Multiple Many-to-Many Sequence Alignment for Combining String-Valued Variables: A G2P Experiment

Steffen Eger
Text Technology Lab
Goethe University Frankfurt am Main
Frankfurt am Main, Germany
steeger@em.uni-frankfurt.de

Abstract

We investigate *multiple many-to-many alignments* as a primary step in integrating supplemental information strings in string transduction. Besides outlining DP based solutions to the multiple alignment problem, we detail an approximation of the problem in terms of multiple sequence segmentations satisfying a coupling constraint. We apply our approach to boosting baseline G2P systems using homogeneous as well as heterogeneous sources of supplemental information.

1 Introduction

String-to-string translation (string transduction) is the problem of converting one string $\mathbf{x}$ over an alphabet Σ into another string $\mathbf{y}$ over a possibly different alphabet Γ. The most prominent applications of string-to-string translation in natural language processing (NLP) are grapheme-to-phoneme conversion, in which $\mathbf{x}$ is a letter-string and $\mathbf{y}$ is a string of phonemes, transliteration (Sherif and Kondrak, 2007), lemmatization (Dreyer et al., 2008), and spelling error correction (Brill and Moore, 2000). The classical learning paradigm in each of these settings is to train a model on pairs of strings $\{(\mathbf{x}, \mathbf{y})\}$ and then to evaluate model performance on test data. Thereby, all state-of-the-art modelings we are aware of (e.g., (Jiampojamarn et al., 2007; Bisani and Ney, 2008; Jiampojamarn et al., 2008; Jiampojamarn et al., 2010; Novak et al., 2012)) proceed by first *aligning* the string pairs $(\mathbf{x}, \mathbf{y})$ in the training data. Also, these modelings acknowledge that alignments may typically be of a rather complex nature in which several $\mathbf{x}$ sequence characters may be matched up with several $\mathbf{y}$ sequence characters; Table 1 illustrates. Once the training data is aligned, since $\mathbf{x}$ and $\mathbf{y}$ sequences are then segmented into equal number of segments, string-to-string translation may be seen as a sequence labeling (tagging) problem in which $\mathbf{x}$ (sub-)sequence characters are observed variables and $\mathbf{y}$ (sub-)sequence characters are hidden states (Jiampojamarn et al., 2007; Jiampojamarn et al., 2010).

ph	oe	n	i	x
f	i	n	ɪ	ks

Table 1: Sample monotone many-to-many alignment between $\mathbf{x} =$ phoenix and $\mathbf{y} =$ fiɪnks.

In this work, we extend the problem of classical string-to-string translation by assuming that, at training time, we have available $(M+2)$-tuples of strings $\{(\mathbf{x}, \hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)}, \mathbf{y})\}$, where $\mathbf{x}$ is the input string, $\hat{\mathbf{y}}^{(m)}$, for $1 \leq m \leq M$, are *supplemental information* strings, and $\mathbf{y}$ is the desired output string; at test time, we wish to predict $\mathbf{y}$ from $(\mathbf{x}, \hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)})$. Generally, we may think of $\hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)}$ as arbitrary strings over arbitrary alphabets $\Sigma^{(m)}$, for $1 \leq m \leq M$. For example, $\mathbf{x}$ might be a letter-string and $\hat{\mathbf{y}}^{(m)}$ might be a transliteration of $\mathbf{x}$ in language L_m (cf. Bhargava and Kondrak (2012)). Alternatively, and this is our model scenario in the current work, $\mathbf{x}$ might be a letter input string and $\hat{\mathbf{y}}^{(m)}$ might be the predicted string of phonemes, given $\mathbf{x}$, produced by an (offline) system T_m. This situation is outlined in Table 3. In the table, we also illustrate a *multiple (monotone) many-to-many alignment* of $(\mathbf{x}, \hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)}, \mathbf{y})$. By this, we mean an alignment where (1) *subsequences* of all $M+2$ strings may be matched up with each other (many-

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 909–919,
Beijing, China, July 26-31, 2015.

to-many alignments), and where (2) the matching up of subsequences obeys monotonicity. Note that such a multiple alignment generalizes classical monotone many-to-many alignments between *pairs* of strings, as shown in Table 1. Furthermore, such an alignment may apparently be quite useful. For instance, while none of the strings $\hat{\mathbf{y}}^{(m)}$ in the table equals the true phonetic transcription $\mathbf{y}$ of $\mathbf{x}$, taking a position-wise majority vote of the multiple alignment of $(\hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)})$ yields $\mathbf{y}$. Moreover, analogously as in the case of pairs of aligned strings, we may perceive the so extended string-to-string translation problem as a sequence labeling task once $(\mathbf{x}, \hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)}, \mathbf{y})$ are multiply aligned, but now, with *additional observed variables* (or *features*), namely, (sub-)sequence characters of each string $\hat{\mathbf{y}}^{(m)}$.

