S-restricted monotone alignments: Algorithm, search space, and applications

Steffen Eger
Goethe University Frankfurt, Grueneburg-Platz 1, 60323 Frankfurt am Main, Germany
steffen.eger@yahoo.com

Abstract
We present a simple and straightforward alignment algorithm for monotone many-to-many alignments in grapheme-to-phoneme conversion and related fields such as morphology, and discuss a few noteworthy extensions. Moreover, we specify combinatorial formulas for monotone many-to-many alignments and decoding in G2P which indicate that exhaustive enumeration is generally possible, so that some limitations of our approach can easily be overcome. Finally, we present a decoding scheme, within the monotone many-to-many alignment paradigm, that relates the decoding problem to restricted integer compositions and that is, putatively, superior to alternatives suggested in the literature.

Title and Abstract in German

S-beschränkte monotone Alignierungen: Algorithmus, Suchraum und Anwendungen

Wir präsentieren einen einfachen Alignment-Algorithmus für monotone 'many-to-many' Alignierungen im Bereich Graphem-zu-Phonem-Konversion und verwandten Gebieten wie z.B. Morphologie, und besprechen sinnvolle Erweiterungen. Darüber hinaus geben wir kombinatorische Formeln im Bereich der monotonen 'many-to-many' Alignierungen und im Bereich des Decoding in G2P an, die suggerieren, dass vollständige Enumeration hier im Allgemeinen möglich ist, sodass ein paar Einschränkungen unseres Ansatzes leicht behoben werden können. Schließlich präsentieren wir ein Decoding-Schema, innerhalb des Paradigmas von 'many-to-many' Alignierungen, dass das Decoding-Problem mit beschränkten Zahlenkompositionen in Beziehung setzt und das in der Literatur vorgeschlagenen Alternativen vermeintlich überlegen ist.

Keywords: many-to-many alignments, monotone alignments, string transduction, restricted integer compositions, grapheme-to-phoneme conversion.

Keywords in German: many-to-many Alignierungen, monotone Alignierungen, String-Überführungen, beschränkte Zahlenkompositionen, Graphem-zu-Phonem-Konversion.

Proceedings of COLING 2012: Technical Papers, pages 781–798,
COLING 2012, Mumbai, December 2012.

1 Introduction

Grapheme-to-phoneme conversion (G2P) is the problem of transducing, or converting, a grapheme, or letter, string $\mathbf{x}$ over an alphabet Σ_x into a phoneme string $\mathbf{y}$ over an alphabet Σ_y. A crucial first step thereby is finding *alignments* between grapheme and phoneme strings in training data. The classic alignment paradigm has assumed alignments that were

(i) *one-to-one* or *one-to-zero*; i.e. one grapheme character is mapped to at most one phoneme character; this assumption has probably been a relic of both the traditional assumptions in machine translation (cf. (Brown et al., 1990)) and in biological sequence alignment (cf. (Needleman and Wunsch, 1970)). In the field of G2P such alignment models are also called ϵ-scattering models (cf. (Black et al., 1998)).

(ii) *monotone*, i.e., the order between characters in grapheme and phoneme strings is preserved.

It is clear that, despite its benefits, the classical alignment paradigm has a couple of limitations; in particular, it may be unable to explain certain grapheme-phoneme sequence pairs, a.o. those where the length of the phoneme string is greater than the length of the grapheme string such as in

exact igzækt

where $\mathbf{x}$ has length 5 and $\mathbf{y}$ has length 6. In the same context, even if an input pair can be explained, the one-to-one or one-to-zero assumption may lead to alignments that, linguistically, seem nonsensical, such as

p	h	o	e	n	i	x
f	–	iː	n	i	k	s

where the reader may verify that, no matter where the ϵ is inserted, some associations will always appear unmotivated. Moreover, monotonicity appears in some cases violated as well, such as in the following,

centre sentər

where it seems, linguistically, that the letter character *r* corresponds to phonemic r and graphemic word final *e* corresponds to ə.

Fortunately, better alignment models have been suggested to overcome these problems. For example, (Jiampojamarn et al., 2007) and (Jiampojamarn and Kondrak, 2010) suggest 'many-to-many' alignment models that address issue (i) above. Similar ideas were already present in (Baldwin and Tanaka, 1999), (Galescu and Allen, 2001) and (Taylor, 2005). (Bisani and Ney, 2008) likewise propose many-to-many alignment models; more precisely, their idea is to *segment* grapheme-phoneme pairs into non-overlapping parts ('co-segmentation'), calling each segment a *graphone*, as in

ph	oe	n	i	x
f	iː	n	i	ks

which consists of five graphones.

The purpose of the present paper is to introduce a simple, flexible and general monotone many-to-many alignment algorithm (in Section 3) that competes with the approach suggested in (Jiampojamarn et al., 2007).[1] Thereby, our algorithm is an intuitive and straightforward generalization of the classical Needleman-Wunsch algorithm for (biological or linguistic) sequence alignment. Moreover, we explore valuable extensions of the presented framework, likewise in Section 3, which may be useful e.g. to detect latent classes in alignments, similar to what has been done in e.g. (Dreyer et al., 2008). We also mention limitations of our procedure, in Section 4, and discuss the naive brute-force approach, exhaustive enumeration, as an alternative; furthermore, by specifying the search space for monotone many-to-many alignments, we indicate that exhaustive enumeration appears generally a feasible option in G2P and related fields. Next, in Section 6.1 we briefly mention how we perform training for string transductions in the monotone many-to-many alignment case. Then, a second contribution of this work is to suggest an alternative decoding procedure when transducing strings $\mathbf{x}$ into strings $\mathbf{y}$, within the monotone many-to-many alignment paradigm (in Section 6.2). We thereby relate the decoding problem to restricted integer compositions, a field in mathematical combinatorics that has received increased attention in the last few years (cf. (Heubach and Mansour, 2004), (Malandro, 2012), (Eger, 2012a)). Finally, we demonstrate the superiority of our approach by applying it to several data sets in Section 7.

It must be mentioned, generally, that we take G2P only as an (important) sample application of monotone many-to-many alignments, but that they clearly apply to other fields of natural language processing as well, such as transliteration, morphology/lemmatization, etc. and we thus also incorporate experiments on morphology data. Moreover, as indicated, we do not question the premise of monotonicity in the current work, but take it as a crucial assumption of our approach, leading to efficient algorithms. Still, 'local non-monotonicities' as exemplified above can certainly be adequately addressed within our framework, as should become clear from our illustrations below (e.g. with higher-order 'steps').

