  for $j_t = l(j)$ to $j_t = j$ {
   $\mathcal{F}^x[i_0][j_t] = \mathcal{F}^x[i_0][j_t-1] + C^\Delta(0,\gamma(j_t))$ }

 for $i_s = l(i)$ to $i_s = i$
 for $j_t = l(j)$ to $j_t = j$                         $loop$
  $M^x = \mathcal{F}^x[i_s-1][j_t-1] + C^\Delta(\gamma(i_s),\gamma(j_t))$
  $D^x = \mathcal{F}^x[i_s-1][j_t] + C^\Delta(\gamma(i_s),0)$
  $I^x = \mathcal{F}^x[i_s][j_t-1] + C^\Delta(0,\gamma(j_t))$
  $TM^x = \mathcal{F}^x[l(i_s)-1][l(j_t)-1] + \mathcal{T}^x[i_s][j_t]$
   1:if($l(i_s) == l(i)$ and $l(j_t) == l(j)$) {
    $\mathcal{F}^x[i_s][j_t] = OP^x(M^x, D^x, I^x)$
    $\mathcal{T}^x[i_s][j_t] = M^x\ \ (*)$}
   2:if($l(i_s) \neq l(i)$ or $l(j_t) \neq l(j)$)
    {$\mathcal{F}^x[i_s][j_t] = OP^x(D^x, I^x, TM^x)$}
```

Figure 1: Viterbi and All-paths tree-distance algorithms. Set x to V throughout for Viterbi, with $OP^V = min$, and x to A for All-paths, with $OP^A = logsum$, where $logsum(x_1 \ldots x_n) = -log(\sum_i(2^{-x_i}))$.

Setting x to V for Viterbi, the algorithm is almost identical to that in (Zhang and Shasha, 1989), except for the asterixed line, which in the original would be:

$$\mathcal{T}^V[i_s][j_t] = \mathcal{F}^V[i_s][j_t](**)$$

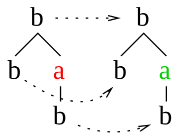
Whereas the original (**) formula for updating the tree table in case 1 updates it to store the true tree-distance between $S[i_s]$ and $T[j_t]$, the (*) variant stores just M^V, the cost of the least-cost script for an alignment of $For(l(i_s), i_s)$ to $For(l(j_t), j_t)$ *in which nodes* i_s *and* j_t *are mapped to each other*. For the Viterbi cost, (*) and (**) could be interchanged and so have $\mathcal{T}^V[i_s][j_t]$ store a cost in which i_s and j_t might not be mapped to each other: in such a case when the values in $\mathcal{T}^V$ are called on in case 2, the TM^V component will just be equal to one or other of the I^V or D^V components over which the minimum is calculated.

Reading the algorithm now with x set to A, $\mathcal{T}^A$ and $\mathcal{F}^A$ represent the 'all-scripts' probabilities, sums over all scripts which serialize a Tai mapping between the relevant trees or forests. Looking again at the aster-

[5] The initialisation sets the left-most column of this to represent the pure deletion cases $For(l(i), i_s)$ to $\emptyset$, and the uppermost row to represent the pure insertion cases $\emptyset$ to $For(l(j), j_t)$

ixed line, through not setting $\mathcal{T}^A[i_s][j_t] = \mathcal{F}^A[i_s][j_t]$, $\mathcal{T}^A[i_s][j_t]$ is *not* the log of the sum of the probabilities of *all* the scripts which can align $S[i_s]$ and $T[j_t]$ but instead the sum over all the cases in which i_s *is mapped to* j_t. For the subsequent use of $\mathcal{T}^A$ in case 2, this is now a necessary feature: if $\mathcal{T}^A[i_s][j_t]$ does not have this interpretation then when these values are called upon in case 2, probabilities of scripts ending in either deletion of i_s or insertion of j_t are doubly counted.