To further motivate our approach, consider the situation of training a new G2P system on the basis of, e.g., Combilex (Richmond et al., 2009). For each letter form in its database, Combilex provides a corresponding phonetic transcription. Now, suppose that, in addition, we can poll an external knowledge source such as Wiktionary for (its) phonetic transcriptions of the respective Combilex letter words as outlined in Table 2. The central question we want to answer is: can we train a system using this *additional* information which performs better than the 'baseline' system that ignores the extra information? Clearly, a system with more information should not perform worse than a system with less information (unless the additional information is highly noisy), but it is a priori not clear at all how the extra information can be included, as Bhargava and Kondrak (2012) note: output predictions may be in distinct alphabets and/or follow different conventions, and simple rule-based conversions may even deteriorate a baseline system's performance. Their solution to the problem is to let the baseline system output its n-best phonetic transcriptions, and then to re-rank these n-best predictions via an SVM reranker trained on the supplemental representations

Input form	Wiktionary	Combilex
neutrino	njuːtɪiːnoʊ	nutrinF
wooded	wʊdɪd	wUd@d
wrench	ɹɛnʧ	rEn<

Table 2: Input letter words, Wiktionary and Combilex transcriptions.

$\mathbf{x}$ = schizo	s	ch	i	z	o
$\hat{\mathbf{y}}^{(1)}$ = skaɪzəʊ	s	k	aɪ	z	əʊ
$\hat{\mathbf{y}}^{(2)}$ = saɪzəʊ	s	-	aɪ	z	əʊ
$\hat{\mathbf{y}}^{(3)}$ = skɪtsə	s	k	ɪ	ts	ə
$\hat{\mathbf{y}}^{(4)}$ = ʃitsəʊ	ʃ	-	i	ts	əʊ
$\hat{\mathbf{y}}^{(5)}$ = skɪtsə	s	k	ɪ	ts	ə
$\mathbf{y}$ = skɪtsəʊ	s	k	ɪ	ts	əʊ

Table 3: Left: Input string $\mathbf{x}$, predictions of 5 systems, and output string $\mathbf{y}$. Right: A multiple many-to-many alignment of $(\mathbf{x}, \hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(5)}, \mathbf{y})$. Skips are marked by a dash ('-').

(see their figure 2). Our approach is much different from this: we character (or substring) align the supplemental information strings with the input letter strings and then sequentially transduce input character substrings as in the standard G2P approach, but where the sequential transducer is aware of the corresponding subsequences of the supplemental information strings.

Our goals in the current work are first, in Section 2, to formally introduce the multiple many-to-many alignment problem, which, to our knowledge, has not yet been formally considered, and to indicate how it can be solved (by standard extensions of well-known DP recursions). Secondly, we outline an 'approximation algorithm', also in Section 2, with much better runtime complexity, to solving the multiple many-to-many alignment problem. This proceeds by optimally *segmenting* individual strings to align *under the global constraint that the number of segments must agree* across strings. Thirdly, we demonstrate experimentally, in Section 5, that multiple many-to-many alignments may be an extremely useful first step in boosting the performance of a G2P model. In particular, we show that by conjoining a base system with additional systems very high performance increases can be achieved. We also investigate the effects of using our introduced approximation algorithm instead of 'exactly' determining alignments. We discuss related work in Section 3, present data and systems in Section 4 and conclude in Section 6.

2 Mult. Many-to-Many Alignm. Models

We now formally define the problem of multiply aligning several strings in a *monotone and many-to-many alignment* manner. For notational convenience, in this section, let the N strings to align be

denoted by $\mathbf{w}_1, \ldots, \mathbf{w}_N$ (rather than $\mathbf{x}, \hat{\mathbf{y}}^{(m)}, \mathbf{y}$, etc.). Let each $\mathbf{w}_n$, for $1 \leq n \leq N$, be an arbitrary string over some alphabet $\Sigma^{(n)}$. Let $\ell_n = |\mathbf{w}_n|$ denote the length of $\mathbf{w}_n$. Moreover, assume that a set $S \subseteq \prod_{n=1}^{N} \{0, \ldots, \ell_n\} \backslash \{\mathbf{0}_N\}$ of *allowable steps* is specified, where $\mathbf{0}_N = (\underbrace{0, \ldots, 0}_{N \text{ times}})$.[1] We interpret the elements of S as follows: if $(s_1, s_2, \ldots, s_N) \in S$, then subsequences of $\mathbf{w}_1$ of length s_1, subsequences of $\mathbf{w}_2$ of length s_2, ..., subsequences of $\mathbf{w}_N$ of length s_N may be matched up with each other. In other words, S defines the types of valid 'many-to-many match-up operations'.[2] While we could drop S from consideration and simply allow every possible matching up of character subsequences, it is convenient to introduce S because algorithmic complexity may then be specified in terms of S, and by choosing particular S, one may retrieve special cases otherwise considered in the literature (see next section).

