Channel modeling and Levenshtein distances with context-dependent weights

Günther J. Wirsching
Mathematisch-Geographische Fakultät
Katholische Universität Eichstätt-Ingolstadt

December 6, 2010

Abstract

A connection between channel modeling in mathematical information theory and a certain extension of Levenshtein distances is established. The model assigns positive real weights to elementary editing operations *substitution*, *deletion*, *insertion*, and *ending*, which may depend on arbitrary finite contexts. Given a context structure, an algorithm for estimating context-dependent probabilities of elementary editing operations from a given finite training corpus is designed. This algorithm is similar to the EM algorithm, the maximization step being replaced by an estimation step to determine probability structures from weighted counts. Some conditions on the context structure are formulated which make this estimation problem algorithmically accessible by reducing it to the general problem of estimating a probability vector on a finite state space from counts.

1 Introduction

In mathematical information theory [4], a *channel* is defined as a triple $(\mathcal{A}, \nu_x, \mathcal{B})$ consisting of an *input alphabet* $\mathcal{A}$, an *output alphabet* $\mathcal{B}$, and a family of probability measures (ν_x). For each input sequence

$$x = (\dots, x_{-1}, x_0, x_1, x_2, \dots) \in \mathcal{A}^{\mathbb{Z}},$$

taken to be infinite in both directions, ν_x is a probability measure on the space $\mathcal{B}^{\mathbb{Z}}$ of output sequences, where, for each measurable set $Z \subset \mathcal{B}^{\mathbb{Z}}$, $\nu_x(Z)$ is the probability that, given the input sequence x, the output is an element of Z. *Channel modeling* refers to the construction of a stochastic process imitating the channel in the sense that the probability measures ν_x can be computed from the process parameters. In his seminal paper [11], Shannon started information theory by proving his fundamental theorems for channels which are modeled by Markov matrices. The aim of this paper is *channel modeling from data* (for a similar approach see [12] and [9]): Given a *training corpus* of input-output pairs, how can one extract good estimates for the probability measures ν_x? Clearly, the family (ν_x) of probability measures is infinite, and usually the training corpus is a finite set of input-output pairs, where each pair consists of a finite $\mathcal{A}$-word as input and a finite $\mathcal{B}$-word as output. There are two major problems attacked in this paper: the deletion-insertion problem and the finiteness problem.

The finiteness problem will be captured in the notion *context structure* to be introduced in section three. The approach to contexts adopted here is rather abstract: a context structure, as defined here, consists of a fixed set of arbitrary algorithms, each of which being designed for classifying a given context situation, taken from a potentially infinite set of possibilities, according to a prescribed set of classes.

To state the deletion-insertion problem, observe that a given input-output pair may be structured in such a way that, in some cases, it appears 'very likely' that the transmission channel just 'deleted' a symbol ($\in \mathcal{A}$) from the input, or, in other cases, just 'inserted' a symbol ($\in \mathcal{B}$) to the output. To come to a mathematical description of this situation, an *editing trellis* is associated to each input-output pair $(s_{\text{in}}, s_{\text{out}})$; if both input and output are finite strings, then the trellis is finite. The trellis contains *arrows* representing the elementary editing operations *substitution*, *deletion*, and *insertion*; a similar trellis is sometimes used to visualize the original Levenshtein algorithm [7] for the computation of distances between strings. By definition, the trellis describes all editing sequences built from the considered editing operations and leading from s_{in} to s_{out}. It turns out that, if an additional elementary editing operation called *ending* is introduced, then the editing trellis described here is an appropriate tool to estimate context-dependent weights from given transmission observations.

Note that an influence of future contexts is not excluded. This is motivated by one special type of channel I have in mind: the channel 'spoken language', where input and output can be considered as sequences of phonemes. Modeling this channel can be applied to speech recognition, where weighted Levenshtein distances without channel modeling background already are in use [10]. Another possible application could be the tracking of sound changes in natural language over centuries, which, according to the neogrammarian hypothesis, should be determined by phonological environment, see, for instance, Malmkjær [8]. In both application, it is intuitively clear that one should not neglect *a priori* possible influence of succeeding phonemes, i. e., in the information theoretic notions adopted here, of *future contexts.* Thus, the channel 'spoken language' is *anticipating* in the sense of information theory.

The content of this paper is organized as follows. After an introductory section on the use of the editing trellis, section three is devoted to an explicit statement and proof of the connection between context-dependent probabilities of elementary editing operations and the probabilities (ν_x) defining the mathematical model of a transmission channel. An algorithm for estimating probabilities of elementary editing operations is presented and explained in detail beginning in section four. Section five addresses the estimation of context-dependent probabilities from counts. The formulation of conditions under which this estimation problem reduces to the general problem of estimating a probability vector on a finite state space from counts, leads to the concept of *normal* context structures.

2 The trellis

Assume that two alphabets are given, a finite non-empty input alphabet $\mathcal{A}$ for the symbols coming in the channel, and a finite non-empty result alphabet $\mathcal{B}$ containing the possible symbols resulting after transmission. In addition, assume existence of two additional symbols which are supposed to be not contained in $\mathcal{A} \cup \mathcal{B}$:

$\langle$ meaning *start message*,
$\rangle$ meaning *end of message.*

Here are some examples to see what can happen when an input string $\langle a\,b\,c \rangle$ is transformed by the channel into an output string.

Input	$\langle \;\; a \;\; b \;\; c \;\; \rangle$	$\langle \;\; a \;\; b \;\; c \;\; \rangle$	$\langle \;\; a \;\; b \;\; c \;\; \rangle$
Output	$\langle \;\; a \;\; x \;\; c \;\; \rangle$	$\langle \;\; a \;\; c \;\; \rangle$	$\langle \;\; a \;\; b \;\; b \;\; c \;\; \rangle$

(1)

In the first case, one would say that the symbol b is misunderstood to x. The second case is the deletion of the second letter b, and the third case is the insertion of a letter b between a and b, or between b and c.

To describe the situation more precisely, imagine a pointer starting at $\langle$ and jumping from symbol to symbol of the input string. At $\langle$, start an output string with $\langle$, and move the pointer to the next input symbol. In practise, it may be difficult to recognize beginning and/or end of a message, but this problem is not considered in this paper.

At each symbol $\in \mathcal{A}$, we have three possible *editing operations*:

Substitution: choose a symbol $x \in \mathcal{B}$ and write it to the output string; move the pointer to the next input symbol.

Deletion: move the pointer to the next input symbol.

Insertion: choose a symbol $x \in \mathcal{B}$ and write it to the output string; don't move the pointer.

In this setting, a correct recognition (or *copying*) of an input symbol is a special substitution and might be called a *self-substitution.* At the string termination symbol $\rangle$, no substitution nor deletion makes sense. But it is possible to insert additional symbols before terminating the output string.