2 S-restricted monotone paths and alignments

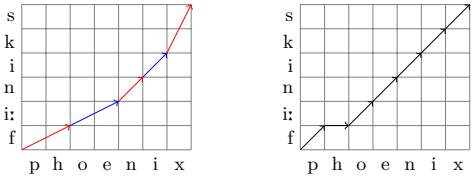

Figure 1: Monotone paths in two-dimensional lattices corresponding to the monotone alignments between $\mathbf{x}$ = phoenix and $\mathbf{y}$ = fi:niks given in Section 1. In the left lattice, we have arbitrarily (but suggestively) colored each step in either red or blue.

Denote by $\mathbb{Z}$ the set of integers, by $\mathbb{N}$ the set of non-negative integers, and by $\mathbb{R}$ the set of real numbers. Consider the two-dimensional lattice $\mathbb{Z}^2$. In $\mathbb{Z}^2$, we call an ordered list of pairs $(\alpha_0, \beta_0) = (0,0), \ldots, (\alpha_k, \beta_k) = (m,n)$ a *path* from $(0,0)$ to (m,n), and we call

[1]The many-to-many alignment algorithm designed in (Jiampojamarn et al., 2007) is an extension of a one-to-one stochastic transducer devised in (Ristad and Yianilos, 1998). Moreover, (Brill and Moore, 2000) learn the weighted edit distance between string pairs where edit operations may encompass arbitrary subsequences of strings, a setting also closely related to our problem of monotone many-to-many alignments.

$(a_i, b_i) := (\alpha_i, \beta_i) - (\alpha_{i-1}, \beta_{i-1})$, $i = 1, \ldots, k$, *steps*. Moreover, we call a path λ in the lattice $\mathbb{Z}^2$ from $(0,0)$ to (m,n) *monotone* if all steps (a,b) are non-negative, i.e. $a \geq 0$, $b \geq 0$, and we call the monotone path λ *S-restricted* for a subset S of $\mathbb{N}^2$ if all steps lie within S, i.e. $(a,b) \in S$.

Note that S-restricted monotone paths define *S-restricted monotone alignments*, between strings $\mathbf{x}$ and $\mathbf{y}$. For example, the two paths in Figure 1 correspond to the two monotone alignments between $\mathbf{x}$ = phoenix and $\mathbf{y}$ = fi:niks illustrated above. Thus, we identify S-restricted monotone paths with S-restricted monotone alignments in the sequel.

Moreover, note that the set and number of S-restricted monotone paths allow simple recursions. To illustrate, the number $T_S(m,n)$ of S-restricted monotone paths from $(0,0)$ to (m,n) satisifies

$$T_S(m,n) = \sum_{(a,b) \in S} T_S(m-a, n-b), \tag{1}$$

with initial condition $T_S(0,0) = 1$ and $T_S(m,n) = 0$ if $m < 0$ or $n < 0$. As will be seen in the next section, under certain assumptions, *optimal* monotone alignments (or, equivalently, paths) can be found via a very similar recursion.

3 An algorithm for S-restricted monotone alignments

Let two strings $\mathbf{x} \in \Sigma_x^*$ and $\mathbf{y} \in \Sigma_y^*$ be given. Moreover, assume that a set S of allowable steps is specified together with a real-valued similarity function $\mathsf{sim} : \Sigma_x^* \times \Sigma_y^* \to \mathbb{R}$ between characters of Σ_x and Σ_y. Finally, assume that the *score* or value of an S-restricted monotone path $\lambda = (\alpha_0, \beta_0), \ldots, (\alpha_k, \beta_k)$ is defined additively linear in the similarity of the substrings of $\mathbf{x}$ and $\mathbf{y}$ corresponding to the steps (a,b) taken, i.e.

$$\text{score}(\lambda) = \sum_{i=1}^{k} \mathsf{sim}(x_{\alpha_{i-1}+1}^{\alpha_i}, y_{\beta_{i-1}+1}^{\beta_i}), \tag{2}$$

where by $x_{\alpha_{i-1}+1}^{\alpha_i}$ we denote the subsequence $x_{\alpha_{i-1}+1} \ldots x_{\alpha_i}$ of $\mathbf{x}$ and analogously for $\mathbf{y}$. Then it is not difficult to see that the problem of finding the path (alignment) with maximal score can be solved efficiently using a very similar (dynamic programming) recursion as in Equation (1), which we outline in Algorithm 1. Moreover, this algorithm is obviously a straightforward generalization of the classical Needleman-Wunsch algorithm, which specifies S as $\{(0,1),(1,0),(1,1)\}$.

Note, too, that in Algorithm 1 we include two additional quantities, not present in the original sequence alignment approach, namely, firstly, the 'quality' q of a step (a,b), weighted by a factor $\gamma \in \mathbb{R}$. This quantity may be of practical importance in many situations. For example, if we specify sim as log-probability (see below), then Algorithm 1 has a 'built-in' tendency to substitute 'smaller', individually more likely steps (a,b) by larger, less likely steps because in the latter case fewer negative numbers are added; if sim assigns strictly positive values, this relationship is reversed. We can counteract these biases by factoring in the *per se* quality of a given step. Also note that if q is added linearly, as we have specified, then the dynamic programming recursion is not violated.

Secondly, we specify a function $\mathsf{L} : \left(\Sigma_x^* \times \Sigma_y^*\right) \times \mathsf{colors} \to \mathbb{R}$, where colors is a finite set of 'colors', that encodes the following idea. Assume that each step $(a,b) \in S$ appears in

C, $C \in \mathbb{N}$, different 'colors', or states. Then, when taking step (a, b) with color $c \in \mathsf{colors}$ (which we denote by the symbol $(a, b)^c$ in Algorithm 1), we assess the 'goodness' of this decision by the 'likelihood' L that the current subsequences of $\mathbf{x}$ and $\mathbf{y}$ selected by the step (a, b) 'belong to'/'are of' color (or state) c. As will be seen below, this allows to very conveniently identify (or postulate) 'latent classes' for character subsequences, while increasing the algorithm's running time only by a constant factor.