As indicated, for us, a multiple alignment of $(\mathbf{w}_1, \ldots, \mathbf{w}_N)$ is any scheme

$$\begin{array}{cccc} \mathbf{w}_{1,1} & \mathbf{w}_{1,2} & \cdots & \mathbf{w}_{1,k} \\ \mathbf{w}_{2,1} & \mathbf{w}_{2,2} & \cdots & \mathbf{w}_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_{N,1} & \mathbf{w}_{N,2} & \cdots & \mathbf{w}_{N,k} \end{array}$$

such that $(|\mathbf{w}_{1,i}|, \ldots, |\mathbf{w}_{N,i}|) \in S$, for all $i = 1, \ldots, k$, and such that $\mathbf{w}_n = \mathbf{w}_{n,1} \cdots \mathbf{w}_{n,k}$, for all $1 \leq n \leq N$. Let $A_S = A_S(\mathbf{w}_1, \ldots, \mathbf{w}_N)$ denote the set of all multiple alignments of $(\mathbf{w}_1, \ldots, \mathbf{w}_N)$. For an alignment $a \in A_S$, denote by $\text{score}(a) = f(a)$ the *score* of alignment a under *alignment model* f, where $f : A_S(\mathbf{w}_1, \ldots, \mathbf{w}_N) \to \mathbb{R}$. We now investigate solutions to the *problem of finding the alignment with maximal score* under different choices of alignment models f, i.e., we search to efficiently solve

$$\max_{a \in A_S(\mathbf{w}_1, \ldots, \mathbf{w}_N)} f(a). \tag{1}$$

Unigram alignment model For our first alignment model f, we assume that $f(a)$, for $a \in A_S$, is the score

$$f(a) = \sum_{i=1}^{k} \mathsf{sim}_1(\mathbf{w}_{1,i}, \ldots, \mathbf{w}_{N,i}) \tag{2}$$

for a real-valued similarity function $\mathsf{sim}_1 : \prod_{n=1}^{N} \left(\Sigma^{(n)}\right)^* \to \mathbb{R}$. We call the model f in (2) a *unigram model* because $f(a)$ is the sum of the similarity scores of the matched-up subsequences $(\mathbf{w}_{1,i}, \ldots, \mathbf{w}_{N,i})$, ignoring context. Due to this independence assumption, solving maximization problem in Eq. (1) under specification (2) is straightforward via a dynamic programming (DP) recursion. To do so, define by $M_{S,\mathsf{sim}_1}(i_1, i_2, \ldots, i_N)$ the score of the best alignment, under alignment model $f = \sum \mathsf{sim}_1$ and set of steps S, of $(\mathbf{w}_1(1 : i_1), \ldots, \mathbf{w}_N(1 : i_N))$.[3] Then, $M_{S,\mathsf{sim}_1}(i_1, \ldots, i_N)$ is equal to

$$\max_{(j_1, \ldots, j_N) \in S} M_{S,\mathsf{sim}_1}(i_1 - j_1, \ldots, i_N - j_N) + \mathsf{sim}_1\big(\mathbf{w}(i_1 - j_1 + 1 : i_1), \ldots, \mathbf{w}(i_N - j_N + 1 : j_N)\big). \tag{3}$$

This recurrence directly leads to a DP algorithm, shown in Algorithm 1, for computing the *score* of the best alignment of $(\mathbf{w}_1, \ldots, \mathbf{w}_N)$; the actual alignment can be found by storing pointers to the maximizing steps taken. If similarity evaluations $\mathsf{sim}_1(\mathbf{w}_{1,i}, \ldots, \mathbf{w}_{N,i})$ are thought of as taking constant time, this algorithm's run time is $\mathcal{O}(\prod_{n=1}^{N} \ell_n \cdot |S|)$. When $\ell = \ell_1 = \cdots = \ell_n$ and $|S| = \ell^N - 1$ ('worst case' size of S), then the algorithm's runtime is thus $\mathcal{O}(\ell^{2N})$, which quickly becomes untractable as N, the number of strings to align, increases.

Of course, the unigram alignment model could be generalized to an m-gram alignment model. An m-gram alignment model would exhibit worst-case runtime complexity of $\mathcal{O}(\ell^{(m+1)N})$ under analogous DP recursions as for the unigram model.

Algorithm 1

procedure UNIGRAM-ALIGN($\mathbf{w}_1, \ldots, \mathbf{w}_N$; S, sim_1)
  $M(i_1, \ldots, i_N) \leftarrow -\infty$ for all $(i_1, \ldots, i_N) \in \mathbb{Z}^N$
  $M(\mathbf{0}_N) \leftarrow 0$
  for $i_1 = 0 \ldots \ell_1$ **do**
    for $\cdots$ **do**
      for $i_N = 0 \ldots \ell_N$ **do**
        if $(i_1, \ldots, i_N) \neq \mathbf{0}_N$ **then**
          $M(i_1, \ldots, i_N) \leftarrow$ Eq. (3)
  return $M(\ell_1, \ldots, \ell_N)$

Separable alignment models For our second model class, assume that, for any $a \in$

[1] Here, $\prod$ denotes the Cartesian product of sets.