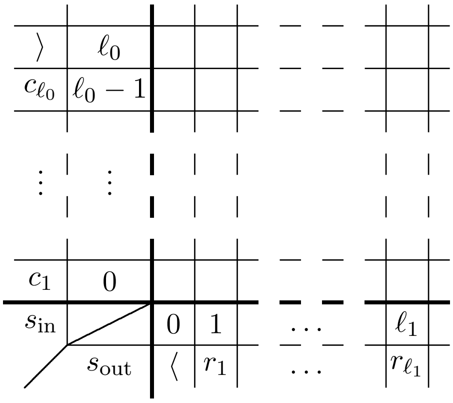
Our next aim is to capture all reasonable editing sequences leading from a given input string of length $\ell_0 \in \mathbb{N}_0$, the *coming* symbols,

$$s_{\text{in}} = \langle c_1 \dots c_{\ell_0} \rangle \in \{\langle\} \times \mathcal{A}^* \times \{\rangle\},$$

to a given output string of length $\ell_1 \in \mathbb{N}_0$, the *received* symbols,

$$s_{\text{out}} = \langle r_1 \dots r_{\ell_1} \rangle \in \{\langle\} \times \mathcal{B}^* \times \{\rangle\};$$

here the usual Kleene star operation is employed to denote, e.g., the set $\mathcal{A}^*$ of all finite $\mathcal{A}$-strings including the empty string. The requested editing sequences will be found in an *editing trellis* consisting of *panels* and *arrows* which is visualized in a grid with $(\ell_0 + 1)$ rows and $(\ell_1 + 1)$ columns:



The horizontal labeling

$$h(i) := \begin{cases} \langle & \text{for } i = 0, \\ r_i & \text{for } i \in \{1, \dots, \ell_1\}. \end{cases}$$

means 'the last received symbol', whereas the vertical labeling

$$v(j) := \begin{cases} c_{j+1} & \text{for } j \in \{0, \dots, \ell_0 - 1\}, \\ \rangle & \text{for } j = \ell_0, \end{cases}$$

means 'the next coming symbol'. Thus, a panel with coordinates (i, j) gets the labeling

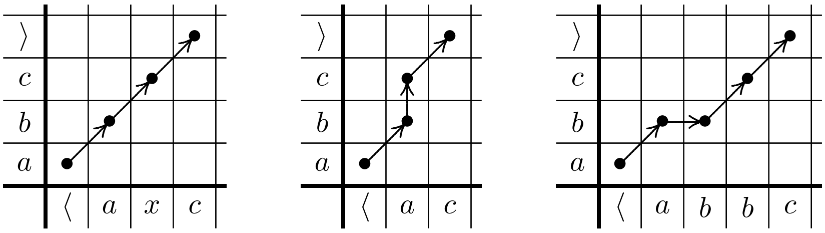
$$p_{ij} := (h(i), v(j)) = (r_i, c_{j+1})$$

describing a situation where the last received symbol is $h(i) = r_i$ and the next coming symbol is $v(j) = c_{j+1}$.

We can find the above mentioned editing operations in the trellis: each editing operation corresponds to an arrow from a panel (i, j) to one of its neighbours.

$$\left.\begin{array}{lll} \textit{Substitution}: & (i,j) \to (i+1, j+1) & \text{(diagonal arrow)} \\ \textit{Deletion}: & (i,j) \to (i, j+1) & \text{(vertical arrow)} \\ \textit{Insertion}: & (i,i) \to (i+1, j) & \text{(horizontal arrow)} \end{array}\right\} \tag{2}$$

The following figure shows possible paths of editing operations leading to the input-output pairs given in (1).



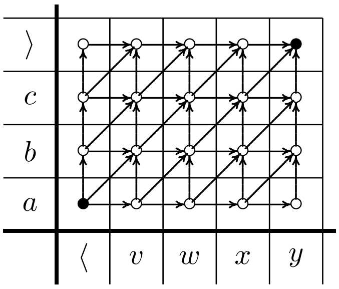
In each trellis, the editing path starts at the panel $(0, 0)$ with labeling

$$p_{00} = (\langle, c_1)$$

and ends at the panel (ℓ_1, ℓ_0) with labeling

$$p_{\ell_1 \ell_0} = (r_{\ell_1}, \rangle).$$

An example for the complete trellis for $s_{\text{in}} = \langle abc\rangle$ and $s_{\text{out}} = \langle vwxy\rangle$ with all possible arrows looks as follows:



(3)

Next we derive a formula for the number, in a general trellis with (ℓ_1+1) and (ℓ_0+1) columns, of possible paths from $(0, 0)$ to (ℓ_1, ℓ_0). One possible path consists of first deleting all symbols from the input string, and subsequently inserting the symbols forming the output string. In this path, we need ℓ_0 deletions and ℓ_1 insertions, making a total of $\ell_0+\ell_1$ editing operations. As a substitution (including copying) can by thought of as a deletion followed by an insertion, any valid editing path from input to output string must obey the formulae

$$n_s + n_d = \ell_0, \qquad n_s + n_i = \ell_1, \tag{4}$$

where n_s denotes the number of substitutions, n_d the number of deletions, and n_i the number of insertions. Given a multi-set M consisting of n_s symbols s, n_d symbols d, and n_i symbols i such that formula (4) is fulfilled, then there is a canonical one-one-correspondance between the set of non-isomorphic orderings of M and the set of possible editing paths in the trellis. Hence the number of editing paths from s_{in} to s_{out} is

$$\begin{aligned} N(s_{\text{in}}, s_{\text{out}}) &= \sum_{n_s=0}^{\min\{\ell_0,\ell_1\}} \binom{\ell_0+\ell_1}{2n_s}\binom{\ell_0+\ell_1-2n_s}{\ell_0-n_s} \\ &= \sum_{n_s=0}^{\min\{\ell_0,\ell_1\}} \frac{(\ell_0+\ell_1)!}{(2n_s)!\,(\ell_0-n_s)!\,(\ell_1-n_s)!}. \end{aligned} \tag{5}$$

Note that it is never algorithmically necessary to examine all these paths separately. Indeed, as we shall see in section 4, a trellis version of the well-known Baum-Welch algorithm extracts essential information about all possible paths in $O(\ell_0\ell_1)$ time.

3 Context structures and probabilities

Suppose that we have an input string $s_{\text{in}} = \langle c_1 \cdots c_{\ell_0}\rangle$ and a corresponding output string $s_{\text{out}} = \langle r_1 \cdots r_{\ell_1}\rangle$. Then consider the trellis T built from s_{in} and s_{out} and associate to each panel (i,j) in T a *context*

$$C_T(i,j) := (R_T(i), L_T(j), F_T(j)) \tag{6}$$

consisting of the following three parts:

$$\left.\begin{array}{lll} R_T(i) \in \{\langle\} \times \mathcal{B}^* : & \langle r_1 \ldots r_i & \text{the } \textit{received} \text{ context,} \\ L_T(j) \in \{\langle\} \times \mathcal{A}^* : & \langle c_1 \ldots c_j & \text{the } \textit{left} \text{ context,} \\ F_T(j) \in \mathcal{A}^* \times \{\rangle\} : & c_{j+1} \ldots c_{\ell_0}\rangle & \text{the } \textit{future} \text{ context.} \end{array}\right\} \tag{7}$$

Moreover, associate to an arrow of type

$$t \in \{\text{substitution, deletion, insertion}\}$$

starting from (i,j) and ending at (i',j') a triple

$$A_T(i,j,t) := (C_T(i,j), t, r) \tag{8}$$

where the components have the following meaning:

$$\left.\begin{array}{rl} C_T(i,j): & \text{the context defined above,} \\[1ex] t \in \{s,d,i,e\}: & \text{the type of the arrow,} \\[1ex] r \in \mathcal{B} \cup \{\varepsilon, \rangle\}: & \text{the received symbol.} \end{array}\right\} \tag{9}$$

Here we've added two more possible 'received symbols': the 'end of message' symbol $\rangle$, and ε, which stands for an empty string. Moreover, in (9) we added a fourth possible type e meaning that we received the 'end of message' symbol $\rangle$. Such an arrow doesn't occur in a trellis like (3), as such a trellis ends with the last received symbol—which is natural when the trellis arises from a given input-output pair. The fourth type e (for *ending*) will be necessary for the probabilistic considerations following in the sequel.