To summarize our generalizations over the traditional sequence alignment approach, (i) we allow arbitrary non-negative steps S corresponding to S-restricted monotone alignments, (ii) we include a goodness measure q that evaluates the 'quality' of a given step $(a, b) \in S$ taken, and (iii) we color each step in C different colors and assess the goodness of color c for the subsequences of $\mathbf{x}$ and $\mathbf{y}$ selected by the current step (a, b) as the 'likelihood' L that these subsequences are of color c. Finally, we define the score of a monotone path as an additive linear combination of all three components discussed so that an efficient dynamic programming recursion applies. Note that the algorithm's running time is $O(C|S|mn)$ and is thus linear in the number of colors, the size of S, and the string lengths m and n.[2]

Algorithm 1 Generalized Needleman-Wunsch (GNW)

procedure GNW($x_1 \dots x_m, y_1 \dots y_n$; S, $\mathsf{sim}, \mathsf{q}, \mathsf{L}$)
  $M_{ij} \leftarrow -\infty$ for all $(i, j) \in \mathbb{Z}^2$ such that $i < 0$ or $j < 0$
  $M_{00} \leftarrow 0$
  for $i = 0 \dots m$ **do**
    for $j = 0 \dots n$ **do**
      if $(i, j) \neq (0, 0)$ **then**
        $M_{ij} \leftarrow \max_{(a,b)^c \in S} \{M_{i-a,j-b} + \mathsf{sim}(x^i_{i-a+1}, y^j_{j-b+1}) + \gamma \mathsf{q}(a, b) + \chi \mathsf{L}\big((x^i_{i-a+1}, y^j_{j-b+1}), c\big)\}$
      end if
    end for
  end for
  return M_{mn} ▷ M_{mn} holds value of path with maximal score
end procedure

Algorithm 2 (Hard) EM Training

procedure EM($\{(\mathbf{x}_i, \mathbf{y}_i) \mid i = 1, \dots, N\}$; S, T, $\hat{\mathsf{sim}}_0, \hat{\mathsf{q}}_0, \hat{\mathsf{L}}_0$)
  $t \leftarrow 0$
  while $t < T$ **do**
    for $i = 1 \dots N$ **do**
      $(\mathbf{x}^a_i, \mathbf{y}^a_i) \leftarrow$ GNW($\mathbf{x}_i, \mathbf{y}_i; S, \hat{\mathsf{sim}}_t, \hat{\mathsf{q}}_t, \hat{\mathsf{L}}_t$)
        ▷ $(\mathbf{x}^a_i, \mathbf{y}^a_i)$ denotes the alignment between $\mathbf{x}_i$ and $\mathbf{y}_i$
    end for
    $\hat{\mathsf{sim}}_{t+1}, \hat{\mathsf{q}}_{t+1}, \hat{\mathsf{L}}_{t+1} \leftarrow f(\{\mathbf{x}^a_i, \mathbf{y}^a_i \mid i = 1, \dots, N\})$
      ▷ The function f extracts (count) updates from the aligned data
    $t \leftarrow t + 1$
  end while
end procedure

[2] But also note the dependence of the running time on the definition of sim, q and L.

As to the similarity measure sim employed in Algorithm 1, a popular choice is to specify it as the (logarithm of the) joint probability of the pair $(\mathbf{u}, \mathbf{v}) \in \Sigma_x^* \times \Sigma_y^*$, but a multitude of alternatives is conceivable here such as the χ^2 similarity, pointwise mutual information, etc. (see for instance the overview in (Hoang et al., 2009)). Also note that $\mathsf{sim}(\mathbf{u}, \mathbf{v})$ is usually initially unknown but can be iteratively estimated via application of Algorithm 1 and count estimates in an EM-like fashion (cf. (Dempster et al., 1977)), see Algorithm 2.[3] As concerns q and L, we can likewise estimate them iteratively from data, specifying their abstract forms via any well-defined (goodness) measures. The associated coefficients γ and χ can be optimized on a development set or set exogenously.

4 Exhaustive enumeration and alignments

In the last section, we have specified a polynomial time algorithm for solving the monotonic S-restricted string alignment problem, under the following restriction; namely, we defined the score of an alignment additively linear in the similarities of the involved subsequences. This, however, entails an independence assumption between successive aligned substrings that oftentimes does not seem justified in linguistic applications. If, on the contrary, we specified the score, $\mathrm{score}(\lambda)$, of an alignment λ between strings $\mathbf{x}$ and $\mathbf{y}$ as e.g.

$$\mathrm{score}(\lambda) = \sum_{i=1}^{k} \log \Pr\big((x_{\alpha_{i-1}+1}^{\alpha_i}, y_{\beta_{i-1}+1}^{\beta_i}) \,|\, (x_{\alpha_{i-2}+1}^{\alpha_{i-1}}, y_{\beta_{i-2}+1}^{\beta_{i-1}})\big)$$

(using joint probability as similarity measure) — this would correspond to a 'bigram scoring model' — then Algorithm 1 would not apply.

To address this issue, we suggest *exhaustive enumeration* as a possibly noteworthy alternative — enumerate *all* S-restricted monotone alignments between strings $\mathbf{x}$ and $\mathbf{y}$, score each of them individually, taking the one with maximal score. This brute-force approach is, despite its simplicity, the most general approach conceivable and works under all specifications of scoring functions. Its practical applicability relies on the sizes of the search spaces for S-restricted monotone alignments and on the lengths of the strings $\mathbf{x}$ and $\mathbf{y}$ involved.

We note the following here. By Equation (1), for the choice $S = \{(1,1),(1,2),(1,3),(1,4),(2,1)\}$, a seemingly reasonable specification in the context of G2P (see next section), the number $T_S(n,n)$ of S-restricted monotone alignments is given as (for explicit formulae for specific S, cf. (Eger, 2012b))

$$1, 1, 3, 7, 16, 39, 95, 233, 572, 1406, 3479, 8647$$

for $n = 1, 2, \ldots, 12$ and e.g. $T_S(15,15) = 134,913$. Moreover, for the distribution of letter string and phoneme string lengths we estimate Poisson distributions (cf. (Wimmer et al., 1994)) with parameters $\mu \in \mathbb{R}$ as listed in Table 4 for the German Celex (Baayen et al., 1996), French Brulex (Content et al., 1990) and English Celex datasets, as used in Section 7. As the table and the above numbers show, there are on average only a few hundred or few thousand possible monotone many-to-many alignments between grapheme and phoneme string pairs, for which exhaustive enumeration appears, thus, quite feasible; moreover, given enough data, it usually does not harm much to exclude a few string pairs, for which alignment numbers are too large.