Example. Let $t_1 = t_2 = (b\ (b)\ (a\ b))$, and suppose the cost-table to the right below, which represents as negated logs the assumptions that all probabilites are 0 except for $p(a,\lambda) = 1/8 = p(\lambda,a)$, $p(b,b) = 1/4$, and $p(\#) = 1/2$. The left is the only Tai-mapping which is associated with a non-zero probability edit-script in this setting



	λ	a	b
λ	1	3	inf
a	3	inf	inf
b	inf	inf	2

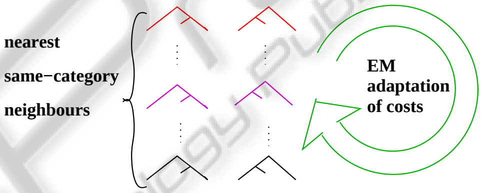
By inspection, $\Theta^A_s(t_1,t_2) = (1/2)^6(1/64)$, $\Theta^V_s(t_1,t_2) = (1/2)^6(1/128)$, and these are the values, or rather their negated logs, which will be calculated[6] by the algorithm in Figure 1. However, if $\mathcal{T}^A[a_3][a_3]$ were to include the probabilities for scripts involving the deletions or insertions of a, $\Theta^A_s(t_1,t_2)$ would be incorrectly calculated to be $(1/2)^6(3/64)$.

As a final remark concerning the algorithm for the Viterbi case, it is straightforward to extend the algorithm so that it returns not just the cost of the best script but also the best script itself. Hence we shall write $(v,V) = TD^V(S,T)$.

4 EM FOR COST ADAPTATION

As noted in section 1, a possible use of a distance measure is for deployment in a k-NN classifier, determining a category for a test item based on its distances to examples in a pre-categorised example set. This is the case in the experiments reported on in section 5. In those experiments the categorised items are the syntax-structures of natural language questions, and the categories are broad semantic categories, such as HUM ('the question expects a human being to be identified as the answer') or LOC ('the question expects a location to be identified as the answer').

For the tree-distance measures, the performance of the classifier is going to vary with the atomic parameter settings in the cost-table C^Δ. One might expect that scripts between pairs of trees (or strings) that belong to the *same category* differ from scripts between pairs of trees (or strings) that belong to *different categories*. For example, for the question-categorisation scenario, on same-category pairs one might expect that the substitution $(who/when)$ to be less frequent that the substitution $state/country$. In terms of the parameters of the stochastic distances this would correspond to $P(who,when) << P(state,country)$, or equivalently in terms of negated logs, $C^\Delta(who,when) >> C^\Delta(state,country)$. This leads to the idea that one might be able to use Expectation-Maximisation techniques (Dempster et al., 1977) to adapt edit-probs from a corpus of *same-category nearest neighbours*.



Such a technique, for the case of stochastic string distance, was first proposed by (Ristad and Yianilos, 1998).

4.1 All-scripts EM

As a first step towards a cost-adaptation algorithm, consider the following *brute-force all-scripts EM algorithm*, EM^A_{bf}, consisting in interations of the following pair of steps

(Exp)$_A$ *generate a virtual corpus of scripts by treating each training pair* (S,T) *as standing for* all *the edit-scripts* σ*, which can relate* S *to* T*, weighting each by its conditional probability* $P(\sigma/\Theta^A_s(S,T)$*, under current probalities* C^Θ