If we imagine the transmission process as a process to find an editing path in a trellis, we would have, at each panel of the trellis, the decision which one of the possible arrows to follow next. Modeling this decision by assigning a probability $p_t(r,c) \in [0,1]$ to each arrow, we have to require that, for a fixed context $c = C_T(i,j)$,

$$p_d(\varepsilon, c) + p_e(\rangle, c) + \sum_{r \in \mathcal{B}} \bigl(p_s(r,c) + p_i(r,c)\bigr) = 1. \tag{10}$$

As there are infinitely many possible contexts $C_T(i,j)$ and we have only finitely many input-output pairs for training, it is necessary to project a given context onto an appropriate candidate from a finite set of context prototypes. To fix ideas, use (7) and denote by

$$\mathcal{C}(\mathcal{A},\mathcal{B}) := (\{\langle\} \times \mathcal{B}^*) \times (\{\langle\} \times \mathcal{A}^*) \times (\mathcal{A}^* \times \{\rangle\})$$

the set of all imaginable contexts. As it doesn't make sense to substitute or delete the 'end of message' character, we introduce, in addition, the notation

$$\mathcal{C}^\circ(\mathcal{A},\mathcal{B}) := (\{\langle\} \times \mathcal{B}^*) \times (\{\langle\} \times \mathcal{A}^*) \times \bigl(\mathcal{A}^+ \times \{\rangle\}\bigr) \subset \mathcal{C}(\mathcal{A},\mathcal{B})$$

for the subset of contexts where the future context comprises more than just the 'end of message' character. The elements of $\mathcal{C}^\circ(\mathcal{A},\mathcal{B})$ are called *inner* contexts, whereas the elements of $\mathcal{C}(\mathcal{A},\mathcal{B}) \setminus \mathcal{C}^\circ(\mathcal{A},\mathcal{B})$ are called *end* contexts.

In many cases it makes sense to consider different contexts for substitution, deletions, insertions, or endings, so we have to use four *context projections*, which are assumed to be surjective:

$$\begin{tikzcd} & \mathcal{C}_s(\mathcal{A},\mathcal{B}) \\ \mathcal{C}^\circ(\mathcal{A},\mathcal{B}) \arrow[ur, "\pi_s"] \arrow[r, "\pi_d"] & \mathcal{C}_d(\mathcal{A},\mathcal{B}) \\ \bigcap & \\ \mathcal{C}(\mathcal{A},\mathcal{B}) \arrow[r, "\pi_i"] \arrow[dr, "\pi_e"'] & \mathcal{C}_i(\mathcal{A},\mathcal{B}) \\ & \mathcal{C}_e(\mathcal{A},\mathcal{B}) \end{tikzcd} \tag{11}$$

By definition, a *context structure* Π consists of four algorithms to evaluate these maps π_s, π_d, π_i, and π_e. We will call a context structure *finite*, if all the ranges $\mathcal{C}_s(\mathcal{A},\mathcal{B})$, $\mathcal{C}_d(\mathcal{A},\mathcal{B})$, $\mathcal{C}_i(\mathcal{A},\mathcal{B})$, and $\mathcal{C}_e(\mathcal{A},\mathcal{B})$, are finite.

Example 1 To give a simple example of a finite context structure, define

$$\left.\begin{aligned} \mathcal{C}_s(\mathcal{A},\mathcal{B}) := \mathcal{C}_d(\mathcal{A},\mathcal{B}) &:= \mathcal{A}, \\ \mathcal{C}_e(\mathcal{A},\mathcal{B}) &:= \mathcal{A} \cup \{\,\rangle\,\}, \\ \mathcal{C}_i(\mathcal{A},\mathcal{B}) &:= \{\varepsilon\}, \\ \pi_s(C_T(i,j)) := \pi_d(C_T(i,j)) &:= c_{j+1}, \\ \pi_e(C_T(i,j)) &:= c_{j+1}, \\ \pi_i(C_T(i,j)) &:= \varepsilon. \end{aligned}\right\} \tag{12}$$

Note that there is no substitution arrow or deletion arrow in a situation with $j = \ell_0$, whence $c_{j+1} \in \mathcal{A}$ is always well-defined. The structure defined in (12) describes one of the simplest reasonable ones, meaning: for substitution or deletion, consider just the symbol to be consumed; for insertion, don't consider any context; to get the probability for ending, consider just the next coming symbol. Note that, if $C_T(i,j)$ is an end context, then the next coming symbol may be the 'end of message' symbol $\rangle$. □

In practise, it is recomendable to use more symbols from $C_T(i,j)$ as far as training data and available storing space allow. The projections π_s, π_d, π_i, π_e, respectively, should map a given context to the longest matching one in $\mathcal{C}_s(\mathcal{A},\mathcal{B})$, $\mathcal{C}_d(\mathcal{A},\mathcal{B})$, $\mathcal{C}_i(\mathcal{A},\mathcal{B})$, $\mathcal{C}_e(\mathcal{A},\mathcal{B})$, respectively, and there has to be fixed a strategy for cases when there is more than one longest match. A good algorithm dealing with context-dependent matching has been evaluated by Kölbl [5].

Let us see how to state formula (10) when the context c is evaluated according to a context structure with four projections. Then the assignment of (estimates for) probabilities needs four functions

$$\begin{aligned} p_s &: \mathcal{B} \times \mathcal{C}_s(\mathcal{A},\mathcal{B}) \to [0,1], \\ p_d &: \mathcal{C}_d(\mathcal{A},\mathcal{B}) \to [0,1], \\ p_i &: \mathcal{B} \times \mathcal{C}_i(\mathcal{A},\mathcal{B}) \to [0,1], \\ p_e &: \mathcal{C}_e(\mathcal{A},\mathcal{B}) \to [0,1]. \end{aligned} \tag{13}$$

As there is no danger of confusion, they bear the same names as the functions occurring in (10). A statement of condition (10) in these terms is

$$\begin{aligned} &\forall \gamma \in \mathcal{C}^\circ(\mathcal{A},\mathcal{B}) : \\ &\quad p_d(\pi_d(\gamma)) + p_e(\pi_e(\gamma)) + \sum_{r\in\mathcal{B}} (p_i(r,\pi_i(\gamma)) + p_s(r,\pi_s(\gamma))) = 1, \end{aligned} \tag{14}$$

$$\forall \gamma \in \mathcal{C}(\mathcal{A},\mathcal{B}) \setminus \mathcal{C}^\circ(\mathcal{A},\mathcal{B}) : \qquad p_e(\pi_e(\gamma)) + \sum_{r\in\mathcal{B}} p_i(r,\pi_i(\gamma)) = 1. \tag{15}$$

Note that the finiteness of the sets $\mathcal{C}_s(\mathcal{A},\mathcal{B})$, $\mathcal{C}_d(\mathcal{A},\mathcal{B})$, $\mathcal{C}_i(\mathcal{A},\mathcal{B})$, and $\mathcal{C}_e(\mathcal{A},\mathcal{B})$, implies that, in total, each of (14) and (15) give finitely many equations.

The ultimate aim of channel modeling is to estimate the probability measures ν_x defining the channel as described in the introduction. For a given input sequence x, thought to be infinite in both directions, ν_x is a probability measure on the space $\mathcal{B}^{\mathbb{Z}}$ of possible output sequences. Assuming that we already received symbols

$$\ldots, y_{-n}, \ldots, y_{-2}, y_{-1}, \tag{16}$$

let us define $C_-(y)$ to be the set of all sequences $z \in \mathcal{B}^{\mathbb{Z}}$ with $z_k = y_k$ for indices $k < 0$. In addition, for $\beta \in \mathcal{B}$ denote by $C_0(\beta) \subset \mathcal{B}^{\mathbb{Z}}$ the set of all output sequences z with $z_0 = \beta$. Then ν_x gives rise to a probability measure on $\mathcal{B}$ assigning to each $\beta \in \mathcal{B}$ the conditional probability $\nu_{x,y}$ that the next received symbol is β, under the condition that x is emitted and the part of y with negative indices has already been received:

$$\nu_{x,y}(\beta) := \frac{\nu_x(C_-(y) \cap C_0(\beta))}{\nu_x(C_-(y))}. \tag{17}$$