[3] The variant of EM that we describe is sometimes called *hard* EM while e.g. (Jiampojamarn et al., 2007) present a soft EM version; but see the discussion in (Samdani et al., 2012).

Dataset	μ_G	μ_P	$P_{[G>15]}$	$P_{[P>15]}$
German-Celex	9.98	8.67	4.80%	1.62%
French-Brulex	8.49	6.71	1.36%	0.15%
English-Celex	8.21	7.39	1.03%	0.40%

Table 1: Avg. grapheme and phoneme string lengths in resp. data set, and probabilities that lengths exceed 15.

5 Choice of S

Choice of the set of steps S is a question of *model selection*, cf. (Zucchini, 2000). Several approaches are conceivable here. First, for a given domain of application one might specify a possibly 'large' set of steps Ω capturing a preferably comprehensive class of alignment phenomena in the domain. This may not be the best option because it may provide Algorithm 1 with too many 'degrees of freedom', allowing it to settle in unfavorable local optima, and thus may lead to suboptimal alignments (we find appropriate step restriction to have dramatic effects on alignment quality, which we investigate more thoroughly in subsequent research). A better, but potentially very costly, alternative is to exhaustively enumerate all possible subsets S of Ω, apply Algorithm 1 and/or Algorithm 2, and evaluate the quality of the resulting alignments with any choice of suitable measures such as alignment entropy (cf. (Pervouchine et al., 2009)), average log-likelihood, Akaike's information criterion (Akaike, 1974) or the like. Another possibility would be to use a comprehensive Ω, but to penalize unlikely steps, which could be achieved by setting γ in Algorithm 1 to a 'large' real number and then, in subsequent runs, employ the remaining steps $S \subseteq \Omega$; we outline this approach in Section 7.

Sometimes, specific knowledge about a particular domain of application may be helpful, too. For example, in the field of G2P, we would expect most associations in alignments to be of the type M-to-1, i.e. one or several graphemes encode a single phoneme. This is because it seems reasonable to assume that the number of phonetic units used in language communities typically exceeds the number of units in alphabetic writing systems — 26 in the case of the Latin alphabet — so that one or several letters must be employed to represent a single phoneme. There may be 1-to-N or even M-to-N relationships but we would consider these exceptions. In the current work, we choose $S = \{(1,1),(2,1),(3,1),(4,1),(1,2)\}$ for G2P data sets, and for the morphology data sets we either adopt from (Eger, 2012b) or use a comprehensive Ω with 'largest' step $(2,2)$.

6 Decoding

Decoding is the process of generating $\hat{\mathbf{y}} \in \Sigma_y^*$ given $\mathbf{x} \in \Sigma_x^*$. Below, we explain how we perform this process, within the S-restricted monotone many-to-many alignment framework.

6.1 Training a string transduction model

We first generate monotone many-to-many alignments between string pairs with one of the procedures outlined in Sections 3 and 4. Then, we train a linear chain conditional random field (CRF; see (Lafferty et al., 2001)) as a graphical model for string transduction on the aligned data. The choice of CRFs is arbitrary; any transduction procedure tr would do, but we decide for CRFs because they generally have good generalization properties. In all cases, we use window sizes of three or four to predict $\mathbf{y}$ string elements from $\mathbf{x}$ string elements.

6.2 Segmentation

Our overall decoding procedure is as follows. Given an input string $\mathbf{x}$, we exhaustively generate all possible segmentations of $\mathbf{x}$, feed the segmented strings to the CRF for transduction and evaluate each individual resulting sequence of 'graphones' with an n-gram model learned on the aligned data, taking the $\mathbf{y}$ string corresponding to the graphone sequence with maximal probability as the most likely transduced string for $\mathbf{x}$. We illustrate in Algorithm 3. As to the size of the search space that this procedure entails, any segmentation

Algorithm 3 Decoding

procedure decode($\mathbf{x} = x_1 \dots x_m$; $\hat{k}$, α, β, tr)
  $Z \leftarrow \emptyset$
  for $s \in \mathscr{C}(m, \hat{k}, \alpha, \beta)$ **do** ▷ $\mathscr{C}(m, \hat{k}, \alpha, \beta)$: the set of all integer compositions of m with $\hat{k}$ parts, each between α and β
    $\hat{\mathbf{y}} \leftarrow \mathsf{tr}(s)$
    $z_{\hat{\mathbf{y}}} \leftarrow \text{ngramScore}(\mathbf{x}, \hat{\mathbf{y}})$
    $Z \leftarrow Z \cup \{z_{\hat{\mathbf{y}}}\}$
  end for
  $z_{\hat{\mathbf{y}}^*} \leftarrow \max_{z_{\hat{\mathbf{y}}}} Z$
  return $\hat{\mathbf{y}}^*$
end procedure

of a string $\mathbf{x}$ of length m with k parts uniquely corresponds to an *integer composition* (a way of writing m as a sum of non-negative integers) of the integer m with k parts, as in,

		ph		oe		n		i		x
7	=	2	+	2	+	1	+	1	+	1

It is a simple exercise to show that there are $\binom{m-1}{k-1}$ integer compositions of m with k parts, where by $\binom{m}{k}$ we denote the respective binomial coefficient. Furthermore, if we put restrictions on the maximal size of parts — e.g. in G2P a reasonable upper bound l on the size of parts would probably be 4 — we have that there are $\binom{k}{m-k}_l$ integer compositions of m with k parts, each between $\alpha = 1$ and $\beta = l$, where by $\binom{k}{m}_{l+1}$ we denote the respective *polynomial coefficient* (Comtet, 1974). To avoid having to enumerate segmentations for all possible numbers k of segment parts of a given input string $\mathbf{x}$ of length m — these would range between 1 and m, entailing $\sum_{k=1}^{m} \binom{m-1}{k-1} = 2^{m-1}$ possible segmentations in total in the case without upper bound[4] — we additionally train a 'number of parts' prediction model with which to estimate k as $\hat{k}$; we call this in short *predictor model*.