(Max) *apply maximum likelihood estimation to the virtual corpus to derive a new probability table.*

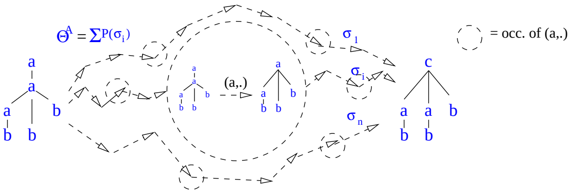
A virtual count or expectation $\gamma_{S,T}(op)$ contributed by S,T for an operaton op can be defined by

$$\gamma_{S,T}(op) = \sum_{\sigma:S\mapsto T}\left[\frac{P(\sigma)}{\Theta^A_s(S,T)} \times freq(op \in \sigma)\right]$$

and the **(Exp)**$_A$ accumulating these values $\gamma_{S,T}(op)$ for all possible op's over all training pairs. The picture below attempts to illustrate this for a particular

[6] The reason for the premultiplying $(1/2)^6$ factor in these numbers is that it is easier in this case to calculate first ignoring $p(\#)$ and from a table in which all entries are twice as large, and then to correct for the over-estimation; the only scripts making any contribution all have length 5

operation (a,λ) occuring in various scripts between a particular tree pair



For the case of *linear trees*, this amounts to the same adaptation proposal as that put forward by (Ristad and Yianilos, 1998). This brute-force algorithm is exponentially expensive. To obtain a feasible equivalent algorithm one may attempt to apply the same strategy as that used by (Ristad and Yianilos, 1998) for the case of linear trees, which is for each tree pair (S,T), to first compute *position-dependent* expectations $\gamma_{(S,T)}[i][j](op)$ for each operation and then sum these position-independent expectations to give the expecations per-pair $\gamma_{(S,T)}(op)$. In this approach, $\gamma_{(S,T)}[i][j](m,m')$, the expectation for a swap (m,m') at (i,j) has the semantics

$$\begin{aligned}\gamma_{(S,T)}[i,j](m,m') &= \Sigma_{\sigma\in E(S,T),(m_i,m'_j)\in\sigma}[\frac{p(\sigma)}{\Theta^A_S(S,T)}] \\ &= \frac{1}{\Theta^A_S(S,T)} \times \Sigma_{\sigma\in E(S,T),(m_i,m'_j)\in\sigma}[p(\sigma)]\end{aligned}$$

or in words, it is *the sum over the conditional probabilities of any script σ containing a m_i,m'_j substitution, given that it is a script between S and T.*

For the case of *linear trees*, the position-dependent expectations $\gamma_{(S,T)}[i][j]$ can be computed feasibly because firstly, the summation in the above can be factorised into a product of 3 terms

$$\begin{aligned}&\Sigma_{\sigma\in E(S,T),(m_i,m'_j)\in\sigma}[p(\sigma)] \qquad (1)\\ &= \Sigma_{\sigma_{pre}\in E(S_{1:1-1},T_{1:j-1})}[p(\sigma_{pre})]\times \\ &\quad p(m,m')\times \\ &\quad \Sigma_{\sigma_{suff}\in E(S_{i+1:I},T_{j+1:J})}[p(\sigma_{suff})]\end{aligned}$$

and secondly the summations over the possible scripts prefixing (m_i,m'_j), and the possible scripts suffixing (m_i,m'_j) can be straightforwardly calculated; the first is the all-scripts algorithm, and the second an easily formulated 'backwards' variant.

For the case of general trees (as opposed to linear trees) (Boyer et al., 2007) propose such a factorisation approach. Their proposal turns out, however, to be unsound, factorizing the problem in a way which is invalid given the ancestry-preservation aspect of Tai mappings[7]. To explain this, consider Figure 2, which reproduces the essentials of an example from their paper[8].

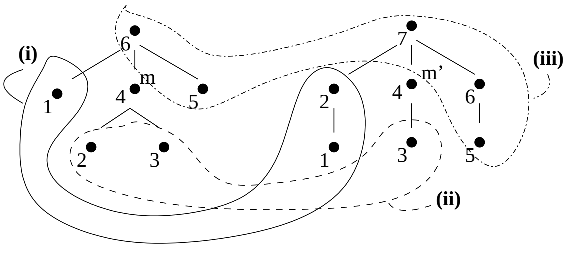
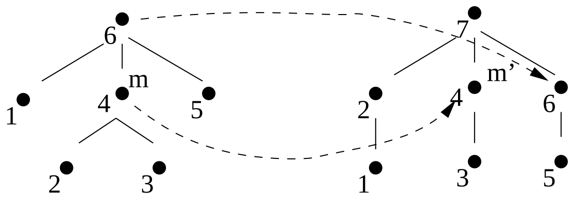


Figure 2: The swap-case in expectation calculation.