[2] In the case of two strings, this is sometimes denoted in the manner M-N (e.g., 3-2, 1-0), indicating that M characters of one string may be matched up with N characters of the other string. Analogously, we could write here s_1-s_2-s_3-$\cdots$.

[3] We denote by $\mathbf{x}(a : b)$ the substring $x_a x_{a+1} \cdots x_b$ of the string $x_1 x_2 \cdots x_t$.

$A_S(\mathbf{w}_1, \ldots, \mathbf{w}_N)$, $f(a)$ decomposes into

$$f(a) = \Psi\Big(f_{\mathbf{w}_1}(\mathbf{w}_{1,1} \cdots \mathbf{w}_{1,k}), \ldots, f_{\mathbf{w}_N}(\mathbf{w}_{N,1} \cdots \mathbf{w}_{N,k})\Big) \tag{4}$$

for some models $f_{\mathbf{w}_1}, \ldots, f_{\mathbf{w}_N}$ and where $\Psi : \mathbb{R}^N \rightarrow \mathbb{R}$ is non-decreasing in its arguments (e.g., $\Psi(f_{\mathbf{w}_1}, \ldots, f_{\mathbf{w}_N}) = \sum_{n=1}^{N} f_{\mathbf{w}_n}$). If $f(a)$ decomposes in such a manner, then $f(a)$ is called *separable*.[4] The advantage with separable models is that we can solve the 'subproblems' $f_{\mathbf{w}_1}, \ldots, f_{\mathbf{w}_N}$ independently. Thus, in order to find optimal multiple alignments of $(\mathbf{w}_1, \ldots, \mathbf{w}_N)$ under such a specification, we would only have to find the best *segmentations* of sequences $\mathbf{w}_n$ under models $f_{\mathbf{w}_n}$, for $1 \leq n \leq N$, subject to the constraint that the segmentations must agree in their *number* of segments (the *coupling variable*). Let $S_{\mathbf{w}_n} \subseteq \{0, 1, \ldots, \ell_n\}$ denote the constraints on *segment lengths*, similar to the interpretation of steps in S. If $f_{\mathbf{w}_n}$ is a unigram segmentation model then the problem of finding the best segmentation of $\mathbf{w}_n$ with exactly j segments can be solved in time $\mathcal{O}(\ell_n \, |S_{\mathbf{w}_n}| \, j)$. Thus, if each $f_{\mathbf{w}_n}$ is a unigram segmentation model, worst-case time complexity for each subproblem would be $\mathcal{O}(\ell_n^3)$ (if string $\mathbf{w}_n$ can be segmented into at most ℓ_n segments) and then the overall problem (1) under specification (4) is solvable in worst-case time $N \cdot \mathcal{O}(\ell^3)$. More generally, if each $f_{\mathbf{w}_n}$ is an m-gram segmentation model, then worst-case time complexity amounts to $N \cdot \mathcal{O}(\ell^{m+2})$. Importantly, this scales *linearly* with the number N of strings to align, rather than *exponentially* as the $\mathcal{O}(\ell^{(m+1)N})$ under the (non-separable) m-gram alignment model discussed above.

Unsupervised alignments The algorithms presented may be applied iteratively in order to induce multiple alignments in an unsupervised (EM-like) fashion in which sim_1 is gradually learnt (e.g., starting from a uniform initialization of sim_1). *We skip details of this, as we do not make us of it in our current experiments*. Rather, in our experiments below, we directly specify sim_1 as a sum of pairwise similarity scores which we extract from alignments produced by an off-the-shelf *pairwise* aligner.

[4] Note the difference between Eqs. (2) and (4). While each $f_{\mathbf{w}_n}$ in (4) operates on a 'row' of an alignment scheme, sim_1 in (2) acts on the 'columns'. In other words, the unigram alignment model correlates the multiply matched-up subsequences, while the separable alignment model assumes independence here.

3 Related work

Monotone alignments have a long tradition, both in NLP and bioinformatics. The classical Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) computes the optimal alignment between two sequences when only single character matches, mismatches, and skips are allowed. It is a special case of the unigram model (2) in optimization problem (1) for which $N = 2$, $S = \{(1, 0), (0, 1), (1, 1)\}$ and sim_1 takes on values from $\{0, -1\}$, depending on whether compared input subsequences match or not. As is well-known, this alignment specification is equivalent to the edit distance problem (Levenshtein, 1966) in which the minimal number of insertions, deletions and substitutions is sought that transforms one string into another. *Substring-to-substring edit operations* — or equivalently, (monotone) many-to-many alignments — have appeared in the NLP context, e.g., in (Deligne et al., 1995), (Brill and Moore, 2000), (Jiampojamarn et al., 2007), (Bisani and Ney, 2008), (Jiampojamarn et al., 2010), or, significantly earlier, in (Ukkonen, 1985), (Véronis, 1988). *Learning* edit distance/monotone alignments in an *unsupervised* manner has been the topic of, e.g., (Ristad and Yianilos, 1998), (Cotterell et al., 2014), besides the works already mentioned. All of these approaches are special cases of our unigram model outlined in Section 2 — i.e., they consider particular S (most prominently, $S = \{(1, 0), (0, 1), (1, 1)\}$) and/or restrict attention to only $N = 2$ strings.[5]