To illustrate the number of possible segmentations with a concrete example, if $\mathbf{x}$ has length $m = 15$, a rather large string size given the values in Table 4, there are

$$2472, 2598, 1902, 990, 364, 91, 14, 1$$

possible segmentations of $\mathbf{x}$ with $k = 8, 9, 10, 11, 12, 13, 14, 15$ parts, each between 1 and 4.

For the sake of completeness, we note that our above discussion presumed that there are no 'empty' parts in integer compositions, that is, that all parts in the integer composition

[4] In the case of upper bounds, (Malandro, 2012) provides asymptotics for the number of restricted integer compositions, which are beyond the scope of the present work, however.

2PKE. abbreche**t**, entgegentrete**t**, zuzieh**t**
z. ab**zu**brechen, entgegen**zu**treten, zu**zu**ziehen
rP. redet, **reibt**, tr**eibt**, verbind**et**
pA. **ge**redet, **ge**r**ieben**, **ge**tr**ieben**, verb**u**nd**en**

Table 2: String pairs in morphology data sets 2PKE and rP (omitting 2PIE and 13SIA for space reasons) discussed by (Dreyer et al., 2008). Changes from one form to the other are in bold (information not given in training). Adapted from (Dreyer et al., 2008).

are integers between 1 and the upper bound l. When converting graphemes to phonemes, we find it unlikely that a sound would be uttered without there being a corresponding letter that gives rise to this sound,[5] i.e. our assumption seems justified. In the general monotone alignment case, however, the zero case would have to be included, e.g. when converting phonemes to graphemes, or in the morphology data sets discussed below, where e.g. segmentations as in

$$\begin{array}{ccccccccccccc} & & \emptyset & & \mathrm{m} & & \mathrm{a} & & \mathrm{c} & & \mathrm{h} & & \mathrm{t} \\ 5 & = & 0 & + & 1 & + & 1 & + & 1 & + & 1 & + & 1 \end{array}$$

seem justified to convert German third person verb form *macht* into participle form *gemacht*. Analogously as above, we find that there are $\binom{k}{m}_{l+1}$ integer compositions of m with k parts, each between 0 and l. To illustrate again, when $m = 15$, there are $37080, 142749, 831204, 2268332, \ldots$ possible segmentations of $\mathbf{x}$ with $k = 8, 9, 10, 11, \ldots$ parts, each between 0 and 4. Obviously, these numbers are much larger than those where all parts are ≥ 1, which is problematic not only from the point of view of computing resources but may also affect accuracy results because more alternatives are provided from which to select. Luckily, as illustrated below, it should usually be possible to specify modeling choices where zero parts do not occur.

7 Experiments

We conduct our experiments on three G2P data sets, the German Celex (G-Celex) and French Brulex data set (F-Brulex) taken from the Pascal challenge (van den Bosch et al., 2006), and the English Celex dataset (E-Celex); and on the four German morphology data sets discussed in (Dreyer et al., 2008), which we refer to, in accordance with the named authors, as rP, 2PKE, 13SIA and 2PIE, respectively. Both for the G2P and the morphology data, we hold monotonicity, by and large, a legitimate assumption so that our approach would appear justified. As to the morphology data sets, we illustrate in Table 7 a few string pair relationships that they contain, as indicated by (Dreyer et al., 2008).

7.1 Alignments

We generate alignments for our data sets using Algorithms 1 and 2 and, as a comparison, we implement an exhaustive search bigram scoring model as indicated in Section 4 in an EM-like fashion similar as in Algorithm 2, employing the CMU SLM toolkit (Clarkson and Rosenfeld, 1997) with Witten-Bell smoothing as n-gram model. For Algorithm 1, which we also refer to as unigram model in the following, we choose steps S as shown in Table

[5] As an exception might be considered e.g. extra terminal vowel sounds like in Italian *sport*, pronounced as s p o r t ə. As pointed out by a reviewer, other such exceptions might include short vowels in Arabic or Hebrew script that are generally not graphemically represented.

E-Celex	$\{(1,1),(2,1),(3,1),(4,1),(1,2)\}$
rP	$\{(0,2),(1,1),(1,2),(2,1),(2,2)\}$
2PKE	$\{(0,2),(1,1),(2,1),(2,2)\}$
13SIA	$\{(1,1),(1,2),(2,1),(2,2)\}$
2PIE	$\{(1,1),(1,2)\}$

Table 3: Data set and choice of S. For all three G2P data sets, we select the same S, exemplarily shown for E-Celex. The choice of S for rP and 2PKE is taken from (Eger, 2012b). For 13SIA and 2PIE we use comprehensive Ω's with largest step $(2,2)$ but the algorithm ends up using just the outlined set of steps.

	Perplexity	$H(L\,\vert\,P)$
2PKE-Uni	7.002 ± 0.04	$\mathbf{0.094 \pm 0.001}$
2PKE-Bi	$\mathbf{6.865 \pm 0.02}$	0.141 ± 0.003
rP-Uni	9.848 ± 0.09	$\mathbf{0.092 \pm 0.003}$
rP-Bi	$\mathbf{9.796 \pm 0.05}$	0.107 ± 0.006
Brulex-Uni	22.488 ± 0.35	$\mathbf{0.706 \pm 0.002}$
Brulex-Bi	$\mathbf{22.215 \pm 0.21}$	0.725 ± 0.003

Table 4: Conditional entropy vs. n-gram perplexity ($n = 2$) of alignments for different data sets. In bold: Statistically best results. $K = 300$ throughout.

3. As similarity measure sim, we use log prob with Good-Turing smoothing and for q we likewise use log prob; we outline the choice of L below. Initially, we set γ and χ to zero. As an alignment quality measure we consider conditional entropy $H(L\,|\,P)$ (or $H(P\,|\,L)$) as suggested by (Pervouchine et al., 2009). Conditional entropy measures the average uncertainty of a (grapheme) substring L given a (phoneme) substring P; apparently, the smaller $H(L\,|\,P)$ the better the alignment because it produces more consistent associations.

In the following, all results are averages over several runs, 5 in the case of the unigram model and 2 in the case of the bigram model. Both for the bigram model and the unigram model, we select K, where $K \in \{50, 100, 300, 500\}$, training samples randomly in each EM iteration for alignment and from which to update probability estimates.