For the pair of trees, we wish to calculate the position-dependent expectation $\gamma[4,4](m,m')$. (Boyer et al., 2007) propose an algorithm implying the correctness of the factorisation

$$\begin{aligned}&\Sigma_{\sigma\in E(S,T),(m_4,m'_4)\in\sigma}[p(\sigma)] && (2)\\ &= \Sigma_{\sigma_1\in E([(\cdot_1)],(\cdot_2(\cdot_1)))}[p(\sigma_1)]\times && [(i)]\\ &\quad \Sigma_{\sigma_2\in E([(\cdot_2)(\cdot_3)],[(\cdot_3)])}[p(\sigma_2)]\times p(m,m') && [(ii)]\\ &\quad \Sigma_{\sigma_3\in E([(\cdot_6(\cdot_5))],[(\cdot_7(\cdot_6(\cdot_5)))])}[p(\sigma_3)] && [(iii)]\end{aligned}$$

where the terms (i)–(iii) corresponds to the indicated regions in Figure 2. The problem is with final term (iii) in the product. Each edit-script $\sigma\in E(S,T)$ represents a Tai mapping between S and T. The summation $\Sigma_{\sigma\in E(S,T),(m_4,m'_4)\in\sigma}[p(\sigma)]$ refers to those scripts which represent a Tai mapping with the property that m_4 is mapped to m'_4. This means that if an ancestor of m_4 is in the mapping (ie. not deleted) then its image under the mapping must be an ancestor of m'_4, and vice-versa. The final term in the product, $\Sigma_{\sigma_3\in E([\cdot_6(\cdot_5)],[\cdot_7(\cdot_6(\cdot_5))])}[p(\sigma_3)]$, sums over *any* script between these two sub-trees of S and T and this will include scripts in which node $\cdot_6$ of S is mapped to $\cdot_6$ of T, and this corresponds to a mapping in which an ancestor of m_4 is mapped to a non-ancestor of m'_4:



For example if the only non-zero probability script from $(\cdot_6(\cdot_5))$ to $(\cdot_7(\cdot_6(\cdot_5)))$ is one mapping the $\cdot_6$ of S to the $\cdot_6$ of T then $\gamma[4,4](m,m')$ should be zero, though according to (2) it will not be.

For general trees, a feasible equivalent to the brute-force EM^{bf}_A remains an unsolved problem.

4.2 Viterbi EM

An approximation to the the All-scripts proposal consists in simply in replacing the $\mathbf{Exp}_A$ step by

[7] A fact which they concede p.c.

[8] Fig. 3 p61

(Exp)$_V$ *generate a virtual corpus of scripts by treating each training pair (S,T) as standing for the* best *edit-script* σ, *which can relate S to T, weighting it by its conditional probability $P(\sigma)/\Theta^A_s(S,T)$, under current costs $\mathcal{C}$*

Where $\mathcal{V}$ is the best-script, the virtual count or expectation $\gamma_{S,T}(op)$ contributed by S,T for the operaton op would in this case be defined by

$$\gamma_{(S,T)}(op) = \frac{\Theta^V_s(S,T)}{\Theta^A_s(S,T)} \times freq(op \in \mathcal{V})$$

and the **(Exp)**$_V$ step accumulates these values $\gamma_{(S,T)}(op)$ for all possible op's over all training pairs. The picture below attempts to illustrate this for a particular operation (a,λ) occuring on the best-path $\mathcal{V}$ between a particular tree pair

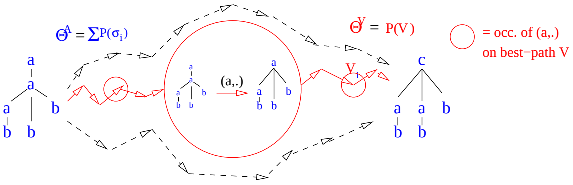


Figure 3 spells out this Viterbi cost-adaptation algorithm for stochastic tree-distance.[9]

```
input: a set P of tree pairs (S,T)
       a cost table C, size |Σ|+1 × |Σ|+1

create tables γ,C_new same size as C

while(conv ≠ true) {
 zero all entries in γ
 for each (S,T) ∈ P {
   let (v, 𝒱) = TD^V(S,T), a = TD^A(S,T)
   γ[λ][λ]+=2^{-v}/2^{-a}
   for each (x,y) ∈ EdOp {
    γ[x][y]+=(freq of (x,y) in 𝒱)×2^{-v}/2^{-a}
   }
 }
 C_new = −log(γ/sum(γ))
 if (C_new ≠ C){C = C_new} else {conv = true}
}
return C
```

Figure 3: Viterbi EM cost adaptation for tree-distance. Note Σ is the label alphabet of the tree-pairs in $\mathcal{P}$. The algorithms TD^V and TD^A are as defined in Figure 1.