To see what is the relation between functions (13) and the $\nu_{x,y}(\beta)$, a more subtle modeling is necessary. Choose a symbol 0 not occurring in the input alphabet $\mathcal{A}$, let $\mathcal{A}_0 := \mathcal{A} \cup \{0\}$, and use the following notation for the set of input strings:

$$\mathcal{S}_{\text{in}}(\mathcal{A}) := \left\{ x \in (\mathcal{A}_0)^{\mathbb{Z}} \;\middle|\; \begin{array}{l} \text{there are integers } \ell_-(x) < 0 \leq \ell_+(x) \\ \text{such that } x_j \in \mathcal{A} \text{ for } \ell_-(x) < j < \ell_+(x) \\ \text{and } x_j = 0 \text{ otherwise.} \end{array} \right\}.$$

Similarly, supposing that $0 \notin \mathcal{B}$ and putting $\mathcal{B}_0 := \mathcal{B} \cup \{0\}$, the set of output strings is denoted by

$$\mathcal{S}_{\text{out}}(\mathcal{B}) := \left\{ y \in (\mathcal{B}_0)^{\mathbb{Z}} \;\middle|\; \begin{array}{l} \text{there are integers } \ell_-(y) < 0 \leq \ell_+(y) \\ \text{such that } y_j \in \mathcal{B} \text{ for } \ell_-(y) < j < \ell_+(y) \\ \text{and } y_j = 0 \text{ otherwise.} \end{array} \right\}.$$

These sets are, in some sense, 'fixed at the origin', meaning the following: If we take $x \in \mathcal{S}_{\text{in}}(\mathcal{A})$ and $y \in \mathcal{S}_{\text{out}}(\mathcal{B})$, omit the 0's to obtain finite strings, and close each of them by symbols $\langle$ and $\rangle$, we obtain

$$\begin{aligned} \hat{x} &:= \langle x_{\ell_-(x)+1} \cdots x_{\ell_+(x)-1} \rangle, \\ \hat{y} &:= \langle y_{\ell_-(y)+1} \cdots y_{\ell_+(y)-1} \rangle. \end{aligned}$$

If we build the trellis T from them, then (y_{-1}, x_0) is the label of a panel (i, j), say, in this trellis. The context of this panel is

$$C_T(i,j) = \Big(\, \langle y_{\ell_-(y)+1} \cdots y_{-1} \; , \; \langle x_{\ell_-(x)+1} \cdots x_{-1} \; , \; x_0 \cdots x_{\ell_+(x)-1} \rangle \, \Big).$$

If π is one the maps $\{\pi_s, \pi_d, \pi_i, \pi_e\}$ from (11), then we write for abbreviation

$$\pi(\hat{y}, \hat{x}) := \pi(C_T(i,j))$$

such that y_{-1} is the just received symbol and x_0 is the next coming symbol.

Observe that $\nu_{x,y}(\beta)$ is defined according to (17) for $x \in \mathcal{S}_{\text{in}}(\mathcal{A})$, $y \in \mathcal{S}_{\text{out}}(\mathcal{B})$, and $\beta \in \mathcal{B}_0$. In order to express $\nu_{x,y}(\beta)$ in terms of the π's (11) and the p's (13), we need the *shift map* by m steps to the left defined by

$$(\cdot)^{(m)} : (\mathcal{A}_0)^{\mathbb{Z}} \to (\mathcal{A}_0)^{\mathbb{Z}}, \quad x_n^{(m)} := x_{n+m} \quad \text{for each } n \in \mathbb{Z}.$$

Now $\nu_{x,y}(\beta)$ should be the probability that $y_0 = \beta$ under the condition that x is the input string and y has been received up to y_{-1}. In principle, there are two possibilities for $y_0 = \beta$: either β is inserted now by a substitution or an insertion, or the next operation is a deletion and β is inserted later. If $\beta = \rangle$, there is the additional possibility that β is inserted by an operation of type e which has been assigned a probability by the function p_e from (13).

Suppose first that $\beta \in \mathcal{B}$. Then the probability that β is put into the output by the subsequent editing operation is

$$p_i\left(\beta, \pi_i(\hat{y}, \hat{x})\right) + p_s\left(\beta, \pi_s(\hat{y}, \hat{x})\right).$$

The probability that β is put into the output by a deletion followed by a substitution or insertion is

$$p_d\left(\pi_d(\hat{y}, \hat{x})\right)\left(p_i\left(\beta, \pi_i(\hat{y}, \hat{x}')\right) + p_s\left(\beta, \pi_s(\hat{y}, \hat{x}')\right)\right).$$

If there are exactly m deletions before β is inserted, then the probability is

$$\begin{aligned} P(\beta \mid x, y, m) := \left(\prod_{j=0}^{m-1} p_d\big(\pi_d(\hat{y}, \hat{x}^{(j)})\big)\right) \cdot \\ \cdot \Big(p_i\big(\beta, \pi_i(\hat{y}, \hat{x}^{(m)})\big) + p_s\big(\beta, \pi_s(\hat{y}, \hat{x}^{(m)})\big)\Big), \end{aligned} \tag{18}$$

where the empty product is set to 1. Because the 'end of message' symbol $\rangle$ has index $\ell_+(x) \geq 0$, there are at most $\ell_+(x)$ symbols $\neq \rangle$ that can be deleted. As the 'end of message' symbol cannot be substituted, formula (18) is only valid for

$$m \in \{0, \ldots, \ell_+(x) - 1\}.$$

For $m = \ell_+(x)$, we have

$$P(\beta \mid x, y, \ell_+(x)) = \left(\prod_{j=0}^{\ell_+(x)-1} p_d\big(\pi_d(\hat{y}, \hat{x}^{(j)})\big)\right) \cdot p_i\big(\beta, \pi_i(\hat{y}, \hat{x}^{(\ell_+(x))})\big). \tag{19}$$

In the case $\beta = 0$, we have to compute the probability that the next received symbol is the 'end of message' character $\rangle$. This probability is measured by the concatenation

$$p_e \circ \pi_e : \mathcal{C}(\mathcal{A}, \mathcal{B}) \to \mathcal{C}_e(\mathcal{A}, \mathcal{B}) \to [0, 1].$$

In addition, we have to take into account that the insertion of $\rangle$ may be preceded by at most $\ell_+(x)$ deletions. If there are exactly $m \in \{0, \ldots, \ell_+(x)\}$ deletions in advance, then

$$P(0 \mid x, y, m) = \left(\prod_{j=0}^{m-1} p_d\big(\pi_d(\hat{y}, \hat{x}^{(j)})\big) \right) \cdot p_e\big(\pi_e(\hat{y}, \hat{x}^{(m)})\big). \tag{20}$$

The next result formalizes the idea that the quantities $P(\beta \mid x, y, m)$ are useful for channel modeling.