In Figure 2, we show learning curves over EM iterations in the case of the unigram and bigram models, and over training set sizes. We see that performance, as measured by conditional entropy, increases over iterations both for the bigram model and the unigram model (in Figure 2), but apparently alignment quality decreases again when too large training set sizes K are considered in the case of the bigram model (omitted for space reasons); similar outcomes have been observed when similarity measures other than log prob are employed in Algorithm 1 for the unigram model, e.g. the χ^2 similarity measure (cf. (Eger, 2012b)). To explain this, we hypothesize that the bigram model (and likewise for specific similarity measures) is more susceptible to overfitting when it is trained on too large training sets so that it is more reluctant to escape 'non-optimal' local minima. We also see that, apparently, the unigram model performs frequently better than the bigram model.

The latter results may be partly misleading, however. Conditional entropy, the way (Pervouchine et al., 2009) have specified it, is a 'unigram' assessment model itself and may therefore be incapable of accounting for certain contextual phenomena. For example, in the 2PKE and rP data, we find alignment possibilities of the following types,

–	g	e	b	t		g	e	–	b	t
ge	g	e	b	en		g	e	ge	b	en

where we list the linguistically 'correct', due to the prefixal character of *ge* in German, alignment on the left and the 'incorrect' alignment on the right. By its specification, Algorithm 1 *must* assign both these alignments the same score and can hence not distinguish between them; *the same holds true for the conditional entropy measure*. To address this issue, we evaluate alignments by a second method as follows. From the aligned data, we extract a random sample of size 1000 and train an n-gram graphone model (that can account for 'positional associations') on the residual, assessing its perplexity on the held-out set of size 1000. Results are shown in Table 4. We see that, in agreement with our visual impression at least for the morphology data, the alignments produced by the bigram model seem to be slightly more consistent in that they reduce perplexity of the n-gram graphone model, whereas conditional entropy proclaims the opposite ranking.

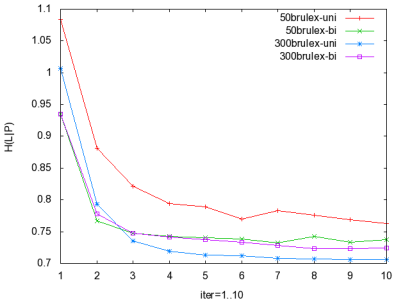

Figure 2: Learning curves over iterations for F-Brulex data, $K = 50$ and $K = 300$, for unigram and bigram models.

7.1.1 Quality q of steps

In Table 5 we report results when experimenting with the coefficient γ of the quality of steps measure q. Overall, we do not find that increasing γ would generally lead to a performance increase, as measured by e.g. $H(L\,|\,P)$. On the contrary, when choosing as set of steps a comprehensive Ω as in Table 5, where we choose $\Omega = \{(a,b)\,|\,a \leq 4, b \leq 4\}\backslash\{(0,0)\}$, for $\gamma = 0$, we find values of 0.278, 0.546, 0.662 for $H(L\,|\,P)$ for G-Celex, F-Brulex and E-Celex, respectively, while corresponding values for $\gamma = 10$ are 0.351, 0.833, 1.401. Contrarily, $H(P\,|\,L)$, the putatively more indicative measure for transduction from $\mathbf{x}$ to $\mathbf{y}$, has 0.499, 0.417, 0.598 for $\gamma = 0$ and 0.378, 0.401, 1.113 for $\gamma = 10$, so that, except for the E-Celex data, $\gamma = 10$ apparently leads to improved $H(P\,|\,L)$ values in this situation, while $\gamma = 0$ seems to lead to better $H(L\,|\,P)$ values.

In any case, from a model complexity perspective,[6] increasing γ may certainly be beneficial. For example, Table 5 shows that with $\gamma = 0$, Algorithm 1 will select up to 15 different steps for the given choice Ω, most of which seem linguistically questionable. On the contrary, with a large γ, Algorithm 1 employs only four resp. five different steps for the G2P data; most

[6]Taking into model complexity is e.g. in accordance with Occam's razor or Akaike's information criterion.

importantly, among these are $(1,1)$, $(2,1)$ and $(3,1)$, all of which are in accordance with linguistic reasoning as e.g. outlined in Section 5. Thus, we can think of q as a 'regularization term' that prevents the algorithm from 'overfitting' the data.

	$(1,1)$	$(2,1)$	$(3,1)$	$(4,1)$	$(1,2)$	$(1,0)$	$(2,3)$	$(3,2)$	$(3,3)$	$(4,2)$	$(4,3)$	$(4,4)$	$(2,2)$	$(0,1)$	$(1,3)$
G-Celex	86.50	11.61	1.77	-	0.10	-	-	-	-	-	-	-	-	-	-
	86.14	8.17	1.63	0.02	0.00	2.56	0.10	0.04	0.01	0.09	0.91	0.28	-	-	-
F-Brulex	78.85	15.08	5.85	-	-	-	0.20	-	-	-	-	-	-	-	-
	75.64	13.80	2.52	0.36	0.07	5.07	0.29	0.10	0.02	0.38	1.01	0.68	-	-	-
E-Celex	88.87	6.58	3.05	-	-	-	-	-	-	-	-	-	-	1.29	0.18
	75.54	8.45	0.75	0.04	1.48	4.57	0.41	0.03	0.16	0.44	2.03	3.03	0.00	2.87	0.12

Table 5: Steps and their frequency masses in percent for different data sets for $\gamma = 10$ (top rows) and $\gamma = 0$ (bottom rows), averaged over two runs. We include only steps whose average occurrence exceeds 10.