Such Viterbi training variants have been found beneficial, for example in the context of parameter training for PCFGs (Benedí and Sánchez, 2005).

5 EXPERIMENTS WITH VITERBI EM COST-ADAPTATION

We have conducted some experiments with this Viterbi EM cost-adaptation approach. In particular we have considered how it might adapt a tree-distance measure that is put to work in a k-NN classification algorithm.

Figure 4 outline the distance-weighted kNN classification algorithm which was used in the experiments.

```
knn_class(Examples,C^Δ,k;T) {
 let 𝒟 = SORT({(S,Δ_s^V(S,T)) | S ∈ Examples }
 while(!resolved) {
   𝒫 = top(k, 𝒟),  𝒱 = weighting( 𝒫)
   if(no winner in 𝒱) { set k = k + 1 }
   else { resolved = true }
 }
return category with highest vote in 𝒱
}
```

Figure 4: Distance-weighted k nearest neighbour classification.

$top(k,\mathcal{D})$ basically picks the first k items from $\mathcal{D}$[10]. The *weighting* converts the panel of distance-rated items to weighted votes for their categories, and in the experiments reported later, the options for the conversion of an item of category C, at distance d, into a vote $vote(C,d)$ are *Majority*: $vote(C,d) = 1$; *Dudani*: $vote(C,d) = (d_{max} - d)/(d_{max} - d_{min})$, or 1 if $d_{max} = d_{min}$, where d_{max} and d_{min} are maximum and minimum distances in the panel (Dudani, 1976).

It can arise that the test tree T contains a symbol for which C^Δ has no entry. One option is to assign all operations involving the symbol some default cost κ. See the Appendix for a proof that the ordering of neigbours is independent of the value chosen for κ

In applying the EM^V cost-adaptation in the context of the k-NN classification algorithm, the training set for cost-adaptation was taken to consists of tree pairs (S,T), where for each example-set tree S, T is a nearest same-category neighbour. The training algorithm should less the stochastic tree-distance between these trees.

EM^V like all other EM algorithms needs an initialisation of its parameters. We will use $C^\Delta{}_u(d)$ for a 'uniform' initialisation with diagonal factor d. This will mean that $C^\Delta{}_u(d)$ is a stochastically valid cost-table, with the additional properties that (i) all diagonal entries are equal (ii) all non-diagonal entries are equal (iii) diagonal entries are d times more probable

[9] Simple modifications of the algorithm as formulated force it to generate a symmetric expectation table γ.

[10] Modulo some niceties concerning ties which space precludes detailing

than non-diagonal. For these purposes the cost-table entry for $p(\#)$ is treated as non-diagonal. As an illustration, for an alphabet of just 2 symbols, the initialisations $C^{\Delta}{}_u(d)$ for d = 3, 10, 100, and 1000 are:

3	λ	*a*	*b*
λ	3.7	3.7	3.7
a	3.7	2.115	3.7
b	3.7	3.7	2.115

10	λ	*a*	*b*
λ	4.755	4.755	4.755
a	4.755	1.433	4.755
b	4.755	4.755	1.433

100	λ	*a*	*b*
λ	7.693	7.693	7.693
a	7.693	1.05	7.693
b	7.693	7.693	1.05

1000	λ	*a*	*b*
λ	10.97	10.97	10.97
a	10.97	1.005	10.97
b	10.97	10.97	1.005

As a *smoothing* option concerning a table C^{Δ} derived by EM^V, let $C^{\Delta}{}_{\lambda}$ be its interpolation with the original $C^{\Delta}{}_u(d)$ as follows

$$2^{-C^{\Delta}{}_{\lambda}[x][y]} = \lambda(2^{-C^{\Delta}[x][y]}) + (1-\lambda)(2^{-C^{\Delta}{}_u(d)[x][y]})$$

with $0 \leq \lambda \leq 1$, with $\lambda = 1$ giving all the weight to the derived table, and $\lambda = 0$ giving all the weight to the initial table.