Theorem 1 *Suppose that $x \in \mathcal{S}_{\text{in}}(\mathcal{A})$, $y \in \mathcal{S}_{\text{out}}(\mathcal{B})$, and $\beta \in \mathcal{B}_0$,*

$$\nu_{x,y}(\beta) := \sum_{m=0}^{\ell_+(x)} P(\beta \mid x, y, m),$$

then $\quad \forall x \in \mathcal{S}_{\text{in}}(\mathcal{A}) \quad \forall y \in \mathcal{S}_{\text{out}}(\mathcal{B}) : \quad \displaystyle\sum_{\beta \in \mathcal{B}_0} \nu_{x,y}(\beta) = 1.$

Proof First let $m \in \{0, \ldots, \ell_+(x) - 1\}$, and let us compute, using (18), (20):

$$\begin{aligned}
&\sum_{\beta \in \mathcal{B}_0} P(\beta \mid x, y, m) = \\
&\quad = \left(\prod_{j=0}^{m-1} p_d\big(\pi_d(\hat{y}, \hat{x}^{(j)})\big) \right) \cdot \Bigg(p_e\big(\pi_e(\hat{y}, \hat{x}^{(m)})\big) + \\
&\qquad\qquad + \sum_{\beta \in \mathcal{B}} \Big(p_i\big(\beta, \pi_i(\hat{y}, \hat{x}^{(m)})\big) + p_s\big(\beta, \pi_s(\hat{y}, \hat{x}^{(m)})\big) \Big) \Bigg) \\
&\quad = \left(\prod_{j=0}^{m-1} p_d\big(\pi_d(\hat{y}, \hat{x}^{(j)})\big) \right) \cdot \Big(1 - p_d\big(\pi_d(\hat{y}, \hat{x}^{(m)})\big) \Big),
\end{aligned} \tag{21}$$

where, in addition, (14) has been used for the last equation. Next consider

$m = \ell_+(x)$. We get, using (19), (20):

$$\sum_{\beta\in\mathcal{B}_0} P(\beta \mid x, y, \ell_+(x)) =$$

$$= \left(\prod_{j=0}^{\ell_+(x)-1} p_d\big(\pi_d(\hat{y}, \hat{x}^{(j)})\big)\right) \cdot$$

$$\cdot \left(p_e\big(\pi_e(\hat{y}, \hat{x}^{(\ell_+(x))})\big) + \sum_{\beta\in\mathcal{B}} p_i\big(\beta, \pi_i(\hat{y}, \hat{x}^{(\ell_+(x))})\big)\right),$$

and (15) proves

$$\sum_{\beta\in\mathcal{B}_0} P(\beta \mid x, y, \ell_+(x)) = \left(\prod_{j=0}^{\ell_+(x)-1} p_d\big(\pi_d(\hat{y}, \hat{x}^{(j)})\big)\right). \tag{22}$$

To prove the theorem, it remains to sum the terms (21) from $m = 0$ to $m = \ell_+(x) - 1$, and then add the term in (22). But this sum is telescoping giving 1, which completes the proof of the theorem. ■

4 Expectation-estimation algorithm

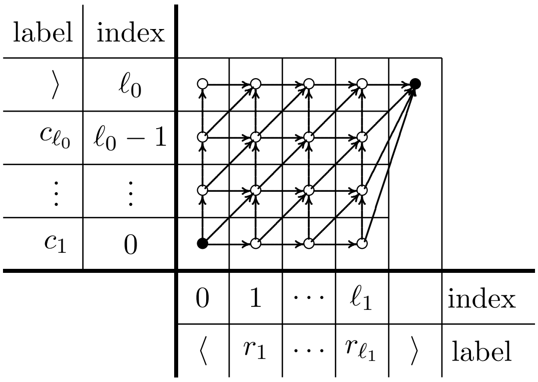
The functions p_s, p_d, p_i, and p_e, announced in (13) to contain the transition probabilities needed for channel modeling are 'trained' using an algorithm similar to a well-known statistical method called *expectation maximization* (abbreviated 'EM'); for a general account, see, for instance, Bishop [2], p. 450f. EM is especially apt when parameters are to be estimated whose direct observation is not possible, e.g., the probabilities of hidden transitions in a trellis.

Like the EM-algorithm, the algorithm presented here starts with an initialization for the parameters to be constructed in a *probability matrix* and improves their values in each step of a *main loop.* The main loop first initializes a *count matrix*, and then triggers a *training loop* over a given training corpus of weighted input-output pairs. For each input-output pair, a trellis is built and evaluated in the sense that to each arrow a *weighted count* is associated, which is then used to increment the count matrix. Having processed all given input-output pairs, the count is the basis for updating the probability matrix. In the classical EM-algorithm, updating is achieved by maximizing some likelhood function. Here we replace maximization by a more general probability estimation step. We shall see that numerical stability would not be guaranteed without certain 'smoothness' conditions, which is one reason for seeking another, but mathematically founded, estimation procedure.

Let us now have a look at the training loop or *inner loop*, and let us fix for the moment an input string $c_1 \dots c_{\ell_0} \in \mathcal{S}_{\rm in}(\mathcal{A})$ and an output string $r_1 \dots r_{\ell_1} \in \mathcal{S}_{\rm out}(\mathcal{B})$. We use the trellis (23) which is an augmented version of (3), augmented by arrows from the last column, i.e. panels with indices $(\ell_1, 0), \dots, (\ell_1, \ell_0)$ and labels

$$(r_{\ell_1}, c_1), \dots, (r_{\ell_1}, c_{\ell_0}), (r_{\ell_1}, \rangle\,),$$

to an additional panel labeled $(\,\rangle\,, \rangle\,)$ (we don't need indices for this additional panel). These augmenting arrows will carry the information used to 'train' the function p_e.



(23)

In the following table we indentify to each editing operation the corresponding arrows in this trellis.

Editing operation	starting indices (i, j) of arrows
Substitution:	$0 \le i \le \ell_1 - 1$ and $0 \le j \le \ell_0 - 1$,
Deletion:	$0 \le i \le \ell_1$ and $0 \le j \le \ell_0 - 1$,
Insertion:	$0 \le i \le \ell_1 - 1$ and $0 \le j \le \ell_0$,
Ending:	$i = \ell_1$ and $0 \le j \le \ell_0$.

(24)

To describe the algorithm, assume a context structure

$$\Pi = (\pi_s, \pi_d, \pi_i, \pi_e)$$

as depicted in (11) to be given. Then the algorithm operates on the following data:

- A real vector

$$\widehat{P} = \Big(\big(\hat{p}_s(r,\gamma) : r \in \mathcal{B}, \gamma \in \mathcal{C}_s(\mathcal{A},\mathcal{B})\big), \big(\hat{p}_d(\gamma) : \gamma \in \mathcal{C}_d(\mathcal{A},\mathcal{B})\big), \\ \big(\hat{p}_i(r,\gamma) : r \in \mathcal{B}, \gamma \in \mathcal{C}_i(\mathcal{A},\mathcal{B})\big), \big(\hat{p}_e(\gamma) : \gamma \in \mathcal{C}_e(\mathcal{A},\mathcal{B})\big) \Big)$$

  containing the current estimates for the probabilities. This vector is held constant during each execution of the body of the main loop.

- A real vector

$$C = \Big(\big(C_s(r,\gamma) : r \in \mathcal{B}, \gamma \in \mathcal{C}_s(\mathcal{A},\mathcal{B}) \big), \big(C_d(\gamma) : \gamma \in \mathcal{C}_d(\mathcal{A},\mathcal{B}) \big), \\ \big(C_i(r,\gamma) : r \in \mathcal{B}, \gamma \in \mathcal{C}_i(\mathcal{A},\mathcal{B}) \big), \big(C_e(\gamma) : \gamma \in \mathcal{C}_e(\mathcal{A},\mathcal{B}) \big) \Big)$$

  containing the count data, which is updated after each execution of the body of the inner loop. Note that we will use weighted counts, whence the components of this vector are non-negative real numbers.

- Three real matrices and a real vector containing the weighted count data associated to the different editing arrows in an inner loop trellis (23). The dimensions of this structures can be taken from (24):

  - Substitution counts: $S(i,j)$ with $0 \le i \le \ell_1 - 1$ and $0 \le j \le \ell_0 - 1$.
  - Deletion counts: $D(i,j)$ with $0 \le i \le \ell_1$ and $0 \le j \le \ell_0 - 1$.
  - Insertion counts: $I(i,j)$ with $0 \le i \le \ell_1 - 1$ and $0 \le j \le \ell_0$.
  - Ending counts: $E(j)$ with $0 \le j \le \ell_0$.

  These four structures are initialized by the current estimates for the probabilities which are available in $\widehat{P}$ at the beginning of the body of the training loop, and are no longer used after updating the count data.

- A real matrix F of size $(\ell_1 + 1) \times (\ell_0 + 1)$ to compute the current estimate $\hat{p}_{\mathrm{io}}$ for the total probability of an input-output pair. This structure is only used in the forward step of the inner loop, where it is initialized anew for each input-output pair.