7.1.2 Colors

We briefly discuss here a possibility to detect latent classes via the concept of colored paths. Assume that a corpus of colored alignments is available and let each color be represented by the contexts (graphones to the left and right) of its members; moreover, define the 'likelihood' L that the pair $p_{x,y} := (x_{\alpha_{i-1}+1}^{\alpha_i}, y_{\beta_{i-1}+1}^{\beta_i})$ is of color c as the (document) similarity (in an information retrieval sense) of $p_{x,y}$'s contexts with color c, which we can e.g. implement via the cosine similarity of the context vectors associated with $p_{x,y}$ and c. For number of colors $C = 2$, we then find, under this specification, the following kinds of alignments when running Algorithms 1 and 2 with $\gamma = 0$ and $\chi = 1$,

a	nn	**u**	al		ph	**o**	n	**e**	me
&	n	**jU**	l		f	**@U**	n	**i**	m

where we use bold font to distinguish the two color classes, and use original E-Celex notation for phonemic characters. It is clear that the algorithm has detected some kind of consonant/vowel distinction on a phonemic level here. We find similar kinds of latent classes for the other G2P data sets, and for the morphology data, the algorithm learns (less interestingly) to detect word endings and starts, under this specification.

7.2 Transductions

We report results of experiments on transducing $\mathbf{x}$ strings to $\mathbf{y}$ strings for the G2P data and the morphology data sets. We exclude E-Celex because training the CRF with our parametrizations (e.g. all features in window size of four) did regularly not terminate, due to the large size of the data set ($> 60{,}000$ string pairs). Likewise for computing resources reasons,[7] we do not use ten-fold cross-validation but, as in (Jiampojamarn et al., 2008), train on the first 9 folds given by the Pascal challenge, testing on the last. Moreover, for the G2P data, we use an ϵ-scattering model with steps $S = \{(1,0),(1,1)\}$ as a predictor model from which to infer the number of parts $\hat{k}$ for decoding and then apply Algorithm 3.[8] For alignments, we use in all cases Algorithms 1 and 2 with $\gamma = 0$ and $\chi = 0$. As reference for the G2P data, we give word accuracy rates as announced by (Bisani and Ney,

[7] E.g. a single run of the CRF on the G-Celex data takes longer than 24 hours on a standard PC.

[8] We train the ϵ-scattering model on data where all multi-character phonemes such as *ks* are merged to a single character, as obtained from the alignments as given by Algorithms 1 and 2.

	CRF-3	CRF-4	CRF-4*	DSE-F	DSE-FL	Mos3	Mos15	M-M+HMM	BN	MeR+A*
F-Brulex		93.7	**94.6**					90.9	93.7	86.7
G-Celex		91.1	**92.6**					89.8		90.2
2PKE	79.8	80.9		74.7	**87.4**	67.1	82.8			
rP	74.1	77.2		69.9	**84.9**	67.6	70.8			
13SIA	85.6	86.5		82.8	**87.5**	73.9	85.3			
2PIE	**94.6**	94.2		88.7	93.4	92.0	94.0			

Table 6: Data sets and word accuracy rates in percent. **DSE-F**: (Dreyer et al., 2008) using 'pure' alignments and features. **DSE-FL**: (Dreyer et al., 2008) using alignments, features and latent classes. **Mos3, Mos15**: Moses system with window sizes of 3 and 15, resp., as reported by (Dreyer et al., 2008). **M-M+HMM**: Many-to-many aligner with HMM and instance-based segmenter for decoding as reported by (Jiampojamarn et al., 2007). **BN**: (Bisani and Ney, 2008) using a machine translation motivated approach to many-to-many alignments. **MeR+A***: Results of Moses system on G2P data as reported by (Rama et al., 2009). **CRF-3** Our approach with window size of 3 and 3-gram scoring model (see Algorithm 3). **CRF-4**: Our approach with window size of 4 and 3-gram scoring model. **CRF-4***: Our approach with window size of 4 and 4-gram scoring model and 2-best lists (i.e. in Algorithm 3, obtain $\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_2$ as the two most probable transductions of s). In bold: Best results (no statistical tests). Underlined: best results using 'pure' alignments.

2008), (Jiampojamarn et al., 2007), and (Rama et al., 2009), who gives the Moses 'baseline' (Koehn et al., 2007).

For the morphology data we use exactly the same training/test data splits as in (Dreyer et al., 2008). Moreover, because (Dreyer et al., 2008) report all results in terms of window sizes of 3, we do likewise for this data. For decoding we do not use a (complex) predictor model here but rely on simple statistics; e.g. we find that for the class 13SIA, k is always in $\{m-2, m-1, m\}$, where m is the length of $\mathbf{x}$, so we apply Algorithm 3 three times and select the best scoring $\hat{\mathbf{y}}$ string. To avoid zeros in the decoding process (see discussion in Section 6.2), we replace the $(0, 2)$ steps used in the rP and 2PKE data sets by a step $(1, 3)$.

Results are shown in Table 6. For the G2P data, our approach always outperforms the best reported results for pipeline approaches (see below), while we are significantly below the results reported by (Dreyer et al., 2008) for the morphology data in two out of four cases. Contrarily, when 'pure' alignments are taken into consideration — (Dreyer et al., 2008) learn very complex latent classes with which to enrich alignments — our results are clearly better throughout. In almost all cases, we significantly beat the Moses 'baseline'.

8 Discussion

We believe our alignment procedure to be superior to the one presented in (Jiampojamarn et al., 2007) (and likewise for the 'machine translation motivated' approach outlined by (Bisani and Ney, 2008)) from a number of perspectives. First, it is more flexible and general in that it allows the specification of arbitrary non-negative steps S and arbitrary similarity measures sim. Moreover, as we have shown, our approach can very easily and conveniently be adapted to incorporate step quality measures, which may turn out to be very useful in detecting the 'right' choice of S (i.e. as a 'regularization term'); and our approach can also easily be generalized to incorporate the modeling of latent classes as e.g. done in (Dreyer et al., 2008), within a polynomial running time framework; further generalizations such as semi-ring specifications (Mohri, 2002) are obvious but not discussed in the current

work (cf. (Eger, 2012b)). Secondly, our algorithm appears very simple and intuitive, while being computationally equivalently tractable and making the same sorts of independence assumptions as in (Jiampojamarn et al., 2007).