Alignments between multiple sequences, i.e., multiple sequence alignment, has also been an issue both in NLP (e.g., Covington (1998), Bhargava and Kondrak (2009)) and bioinformatics (e.g., Durbin et al. (1998)). An interesting application of alignments of multiple sequences is to determine what has been called *median string* (Kohonen, 1985) or *Steiner consensus string* (Gusfield, 1997), defined as the string $\bar{\mathbf{s}}$ that minimizes the sum of distances, for a given distance function $d(\mathbf{x}, \mathbf{y})$, to a list of strings $\mathbf{s}_1, \ldots, \mathbf{s}_N$ (Jiang et al., 2012); typically, d is the standard edit distance. As Gusfield (1997) shows, the Steiner consensus string may be retrieved from a multiple align-

[5] In Cotterell et al. (2014), context influences alignments, so that the approach goes beyond the unigram model sketched in (2), but there, too, the focus is on the situation $N = 2$ and $S = \{(1, 0), (0, 1), (1, 1)\}$.

ment of $\mathbf{s}_1, \ldots, \mathbf{s}_N$ by concatenating the column-wise majority characters in the alignment, ignoring skips. Since median string computation (and hence also the multiple many-to-many alignment problem, as we consider) is an NP-hard problem (Sim and Park, 2003), designing approximations is an active field of research. For example, Marti and Bunke (2001) ignore part of the search space by declaring matches-up of distant characters as unlikely, and Jiang et al. (2012) apply an approximation based on string embeddings in vector spaces. Paul and Eisner (2012) apply dual decomposition to compute Steiner consensus strings. Via the approach taken in this paper, median strings may be computed in case d is a (distance) function taking *substring-to-substring edit operations* into account, a seemingly straightforward, yet extremely useful generalization in several NLP applications, as indicated in the introduction.

Our approach may also be seen in the context of classifier combination for string-valued variables. While ensemble methods for structured prediction have been considered in several works (see, e.g., Nguyen and Guo (2007), Cortes et al. (2014), and references therein), a typical assumption in this situation is that the sequences to be combined have equal length, which clearly cannot be expected to hold when, e.g., the outputs of several G2P, transliteration, etc., systems must be combined. In fact, the multiple many-to-many alignment models investigated in this work could act as a preprocessing step in this setup, since the alignment precisely serves the functionality of segmenting the strings into equal number of segments/substructures. Of course, combining outputs with varying number of elements is also an issue in machine translation (e.g., Macherey and Och (2007), Heafield et al. (2009)), but, there, the problem is harder due to the potential non-monotonicities in the ordering of elements, which typically necessitates (additional) heuristics. One approach for constructing multiple alignments is here *progressive multiple alignment* (Feng and Doolittle, 1987) in which a multiple (typically one-to-one) alignment is iteratively constructed from successive pairwise alignments (Bangalore et al., 2001). Matusov et al. (2006) apply word reordering and subsequent pairwise monotone one-to-one alignments for MT system combination.

4 Data and systems

4.1 Data

We conduct experiments on the General American (GA) variant of the Combilex data set (Richmond et al., 2009). This contains about 144,000 grapheme-phoneme pairs as exemplarily illustrated in Table 2. In our experiments, we split the data into two disjoint parts, one for testing (about 28,000 word pairs) and one for training/development (the remainder).

4.2 Systems

BASELINE Our baseline system is a linear-chain conditional random field model (CRF)[6] (Lafferty et al., 2001) which we apply in the manner indicated in the introduction: after many-to-many aligning the training data as in Table 1, at *training time*, we use the CRF as a tagging model that is trained to label each input character subsequence with an output character subsequence. As *features* for the CRF, we use all n-grams of subsequences of $\mathbf{x}$ that fit inside a window of size 5 centered around the current subsequence (*context features*). We also include *linear-chain features* which allow previously generated output character subsequences to influence current output character subsequences. In essence, our baseline model is a standard discriminative approach to G2P. It is, all in all, the same approach as described in Jiampojamarn et al. (2010), except that we do not include joint n-gram features. At *test time*, we first segment a new input string $\mathbf{x}$ and then apply the CRF. Thereby, we train the segmentation module on the segmented $\mathbf{x}$ sequences, as available from the aligned training data.[7]

BASELINE+X As competitors for the baseline system, we introduce systems that rely on the predictions of one or several additional (black box/offline) systems. *At training time*, we first multiply many-to-many align the input string $\mathbf{x}$, the predictions $\hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)}$ and the true transcription $\mathbf{y}$ as illustrated in Table 3 (see Section 4.3 for details). Then, as for the baseline system, we train a CRF to label each input character

[6] We made use of the CRF++ package available at https://code.google.com/p/crfpp/.