The dataset used was a natural language processing one, being a corpus of (broadly) semantically categorised, and syntactically analysed questions, which was created by from two pre-existing datasets. QuestionBank (QB) is a hand-corrected treebank for questions (Judge et al., 2006; Judge, 2006b), (Judge, 2006a). A substantical percentage of the questions in QB are taken from a corpus of semantically categorised, syntactically unannotated questions (CCG, 2001). From these two corpora we created a corpus of 2755 semantically categorised, syntactically analysed questions, spread over the semantic categories as follows[11]

Cat	HUM	ENTY	DESC	NUM	LOC	ABBR
N	647	621	533	461	455	38
%	23.48	22.54	19.35	16.73	16.52	1.38

For further details of the software and data see (Emms, 2011). Figure 5 shows some results of a first set of experiments, with unit-costs and then with some stochastic variants. For the stochastic variants, the cost initialisation was $C^{\Delta}{}_u(3)$ in each case. All the experiments followed a stratified 10-fold cross-validation approach. The data was randomly split into 10 equal size folds, with approximately equal distribution of the categories in each. Then in turn each fold has taken as the test data, and the remaining 9 folds used as the example set. When cost-adaptation was applied this means that the training pairs for EM^V come from the example set. The figure shows results using the *Dudani-voting* variant of k-NN; the *Majority-voting* variant was less effective.

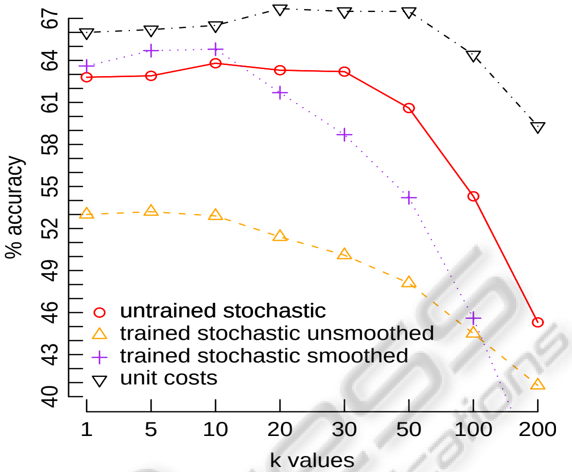


Figure 5: Categorisation performance with unit costs and some stochastic variants.

The first thing to note is that performance with unit-costs (▽, max. 67.7%) exceeds performance with the non-adapted $C^{\Delta}{}_u(3)$ costs (○, max. 63.8%). Though not shown, this remains the case with far higher settings of the diagonal factor. Performance after applying EM^V to adapt costs (△, max. 53.2%) is worse than the initial performance (○, max. 63.8%). A Leave-One-Out evaluation, in which *example-set* items are categorised using the method on the remainder of the example-set, gives accuracies of 91% to 99%, indicating EM^V has made the best-scripts connecting the training pairs *too* probable, *over-fitting* the cost table. The vocabulary is sufficiently thinly spread over the training pairs that its quite easy for the learning algorithm to fix costs which make almost everything but exactly the training pairs have zero probability. The performance when smoothing is applied (+,max. 64.8%), interpolating the adapted costs with the initial cost, with $\lambda = 0.99$, is considerably higher than without smoothing (△), attains a slightly higher maximum than with unadapted costs (○), but is still worse than with unit costs (▽).

The following is a selection from the top 1% of adapted swap costs.

8.50	?	.	12.31	The	the
8.93	NNP	NN	12.65	you	I
9.47	VBD	VBZ	13.60	can	do
9.51	NNS	NN	13.83	many	much
9.78	a	the	13.92	city	state
11.03	was	is	13.93	city	country
11.03	’s	is			

For the data-set used, these learned preferences are to some extent intuitive, exchanging punctuation marks,

[11] See (CCG, 2001) for details of the semantic category labels

words differing only by capitalisation, related parts of speech (VBD vs VVZ etc), verbs and their contractions and so on. One might expect this discounting of these swaps relative to others to assist the categorisation, though the results reported so far indicate that it did not.