As regards decoding, (Jiampojamarn et al., 2007) use a grapheme segmentation module where each grapheme letter can form a chunk with its neighbor or stand alone, a decision that is based on local context and instance-based learning. We hold this approach to be insufficient because (besides the obvious drawback that, as we have shown, larger chunks than two seem appropriate for G2P) it unecessarily restricts the search space for grapheme segmentation; once a decision is made to join two letters, it cannot be reversed and alternative segmentations are not considered. The same holds true for the phrasal decoder approach outlined in (Jiampojamarn et al., 2008), the critique of which is already uttered in (Dreyer et al., 2008), namely, that the input string is segmented into substrings which are transduced *independently* of each other, ignoring context. Contrarily, for decoding $\mathbf{x}$, we compute *all* possible segmentations of $\mathbf{x}$ and score them (in conjunction with the transduced $\hat{\mathbf{y}}$ strings) with higher order n-gram models, which is clearly superior to the named approaches because it takes both context into account and does not restrict search space. Moreover, *given an adequate predictor model*, we found that enumerating all possible restricted integer compositions is so fast that no further investigation of restricting search space is necessary.[9]

While we thus believe our individual components for alignment and decoding to be superior to the mentioned approaches, our modeling of string transductions adheres to a pipeline approach — in (Jiampojamarn et al., 2008)'s words — that, as they suggest, is inferior to a unified framework, as they present it. All our components can be integrated within such a framework, which is scope for future research. In contrast with (Dreyer et al., 2008), we believe our alignments (*per se*) to be more adequate (they use one-to-one and one-to-zero alignments), which the performance measures corroborate, while their idea to enrich alignments with a multitude of latent classes (more complex than representable in our framework) obviously outperforms our method on certain data sets such as those encountered in morphology, where e.g. latent *word* classes may be of great importance.

Conclusion

We have presented a simple and general framework for generating monotone many-to-many alignments that competes with (Jiampojamarn et al., 2007)'s alignment procedure. Moreover, we have discussed crucial independence assumptions and, thus, limitations of this algorithm and shown that exhaustive enumeration (among other methods) can overcome these problems — in particular, due to the relatively small search space — in the field of monotone alignments. Additionally, we have discussed problems of standard alignment quality measures such as conditional entropy and have suggested an alternative decoding procedure for string transduction within the monotone many-to-many alignment framework that addresses the limitations of the procedures suggested by (Jiampojamarn et al., 2007) and (Jiampojamarn et al., 2008). In future work, we intend to explore more extensively, in particular, the effects of appropriate step restriction and regularization upon alignment quality.

[9] We used the algorithm presented in (Updyke, 2010).

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

Baayen, H., Piepenbrock, R., and Gulikers, L. (1996). *The CELEX2 lexical database*. Linguistic Data Consortium, Philadelphia.

Baldwin, T. and Tanaka, H. (1999). Automated Japanese grapheme-phoneme alignment. In *Proc. of the International Conference on Cognitive Science*, pages 349–354.

Bisani, M. and Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434–451.

Black, A., Lenzo, K., and Pagel, V. (1998). Issues in building general letter to sound rules. In *The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis. ISCA*.

Brill, E. and Moore, R. (2000). An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics.

Brown, P., Cocke, J., , Pietra, S. D., Pietra, V. D., Jelinek, F., Lafferty, J., Mercer, R., and Roossin, P. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Clarkson, P. and Rosenfeld, R. (1997). Statistical language modeling using the CMU-Cambridge toolkit. In *Proceedings ESCA Eurospeech*.

Comtet, L. (1974). *Advanced Combinatorics*. D. Reidel Publishing Company.

Content, A., Mousty, P., and Radeau, M. (1990). Une base de données lexicales informatisée pour le francais écrit et parlé. In *L'Année Psychologique*, pages 551–566.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Dreyer, M., Smith, J., and Eisner, J. (2008). Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1080–1089, Honolulu, Hawaii.

Eger, S. (2012a). On S-restricted f-weighted integer compositions and extended binomial coefficients. Submitted.

Eger, S. (2012b). Sequence alignment with arbitrary steps and further generalizations, with applications to alignments in linguistics. Submitted.

Galescu, L. and Allen, J. (2001). Bi-directional conversion between graphemes and phonemes using a joint n-gram model. In *Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Perthshire, Scotland.

Heubach, S. and Mansour, T. (2004). Compositions of n with parts in a set. *Congressus Numerantium*, 164:127–143.

Hoang, H., Kim, S., and Kan, M.-Y. (2009). A re-examination of lexical association measures. In *MWE '09 Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*.

Jiampojamarn, S., Cherry, C., and Kondrak, G. (2008). Joint processing and discriminative training for letter-to-phoneme conversion. In *46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, pages 905–913.

Jiampojamarn, S. and Kondrak, G. (2010). Letter-phoneme alignment: An exploration. In *The 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 780–788.

Jiampojamarn, S., Kondrak, G., and Sherif, T. (2007). Applying many-to-many alignments and Hidden Markov models to letter-to-phoneme conversion. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2007)*, pages 372–379, Rochester, NY.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic. Association for Computational Linguistics.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289.

Malandro, M. (2012). Asymptotics for restricted integer compositions. Preprint available at `http://arxiv.org/pdf/1108.0337v1`.

Mohri, M. (2002). Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350.

Needleman, S. and Wunsch, C. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453.

Pervouchine, V., Li, H., and Lin, B. (2009). Transliteration alignment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 136–144.

Rama, T., Kumar, A., and Kolachina, S. (2009). Modeling letter to phoneme conversion as a phrase based statistical machine translation problem with minimum error rate training. In *NAACL HLT 2009 Student Research Workshop*, pages 90–95, Colorada, USA.

Ristad, E. and Yianilos, P. (1998). Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–533.

Samdani, R., Chang, M.-W., and Roth, D. (2012). Unified expectation maximization. In *HLT-NAACL*, pages 688–698.

Taylor, P. (2005). Hidden Markov models for grapheme to phoneme conversion. In *Proceedings of the 9th European Conference on Speech Communication and Technology 2005*.

Updyke, J. (2010). A unified approach to algorithms generating unrestricted and restricted integer compositions and integer partitions. *Journal of Mathematical Modelling and Algorithms*, 9(1):53–97.

van den Bosch, A., Chen, S., Daelemans, W., Damper, R., Gustafson, K., Marchand, Y., and Yvon, F. (2006). Pascal letter-to-phoneme conversion challenge. `http://www.pascalnetwork.org/Challenges/PRONALSYL`.

Wimmer, G., Köhler, R., Grotjahn, R., and Altmann, G. (1994). Towards a theory of word length distribution. *Journal of Quantitative Linguistics*, 1:98–106.

Zucchini, W. (2000). An introduction to model selection. *Journal of Mathematical Psychology*, 44:41–61.