[7] To be more precise on the training of the segmentation module, in an alignment as in Table 1, we consider the segmented $\mathbf{x}$ string — ph-oe-n-i-x — and then encode this segmentation in a binary string where 1's indicate splits. Thus, segmentation becomes, again, a sequence labling task; see, e.g., Bartlett et al. (2008) or Eger (2013) for details.

subsequence with the corresponding output character subsequence. However, this time, the CRF has access to the subsequence suggestions (as the alignments indicate) produced by the offline systems. As *features* for the extended models, we additionally include context features for all predicted strings $\hat{\mathbf{y}}^{(m)}$ (all n-grams in a window of size 3 centered around the current subsequence prediction). We also include a joint feature firing on the tuple of the current subsequence value of $\mathbf{x}$, $\hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)}$. To illustrate, when BASELINE+X tags position 2 in the (split up) input string in Table 3, it sees that its value is *ch*, that the previous input position contains *s*, that the next contains *i*, that the next two contain (*i*,*z*), that the prediction of the first system at position 2 is k, that the first system's next prediction is ai, and so forth. *At test time*, we first multiply many-to-many align $\mathbf{x}, \hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)}$, and then apply the enhanced CRF.

4.3 Alignments

To induce multiple monotone many-to-many alignments of input strings, offline system predictions and output strings, we proceed in one of two manners.

Exact alignments *Firstly*, we specify sim_1 in Eq. (2), as $\mathsf{sim}_1(\mathbf{x}_i, \hat{\mathbf{y}}_i^{(1)}, \ldots, \hat{\mathbf{y}}_i^{(M)}, \mathbf{y}_i) =$

$$\Big(\sum_{m=1}^{M} \mathsf{psim}(\mathbf{x}_i, \hat{\mathbf{y}}_i^{(m)})\Big) + \mathsf{psim}(\mathbf{x}_i, \mathbf{y}_i),$$

where psim is a pair-similarity function. The advantage with this specification is that the similarity of a tuple of subsequences is defined as the sum of *pairwise* similarity scores, which we can directly estimate from pairwise alignments of $(\mathbf{x}, \hat{\mathbf{y}}^{(m)})$ that an off-the-shelf pairwise aligner can produce (we use the Phonetisaurus aligner for this). We set $\mathsf{psim}(\mathbf{u}, \mathbf{v})$ as log-probability of observing the tuple $(\mathbf{u}, \mathbf{v})$ in the training data of pairwise aligned sequences. To illustrate, we define the similarity of (o,əʊ,əʊ,ə,əʊ,ə,əʊ) in the example in Table 3 as the pairwise similarity of (o,əʊ) (as inferred from pairwise alignments of $\mathbf{x}$ strings and system 1 transcriptions) plus the pairwise similarity of (o,əʊ) (as inferred from pairwise alignments of $\mathbf{x}$ strings and system 2 transcriptions), etc. At test time, we use the same procedure but drop the term $\mathsf{psim}(\mathbf{x}_i, \mathbf{y}_i)$ when inducing alignments. For our current purposes, we label the outlined modus as **exact (alignment) modus**.

Approx. alignments *Secondly*, we derive the optimal multiple many-to-many alignment of the strings in question by choosing an alignment that satisfies the condition that (1) each individual string $\mathbf{x}, \hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)}, \mathbf{y}$ is optimally *segmented* (e.g., *ph-oe-n-i-x* rather than *pho-eni-x*, f-i-n-ɪ-ks rather than f-inɪk-s) subject to the global constraint that (2) the number of segments must agree across the strings to align. This constitutes a separable alignment model as discussed in Section 2, and thus has much lower runtime complexity as the first model. Segmentation models can be directly learned from the pairwise alignments that Phonetisaurus produces by focusing on either the segmented $\mathbf{x}$ or $\mathbf{y}/\hat{\mathbf{y}}^{(m)}$ sequences; we choose to implement bigram individual segmentation models. This second model type may be considered an *approximation* of the first, since in a good alignment, we would not only expect individually good segmentations and agreement of segment numbers but also that subsegments are *likely correlations* of each other, precisely as our first model type captures. Therefore, we shall call this alignment modus **approximate (alignment) modus**, for our present purposes.

5 Experiments

We now describe two sets of experiments, a **controlled experiment** on the Combilex data set where we can design our offline/black box systems ourselves and where the black box systems are trained on a similar distribution as the baseline and the extended baseline systems. In particular, the black box systems operate on the same output alphabet as the extended baseline systems, which constitutes an 'ideal' situation. Thereafter, we investigate how our extended baseline system performs in a **'real-world' scenario**: we train a system on Combilex that has as supplemental information corresponding Wiktionary (and PTE, as explained below) transcriptions.

Throughout, we use as accuracy measures for all our systems **word accuray** (WACC). Word accuracy is defined as the number of correctly transcribed strings among all transcribed strings in a test sample. WACC is a strict measure that penalizes even tiny deviations from the gold-standard transcriptions, but has nowadays become standard in G2P.