We are now ready to give the algorithm in detail. To start, we first give a survey to describe its general structure in pseudo-code.

```
Initialize the vector $\widehat{P}$ with appropriate initial probabilities.
Main loop.
    Initialize the count vector $C$ by zeros.
    Expectation step.
        Training loop. Do for each input-output pair:
            Initialize the structures $S$, $D$, $I$, and $E$ using $\widehat{P}$.
            Forward step: Compute $\hat{p}_{\mathrm{io}}$.
            Backward step: Normalize $S$, $D$, $I$, and $E$.
            Update the count vector $C$.
            Continue with next input-output pair.
        End of Training loop.
    Estimation step.
        Determine   the next vector $\widehat{P}$ from the count vector $C$.
    Continue or end,   the decision being based on some stop condition.
End of main loop.
```

Next we go deeper into the details, beginning with the expectation step which consists of a loop over all input-output pairs in a previously specified training corpus. For each training pair $(s_{\text{in}}, s_{\text{out}})$, we use our current estimate of probabilities to approximately compute the conditional probability $\hat{p}_{\text{io}}$ for receiving s_{out} when s_{in} is transmitted through the channel. To compute the total probability to receive s_{out} when s_{in} is sent from the current estimates recorded in $\widehat{P}$, we would have to consider all possible editing paths leading from s_{in} to s_{out} and sum up estimates for the path probabilities. In order to compute a path probability, we would have to multiply the conditional (i.e., context-dependent) probabilities of the elementary editing operations the path consists of.

In this algorithm, we only take into account the paths occurring in the trellis, which are just the compositions of arrows leading from the panel indexed $(0, 0)$ to the panel labeled $(\rangle\, , \rangle\,)$; note that the number of these paths can be computed using formula (5). The restriction to this trellis is forced by the choice of what kind of arrows are considered, i.e., by the types of elementary editing operations and associated movings of the pointer to the next coming symbol.

In order to compute the probability estimate $\hat{p}_{\text{io}}$, we do not expand all paths but use a trick from the statistical treatment of Hidden Markov chains which is well-known as *Baum-Welch* algorithm, see [1] for an early source. It consists of computing, for each panel (i, j), the total probability of all trellis paths leading from panel $(0, 0)$ to it, which can easily be done line by line, starting with the lowest line and working through each line from left to right (or, alternatively, row by row, starting with the left-most row and working through each row upwards).

Our first step for the inner loop is initialization of our auxiliary structures.

```
Initialize   the structures $S$, $D$, $I$, and $E$ using $\widehat{P}$:
    for   $0 \le i \le \ell_1 - 1$ and $0 \le j \le \ell_0 - 1$
        put   $S(i, j) := \hat{p}_s(r_i, \pi_s(C_T(i, j)))$
    for   $0 \le i \le \ell_1$ and $0 \le j \le \ell_0 - 1$
        put   $D(i, j) := \hat{p}_d(r_i, \pi_d(C_T(i, j)))$
    for   $0 \le i \le \ell_1 - 1$ and $0 \le j \le \ell_0$
```

```
        put  $I(i,j) := \hat{p}_i(r_i, \pi_i(C_T(i,j)))$
    for  $0 \le j \le \ell_0$
        put  $E(j) := \hat{p}_e(\pi_e(C_T(\ell_1, j)))$
```

End of Initialize.

Then the forward step to compute our approximation $\hat{p}_{\rm io}$ proceeds line by line through the trellis, starting with the lowest line and moving upwards up to the line with index ℓ_1:

Forward step to compute $\hat{p}_{\rm io}$.

```
    put  $F(0,0) := 1$,
    for  $i = 1$  step  $1$  while  $i \le \ell_1$  do
        put  $F(i,0) := F(i-1,0) * I(i-1,0)$,
    for  $j = 1$  step  $1$  while  $j \le \ell_0$  do
        put  $F(0,j) := F(0,j-1) * D(0,j-1)$,
        for  $i = 1$  step  $1$  while  $i \le \ell_1$  do
            put  $F(i,j) := F(i-1,j-1) * S(i-1,j-1)$,
            augment  $F(i,j)$  by  $F(i-1,j) * D(i-1,j)$,
            augment  $F(i,j)$  by  $F(i,j-1) * I(i,j-1)$,
    put  $\hat{p}_{\rm io} := F(\ell_1,0) * E(0)$,
    for  $j = 1$  step  $1$  while  $j \le \ell_0$  do
        augment  $\hat{p}_{\rm io}$  by  $F(\ell_1,j) * E(j)$,
```

End of Forward step.

To use the data from the present input-output pair $(s_{\rm in}, s_{\rm out})$ for incrementing our count vector, a possibility would be to choose the path which is 'most likely' according to the current probability estimates, and increment by 1 the corresponding context-dependent count for each arrow of this maximal path. But it seems to be more natural, and more precise, to use the data collected in the structures S, D, I, and E, to compute *weighted counts* for incrementation. To this end, we first compute approximations to the conditional arrow probabilities, conditioned both on contexts and the given observation $(s_{\rm in}, s_{\rm out})$. This is achieved by taking, for each arrow, the quotient of the initial arrow probability as above and $\hat{p}_{\rm io}$.

Backward step to normalize S, D, I, and E.

```
    for  $0 \le i \le \ell_1 - 1$ and $0 \le j \le \ell_0 - 1$
        replace  $S(i,j)$  by  $S(i,j)/\hat{p}_{\rm io}$,
    for  $0 \le i \le \ell_1$ and $0 \le j \le \ell_0 - 1$
```

```
        replace  $D(i,j)$  by  $D(i,j)/\hat{p}_{\mathrm{io}}$,
    for  $0 \le i \le \ell_1 - 1$ and $0 \le j \le \ell_0$
        replace  $I(i,j)$  by  $I(i,j)/\hat{p}_{\mathrm{io}}$,
    for  $0 \le j \le \ell_0 - 1$
        replace  $E(j)$  by  $E(j)/\hat{p}_{\mathrm{io}}$,
End of Backward step.
```

Then the obtained *wieghted counts* are used to increment (or *update*) the count vector.

```
Update  the count vector $C$:
    for  $0 \le i \le \ell_1 - 1$ and $0 \le j \le \ell_0 - 1$
        augment  $N_s(r_i, \pi_s(C_T(i,j)))$  by  $S(i,j)$,
    for  $0 \le i \le \ell_1$ and $0 \le j \le \ell_0 - 1$
        augment  $N_d(\pi_d(C_T(i,j)))$  by  $D(i,j)$,
    for  $0 \le i \le \ell_1 - 1$ and $0 \le j \le \ell_0$
        augment  $N_i(r_i, \pi_i(C_T(i,j)))$  by  $I(i,j)$,
    for  $0 \le j \le \ell_0 - 1$
        augment  $N_e(\pi_e(C_T(\ell_1, j)))$  by  $E(j)$.
End of Update.
```

There are three steps of the algorithm remaining: initialization, determination of the next $\widetilde{P}$, and the stop condition. The problem how to determine the next $\widetilde{P}$ from the count vector C will be dealt with in the next section. The stop condition is not really addressed on in this paper: A good idea would be to use some measure for the size of the difference between the current probability estimate $\widehat{P}$ and the preceding one, and, in addition, to bound in advance the total looping number of the main loop.