Recall that the cost-table for the stochastic edit distance is a representation of probabilites, with probabilities represented by their negated base-2 logarithms. A 0 in this case represents the probability 1. Because in a stochastically valid cost table, the sum over all the represented probabilities must be 1, a single 0 entry in a cost table implies *infinite* cost entries everywhere else. This means that a stochastically valid cost table cannot have zero costs on the diagonal, which is the situation of the unit-cost table, c_{01}. This aspect perhaps mitigates against success. The diagonal factor d in the cost initialisation is designed to make the entries on the diagonal more probable than other entries, but even with very high values for d, indicating a high ratio between the diagonal and off-diagonal probabilities, the diagonal costs are not negligible. This means that the unit-cost setting, c_{01}, which is clearly 'uniform' in a sense, is not directly emulated by the 'uniform' stochastic initialisations $C^{\Delta}{}_u(d)$. The performance with the *unadapted* uniform stochastic initialisation was below the performance with unit-costs. Although results in Figure 5 show just the outcomes with $c_u(3)$, this remained the case with far larger values of the diagonal factor d. This invites consideration of outcomes if a final step is applied in which *all the entries on the cost-table's diagonal are zeroed*. In work on adapting cost-tables for a stochastic version of *string distance* used in duplicate detection, (Bilenko and Mooney, 2003) used essentially this same approach. Figure 6 shows outcomes when the trained and smoothed costs finally have the diagonal zeroed.
The ($\bigtriangledown$) series once again shows the outcomes with unit-costs. In this experiment, with the diagonal zeroed, this is necessarily also the outcome obtained with any unadapted uniform stochastic initialisation $C^{\Delta}{}_u(d)$. The other lines in the plot show the outcomes obtained with costs adapted by EM^V, smoothed at varius levels of interpolation ($\lambda \in \{0.99, 0.9, 0.5, 0.1\}$) and with the diagonal zeroed. Now the unit costs base-line is clearly out-performed, the best result being 72.5% ($k = 20$, $\lambda = 0.99$), as compared to 67.5% for unit-costs ($k = 20$). Also better results are obtained with the higher levels of the interpolation factor, indicating greater weight given to the values obtained by EM^V and less to the stochastic initialisation.

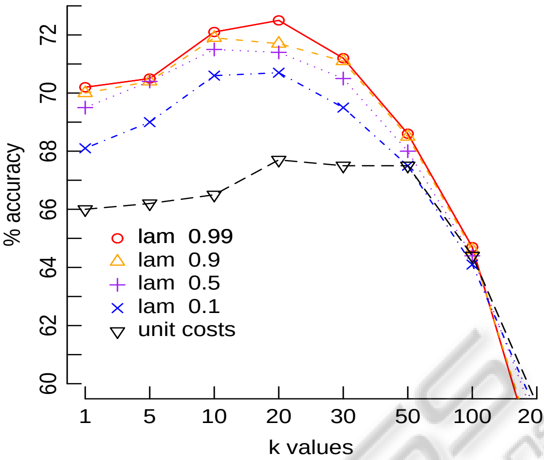


Figure 6: Categorisation performance: adapted costs with smoothing and zeroing.

6 CONCLUSIONS

One can tentatively conclude on the basis of these experiments that the Viterbi EM cost-adaptation can increase the performance of a tree-distance based classifier, and improve it to above that attained in the unit-cost setting, though this does require smoothing of derived probabilites and a final step of zeroing the diagonal.

Experiments with further data-sets is required. One type of data of interest would be the digit-recognition data-set represented by a tree-encoding of outline looked at by (Bernard et al., 2008). It would also be interest to look at applications not to do with categorisation per-se. For example, in the NLP-related tasks of question-answering and entailment recognition, the aim is assess pairs of sentences for their likelihood to be a question-answer or hypothesis-conclusion pair. A training set of such pairs could also serve as potential input to the cost adaptation algorithm.

Alignments between different pairs of trees can end up being represented by the same edit-script. A minimal example is that the script $(a,A)(b,B)(c,C)$ can serve to connect both the pairs $(t1,t2)$ and $(t1',t2')$, where:

$$\begin{array}{ll} t1 : (c(a)(b)) & t2 : (C(A)(B)) \\ t1' : (c(b(a))) & t2' : (C(B(A))) \end{array}$$

Therefore the All-script and Viterbi-script stochastic edit-distances are only a step towards a fully fledged generative model of aligned trees. A fully-fledged model would include further factors to divide the probability $P(a,A) \times P(b,B) \times P(c,C)$

between the tree-pairs. A direction for further work is the investigation of such a model of aligned trees, and how it relates to some other recent proposals concerning adaptive tree measures such as (Takasu et al., 2007), (Dalvi et al., 2009)

ACKNOWLEDGEMENTS

This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Trinity College Dublin.