5.1 A controlled experiment

In our first set of experiments, we let our offline/black box systems be the Sequitur G2P modeling toolkit (Bisani and Ney, 2008) (S) and the Phonetisaurus modeling toolkit (Novak et al., 2012) (P). We train them on *disjoint sets* of 20,000 grapheme-to-phoneme Combilex string pairs each. The performance of these two systems, on the test set of size 28,000, is indicated in Table 4. Next, we train BASELINE on *disjoint* sets (disjoint from both the training sets of P and S) of size 2,000, 5,000, 10,000 and 20,000. Making BASELINE's training sets disjoint from the training sets of the offline systems is both realistic (since a black box system would typically follow a partially distinct distribution from one's own training set distribution) and also prevents the extended baseline systems from fully adapting to the predictions of either P or S, whose training set accuracy is an upward biased representation of their true accuracy. As baseline extensions, we consider the systems BASELINE+P (+P), and BASELINE+P+S (+P+S).[8]

	Phonetisaurus	Sequitur
WACC	72.12	71.70

Table 4: Word-accuracy (in %) on the test data, for the two systems indicated.

Results are shown in Figures 1 and 2. We see that conjoining the base system with the predictions of the offline Phonetisaurus and Sequitur models substantially increases the baseline WACC, *especially in the case of little training data*. In fact, WACC increases here by almost 100% when the baseline system is complemented by $\hat{\mathbf{y}}^{(\mathrm{P})}$ and $\hat{\mathbf{y}}^{(\mathrm{S})}$. As training set size increases, differences become less and less pronounced. Eventually, we would expect them to drop to zero, since beyond some training set size, the additional features may provide no new information.[9] We also note that conjoining the two systems is more valuable than conjoining only one system, and, in Figure 2, that the models which are based on exact multiple alignments outperform the models based on approximate alignments, but not by a wide margin.

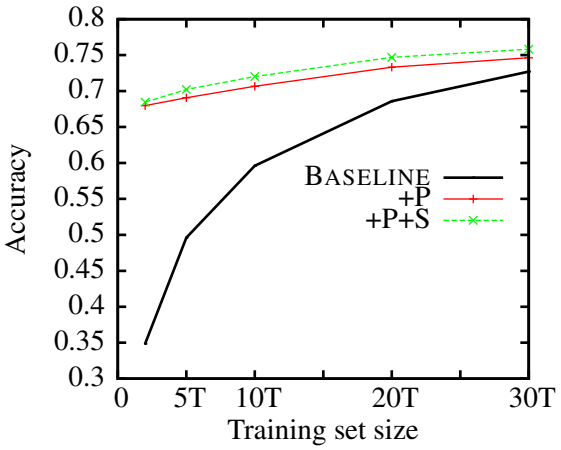

Figure 1: WACC as a function of training set size for the system indicated. Exact align. modus.

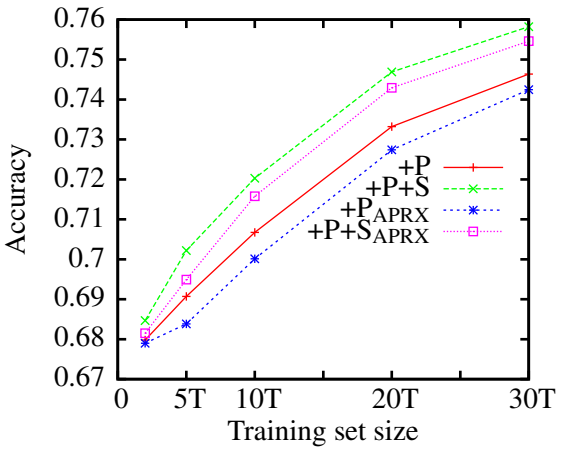

Figure 2: Comparison of models based on exact and approximate alignments; WACC as a function of training set size. APRX denotes the approximation alignment model.

Concerning differences in *alignments* between the two alignment types, exact vs. approximate, an illustrative example where the approximate model fails and the exact model does not is ('false' alignment based on the approximate model indicated):

r	ee	n	t	e	r	e	d
r	i	E	n	t	@‘	r	d
r	i	E	n	t	@‘	r	d

which nicely captures the inability of the approximate model to account for correlations between the matched-up subsequences. That is, while the segmentations of the three shown sequences appear acceptable, a matching of graphemic *t* with

[8] We omit BASELINE+S since it yielded similar results as BASELINE+P.

[9] In fact, in follow-up work, we find that the additional information may also confuse the base system when training set sizes are large enough.

phonemic n, etc., seems quite unlikely. Still, it is very promising to see that these differences in alignment quality translate into very small differences in overall string-to-string translation model performance, as Figure 2 outlines. Namely, differences in WACC are typically on the level of 1% or less (always in favor of the exact alignment model). This is a very important finding, as it indicates that string-to-string translation need not be (severely) negatively impacted by switching to the approximate alignment model, a tractable alternative to the exact models, which quickly become practically infeasible as the number of strings to align increases.