To see the numerical problem in which we would run with a too naïve initialization, consider a situation where input alphabet and output alphabet coincide, $\mathcal{A} = \mathcal{B}$, and where initialization of $\widehat{P}$ is done in the 'natural' way, i.e., for any given context, *self-substitution* gets probability one and any error gets probability zero. With this current probability estimate, if there is a training pair $(s_{\mathrm{in}}, s_{\mathrm{out}})$ with $s_{\mathrm{in}} \neq s_{\mathrm{out}}$, we would have $\hat{p}_{\mathrm{io}} = 0$ for this pair, and the backward step would be impossible. To guarantee possibility of the backward step of the described algorithm, we choose a real threshold $m > 0$, and impose the following condition:

$$\text{each component of } \widehat{P} \text{ is to be } \ge m. \tag{25}$$

Possibility of the backward step is clearly guaranteed if the threshold fulfills $m^{\ell_0+\ell_1} > x_{\mathrm{min}}$, where (ℓ_0, ℓ_1) are the maximal lengths of input-output pairs, and x_{min} is the smallest positive number expressible on the employed computation machine.

5 Estimation of probabilities

The vector $\widehat{P}$ needed in the algorithm is structured according to the context structure employed, and subject to conditions (14) and (15). Hence, $\widehat{P}$ is much more complicate than a standard probability vector which may be defined as a finite sequence of non-negative real numbers which add up to one. Estimating such a probability vector from data, being a special point estimation problem, is well-studied in probability theory, see, for instance, Lehmann and Casella [6]. The aim of this section to give conditions on the context structure which imply that the problem of estimating a vector $\widehat{P}$ of probabilities as it is needed in the algorithm can be reduced to finitely many such point estimation problems. An method based on a Bayesian procedure using a loss function leading to a probability vector meeting condition (25), is going to be described in [3].

Let $\Pi := (\pi_s, \pi_d, \pi_i, \pi_e)$ be a finite context structure consisting of four algorithmically defined maps as in (11),

$$\begin{aligned} &\pi_s : \mathcal{C}^\circ(\mathcal{A},\mathcal{B}) \to \mathcal{C}_s(\mathcal{A},\mathcal{B}), \quad &&\pi_d : \mathcal{C}^\circ(\mathcal{A},\mathcal{B}) \to \mathcal{C}_d(\mathcal{A},\mathcal{B}), \\ &\pi_i : \mathcal{C}(\mathcal{A},\mathcal{B}) \to \mathcal{C}_i(\mathcal{A},\mathcal{B}), \quad &&\pi_e : \mathcal{C}(\mathcal{A},\mathcal{B}) \to \mathcal{C}_e(\mathcal{A},\mathcal{B}). \end{aligned}$$

The algorithm described in the preceding section requires a fixed context structure Π. Then both the probability vector $\widehat{P}$ and the count vector C, as they occur in the algorithm, look like

$$\begin{aligned} Q = \Big(&\big(Q_s(r,\gamma) : r \in \mathcal{B}, \gamma \in \mathcal{C}_s(\mathcal{A},\mathcal{B})\big), \big(Q_d(\gamma) : \gamma \in \mathcal{C}_d(\mathcal{A},\mathcal{B})\big), \\ &\big(Q_i(r,\gamma) : r \in \mathcal{B}, \gamma \in \mathcal{C}_i(\mathcal{A},\mathcal{B})\big), \big(Q_e(\gamma) : \gamma \in \mathcal{C}_e(\mathcal{A},\mathcal{B})\big)\Big), \end{aligned} \tag{26}$$

where each entry of this vector is a non-negative real number. We say that a vector Q *corresponds* to a context structure Π, if it can be written in the form (26). This means, Q corresponds to Π if and only if Q is indexed on the index set

$$\mathcal{I}_\Pi := \mathcal{I}(\pi_s) \cup \mathcal{I}(\pi_d) \cup \mathcal{I}(\pi_i) \cup \mathcal{I}(\pi_e),$$

where

$$\begin{aligned} &\mathcal{I}(\pi_s) := \{s\} \times \mathcal{B} \times \mathcal{C}_s(\mathcal{A},\mathcal{B}), \quad &&\mathcal{I}(\pi_d) := \{d\} \times \mathcal{C}_d(\mathcal{A},\mathcal{B}), \\ &\mathcal{I}(\pi_i) := \{i\} \times \mathcal{B} \times \mathcal{C}_i(\mathcal{A},\mathcal{B}), \quad &&\mathcal{I}(\pi_e) := \{e\} \times \mathcal{C}_e(\mathcal{A},\mathcal{B}). \end{aligned}$$

Comparing this notation to (26), we infer that a real vector Q which corresponds to a context structure Π can be viewed as a map

$$Q : \mathcal{I}_\Pi \to \mathbb{R}. \tag{27}$$

The projections π_s, π_d, π_i, and π_e together define a map Δ_Π from all possible contexts to the power set of $\mathcal{I}_\Pi$, which assigns to each context $\gamma \in$

$\mathcal{C}(\mathcal{A},\mathcal{B})$ the set of those indices of a vector corresponding to Π which contain a projection of γ. To get an explicit expression of this map, first define, for each type $t \in \{s,d,i,e\}$, the set-valued map $\delta_t : \mathcal{C}(\mathcal{A},\mathcal{B}) \to \mathcal{P}(\mathcal{I}(\pi_t))$ by

$$\begin{aligned}
\delta_s(\gamma) &:= \begin{cases} \varnothing & \text{if } \gamma \notin \mathcal{C}^\circ(\mathcal{A},\mathcal{B}), \\ \{(s,r,\pi_s(\gamma)) \mid r \in \mathcal{B}\} & \text{otherwise,} \end{cases} \\
\delta_d(\gamma) &:= \begin{cases} \varnothing & \text{if } \gamma \notin \mathcal{C}^\circ(\mathcal{A},\mathcal{B}), \\ \{(d,\pi_d(\gamma))\} & \text{otherwise,} \end{cases} \\
\delta_i(\gamma) &:= \{(i,r,\pi_i(\gamma)) \mid r \in \mathcal{B}\}, \\
\delta_e(\gamma) &:= \{(e,\pi_e(\gamma))\}.
\end{aligned} \tag{28}$$

Then the map $\Delta_\Pi : \mathcal{C}(\mathcal{A},\mathcal{B}) \to \mathcal{P}(\mathcal{I}_\Pi)$ is given by

$$\Delta_\Pi(\gamma) := \delta_s(\gamma) \cup \delta_d(\gamma) \cup \delta_i(\gamma) \cup \delta_e(\gamma). \tag{29}$$

With this notation, for a vector Q corresponding to Π, the conjunction of conditions (14) and (15) is equivalent to the condition

$$\forall \gamma \in \mathcal{C}(\mathcal{A},\mathcal{B}) : \quad \sum_{x \in \Delta_\Pi(\gamma)} Q(x) = 1. \tag{30}$$

As the set of possible contexts $\mathcal{C}(\mathcal{A},\mathcal{B})$ is infinite, condition (30) involves infinitely many equations: one for each γ. The set of variables which occur in these equations is $\mathcal{I}_\Pi$, which is a finite set, whence the set of *different* equations among the equations (30) is also finite. The problem is that the equations cannot be treated independently, as there may be different equations sharing variables. This means that there may be contexts $\gamma, \gamma' \in \mathcal{C}(\mathcal{A},\mathcal{B})$ with $\Delta_\Pi(\gamma) \neq \Delta_\Pi(\gamma')$ and $\Delta_\Pi(\gamma) \cap \Delta_\Pi(\gamma') \neq \varnothing$.