REFERENCES

Benedí, J.-M. and Sánchez, J.-A. (2005). Estimation of stochastic context-free grammars and their use as language models. *Computer Speech and Language*, 19(3):249–274.

Bernard, M., Boyer, L., Habrard, A., and Sebban, M. (2008). Learning probabilistic models of tree edit distance. *Pattern Recogn.*, 41(8):2611–2629.

Bilenko, M. and Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pages 39–48.

Boyer, L., Habrard, A., and Sebban, M. (2007). Learning metrics between tree structured data: Application to image recognition. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, pages 54–66.

CCG (2001). corpus of classified questions by Cognitive Computation Group, University of Illinois l2r.cs.uiuc.edu/∼cogcomp/Data/QA/QC.

Dalvi, N., Bohannon, P., and Sha, F. (2009). Robust web extraction: an approach based on a probabilistic treeedit model. In *SIGMOD 09: Proceedings of the 35th SIGMOD international conference on Management of data*, pages 335–348, New York, NY, USA. ACM.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statistical Society*, B 39:138.

Dudani, S. (1976). The distance-weighted k-nearest neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6:325–327.

Emms, M. (2011). Tree-distance code and datasets reported on in experiments www.scss.tcd.ie/Martin.Emms/TreeDist.

Emms, M. and Franco-Penya, H.-H. (2011). On order equivalences between distance and similarity measures on sequences and trees. In *Proceedings of ICPRAM 2012 International Conference on Pattern Recognition Application and Methods*.

Judge, J. (2006a). Corpus of syntactically annotated questions http://www.computing.dcu.ie/jjudge/qtreebank/.

Judge, J. (2006b). *Adapting and Developing Linguistic Resources for Question Answering*. PhD thesis, Dublin City University.

Judge, J., Cahill, A., and van Genabith, J. (2006). Questionbank: creating a corpus of parse-annotated questions. In *ACL 06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 497–504, Morristown, NJ, USA. Association for Computational Linguistics.

Kuboyama, T. (2007). *Matching and Learning in Trees*. PhD thesis, Graduate School of Engineering, University of Tokyo.

Ristad, E. S. and Yianilos, P. N. (1998). Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(5):522–532.

Tai, K.-C. (1979). The tree-to-tree correction problem. *Journal of the ACM (JACM)*, 26(3):433.

Takasu, A., Fukagawa, D., and Akutsu, T. (2007). Statistical learning algorithm for tree similarity. In *ICDM 07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 667672, Washington, DC, USA. IEEE Computer Society.

Wagner, R. A. and Fischer, M. J. (1974). The string-tostring correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173.

Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18:1245–1262.

APPENDIX

Proof Concerning Out of Table Costs

Let C^{Δ} be a cost-table associated with a given label alphabet Σ, let T be a tree with n symbols $\notin \Sigma$, and let κ be a fixed, out-of-table cost for any (x,u), where $x \in \Sigma \cup \{\lambda\}$, $u \notin \Sigma$. Suppose S is a tree whose labels are in Σ. Every $\sigma \in E(S,T)$ involve n out-of-table events. Suppose $\mathcal{V}$ is the least-cost script, with cost $cost_{\kappa}(\mathcal{V})$. Now suppose under a higher setting of κ' for out-of-table costs, that $\mathcal{V}' \neq \mathcal{V}$ is the least-cost script, so $cost_{\kappa'}(\mathcal{V}') < cost_{\kappa'}(\mathcal{V})$. But recosting according to κ gives $cost_{\kappa}(\mathcal{V}') < cost_{\kappa}(\mathcal{V})$, which contradicts minimality of $\mathcal{V}$ under κ. So the minimal script is invariant to changes of κ, and $\Delta^{\nu}_{\kappa'}(S,T) - \Delta^{\nu}_{\kappa}(S,T) = n \times (\kappa - \kappa')$. It follows that neighbour ordering is invariant to changes of κ.

□