To understand what this means for the maps defining the context structure, we say that a context structure $\Pi = (\pi_s, \pi_d, \pi_i, \pi_e)$ is *normal*, or *uses equal contexts*, if there are bijections β_s, β_i, and an injection ι_d, which extend diagram (11) to the following commutative diagram:

$$\begin{tikzcd}
 & \mathcal{C}_s(\mathcal{A},\mathcal{B}) \arrow[d, "\beta_s"] \\
\mathcal{C}^\circ(\mathcal{A},\mathcal{B}) \arrow[ru, "\pi_s"] \arrow[r, "\pi_d"] \arrow[d, phantom, "\bigcap"] & \mathcal{C}_d(\mathcal{A},\mathcal{B}) \arrow[d, "\iota_d"] \\
\mathcal{C}(\mathcal{A},\mathcal{B}) \arrow[r, "\pi_i"] \arrow[rd, "\pi_e"'] & \mathcal{C}_i(\mathcal{A},\mathcal{B}) \arrow[d, "\beta_i"] \\
 & \mathcal{C}_e(\mathcal{A},\mathcal{B})
\end{tikzcd} \tag{31}$$

Example 2 The following context structure uses equal contexts:

$$\begin{aligned}
\mathcal{C}_s(\mathcal{A},\mathcal{B}) :=& \quad \mathcal{C}_d(\mathcal{A},\mathcal{B}) := \big(\mathcal{B} \cup \{\,\langle\,\}\big) \times \mathcal{A}, \\
\pi_s(C_T(i,j)) :=& \quad \pi_d(C_T(i,j)) := (r_i, c_{j+1}), \\
\mathcal{C}_e(\mathcal{A},\mathcal{B}) :=& \quad \mathcal{C}_i(\mathcal{A},\mathcal{B}) := \big(\mathcal{B} \cup \{\,\langle\,\}\big) \times \big(\mathcal{A} \cup \{\,\rangle\,\}\big), \\
\pi_i(C_T(i,j)) :=& \quad \pi_e(C_T(i,j)) := (r_i, c_{j+1}).
\end{aligned}$$

Indeed, if we set β_s to be the identity map on $\big(\mathcal{B} \cup \{\,\langle\,\}\big) \times \mathcal{A}$, ι_d to the canonical inclusion $\big(\mathcal{B} \cup \{\,\langle\,\}\big) \times \mathcal{A} \hookrightarrow \big(\mathcal{B} \cup \{\,\langle\,\}\big) \times \big(\mathcal{A} \cup \{\,\rangle\,\}\big)$, and β_i to the identity map on $\big(\mathcal{B} \cup \{\,\langle\,\}\big) \times \big(\mathcal{A} \cup \{\,\rangle\,\}\big)$, then diagram (31) commutes. □

Lemma 2 *For a context structure* Π*, the following assertions are equivalent:*

(a) Π *is normal.*

(b) *For* $\varrho, \varrho' \in \mathcal{C}(\mathcal{A},\mathcal{B})$*, either* $\Delta(\varrho) = \Delta(\varrho')$ *or* $\Delta(\varrho) \cap \Delta(\varrho') = \varnothing$*.*

Proof We will need the following assertion, which is an immediate consequence of (28):

$$\forall\, t, t' \in \{s, d, i, e\} : \quad t \neq t' \Rightarrow \delta_t(\varrho) \cap \delta_t(\varrho') = \varnothing. \tag{32}$$

To prove the implication (a) $\Rightarrow$ (b), let $\varrho, \varrho' \in \mathcal{C}(\mathcal{A},\mathcal{B})$ be such that $\Delta_\Pi(\varrho) \neq \Delta_\Pi(\varrho')$. By definition of Δ_Π, this inequality means

$$\exists\, t \in \{s, d, i, e\} : \quad \delta_t(\varrho) \neq \delta_t(\varrho').$$

This implies $\pi_t(\varrho) \neq \pi_t(\varrho')$, from which we infer by chasing through diagram (31), and thereby using injectivity of β_s, ι_d, and β_i, that

$$\forall\, t \in \{s, d, i, e\} : \quad \begin{cases} \text{either} & t \in \{s, d\} \text{ and } \varrho \notin \mathcal{C}^\circ(\mathcal{A},\mathcal{B}), \\ \text{or} & t \in \{s, d\} \text{ and } \varrho' \notin \mathcal{C}^\circ(\mathcal{A},\mathcal{B}), \\ \text{or} & \pi_t(\varrho) \neq \pi_t(\varrho'). \end{cases}$$

By the construction of the $\delta_t(\varrho)$ in (28), this proves

$$\forall\, t \in \{s, d, i, e\} : \quad \delta_t(\varrho) \cap \delta_t(\varrho') = \varnothing. \tag{33}$$

Again by the definition of Δ_Π, combining this with (32) implies

$$\Delta_\Pi(\varrho) \cap \Delta_\Pi(\varrho') = \varnothing,$$

which completes the proof of (b).

A proof of implication (b) $\Rightarrow$ (a) requires that we prove existence of bijections β_s, β_i, and an injection ι_d, which make diagram (31) commute.

To construct β_s, let $\sigma \in \mathcal{C}_s(\mathcal{A},\mathcal{B})$ and use surjectivity of π_s to choose $\varrho \in \mathcal{C}^\circ(\mathcal{A},\mathcal{B})$ such that $\sigma = \pi_s(\varrho)$. We will prove that

$$\beta_s(\sigma) := \pi_d(\varrho)$$

is well-defined. To see this, suppose that $\varrho' \in \mathcal{C}^\circ(\mathcal{A},\mathcal{B})$ is another context giving $\sigma = \pi_s(\varrho')$. Then, by (28), $\delta_s(\varrho) = \delta_s(\varrho') \neq \varnothing$, which implies $\Delta_\Pi(\varrho) \cap \Delta_\Pi(\varrho') \neq \varnothing$, from which we infer by (b) that

$$\Delta_\Pi(\varrho) = \Delta_\Pi(\varrho').$$

Now (32) and (28) prove that $\pi_d(\varrho) = \pi_d(\varrho')$ as required.

The injection ι_d and the other bijection β_i are defined by

$$\begin{aligned} \iota_d(\sigma) &:= \pi_i(\varrho) \quad \text{for} \quad \sigma \in \mathcal{C}_d(\mathcal{A},\mathcal{B}) \quad \text{and} \quad \varrho \in \pi_d^{-1}(\sigma), \\ \beta_i(\sigma) &:= \pi_e(\varrho) \quad \text{for} \quad \sigma \in \mathcal{C}_i(\mathcal{A},\mathcal{B}) \quad \text{and} \quad \varrho \in \pi_i^{-1}(\sigma), \end{aligned}$$

respectivly. The proofs of well-definedness run exactly as in case of the bijection β_s. ■

Now we are ready to formulate a result which reduces the probability estimation problem of the algorithm to the estimation of finitely many probability measures on finite state spaces from data.

Corollary 3 *Let* Π *be a finite context structure using equal contexts. Then there is a partition of* $\mathcal{I}_\Pi$ *into pairwise disjoint sets* $\Delta_1, \ldots, \Delta_\nu$ *such that condition* (30) *is equivalent to*

$$\forall i \in \{1, \ldots, \nu\} : \quad \sum_{x \in \Delta_i} Q(x) = 1.$$

References

[1] Leonard E. Baum, Ted Petrie, Georges Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Statist.*, 41(1):164–171, 1970.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[3] Hans Fischer and Günther Wirsching. Bayesian one-sided loss probability vector estimation and discretization. *In preparation.*

[4] A. I. Khinchin. *Mathematical Foundations of Information Theory.* Dover Publications, Inc., New York, 1957.

[5] Christian Kölbl. *Algorithmus eines HMM-Compilers. Implementierung und Evaluierung eines Compilers zur Transkription von Phonetik-Strings in HMM-Folgen.* Verlag Dr. Müller, May 2009.

[6] E. L. Lehmann and George Casella. *Theory of Point Estimation.* Springer, New York, 1998.

[7] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

[8] Kirsten Malmkjær, editor. *The Linguistics Encyclopedia.* London and New York, second edition, 2002.

[9] E.S. Ristad and P.N. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(2):522–532, May 1998.

[10] Ronald Römer. *Private communication*, 2007.

[11] Claude Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[12] Peter N. Yianilos. *Topics in Computational Hidden State Modeling.* PhD thesis, Princeton University, Dep. of Computer Science, 1997.