5.2 Real-world experiments

To test whether our approach may also succeed in a 'real-world setting', we use as offline/black box systems GA Wiktionary transcriptions of our input forms as well as PhotoTransEdit (PTE) transcriptions,[10] a lexicon-based G2P system which offers both GA and RP (received pronunciation) transcription of English strings. *We train and test on input strings for which both Combilex and PTE transcriptions are available, and for which both Combilex and Wiktionary transcriptions are available*.[11] Test set sizes are about 1,500 in the case of PTE and 3,500 in the case of Wiktionary. We only test here the performance of the exact alignment method, noting that, as before, approximate alignments produced slightly weaker results.

Clearly, Wiktionary and PTE differ from the Combilex data. First, both Wiktionary and PTE use different numbers of phonemic symbols than Combilex, as Table 5 illustrates. Some differences

Dataset	$\lvert\Sigma\rvert$
Combilex	54
$\text{Wiktionary}_{\text{GA}}$	107
$\text{Wiktionary}_{\text{RP}}$	116
PTE_{GA}	44
PTE_{RP}	57

Table 5: Sizes of phonetic inventaries of different data sets.

arise from the fact that, e.g., lengthening of vowels is indicated by two output letters in some data sets and only one in others. Also, phonemic transcription conventions differ, as becomes most strikingly evident in the case of RP vs. GA transcriptions — Table 6 illustrates. Finally, Wiktionary has many more phonetic symbols than the other datasets, a finding that we attribute to its crowd-sourced nature and lacking of normalization. Despite these differences in phonemic annotation standards between Combilex, Wiktionary and PTE, we observe that conjoining input strings with predicted Wiktionary or PTE transcriptions via multiple alignments leads to very good improvements in WACC over only using the input string as information source. Indeed, as shown in Table 7, for PTE, WACC increases by as much as 80% in case of small training sample (1,099 string pairs) and as much as 37% in case of medium-sized training sample (2,687 string pairs). Thus, comparing with the previous situation of homogenous systems, we also observe that the gain from including heterogeneous system is relatively weaker, as we would expect due to distinct underlying assumptions, but still impressive. Performance increases when including Wiktionary are slightly lower, most likely because it constitutes a very heterogenous source of phonetic transcriptions with user-idiosyncratic annotations (however, training set sizes are also different).[12]

	BASEL.	BASEL.+PTE_{GA}	BASEL.+PTE_{RP}
1,099	31.34	56.47	50.22
2,687	45.75	60.80	62.80
	BASEL.	BASEL.+Wik_{GA}	BASEL.+Wik_{RP}
2,000	38.44	60.71	62.18
5,000	51.69	65.81	65.96
10,000	58.97	67.30	68.66

Table 7: Top: WACC in % for baseline CRF model and the models that integrate PTE in the GA versions and RP versions, respectively. Bottom: BASELINE and BASELINE+Wiktionary.

6 Conclusion

We have generalized the task description of string transduction to include supplemental information strings. Moreover, we have suggested multiple

[10] Downloadable from http://www.photransedit.com/.

[11] This yields a clear method of comparison. An alternative would be to provide predictions for missing transcriptions. In any case, by our task definition, all systems must provide a hypothesis for an input string.

[12] To provide, for the interested reader, a comparison with Phonetisaurus and Sequitur: for the Wiktionary GA data, performance of Phonetisaurus is 41.80% (training set size 2,000), 55.70% (5,000) and 62.47% (10,000). Respective numbers for Sequitur are 40.58%, 54.84%, and 61.58%. On PTE, results are, similarly, slightly higher than our baseline, but substantially lower than the extended baseline.

b	o	t	ch	i	ng		b	a	rr	ed		a	s	th	m	a	t	i	c	s
b	o	t	ʃ	ɪ	ŋ		b	a	-	d		æ	s	-	m	æ	t	ɪ	k	s
b	A	-	tS	I	N		b	A	r	d		a	z	0	m	a	t	I	k	s

Table 6: Multiple alignments of input string, predicted PTE transcription and true (Combilex) transcription. Differences may be due to alternative phonemic conventions (e.g., Combilex has a single phonemic character representing the sound tʃ) and/or due to differences in pronunciation in GA and RP, resp.

many-to-many alignments — and a subsequent standardly extended discriminative approach — for solving string transduction (here, G2P) in this generalized setup. We have shown that, in a real-world setting, our approach may significantly beat a standard discriminative baseline, e.g., when we add Wiktionary transcriptions or predictions of a rule-based system as additional information to the input strings. The appeal of this approach lies in the fact that almost *any sort of external knowledge source may be integrated to improve the performance of a baseline system*. For example, supplemental information strings may appear in the form of transliterations of an input string in other languages; they may be predictions of other G2P systems, whether carefully manually crafted or learnt from data; they might even appear in the form of phonetic transcriptions of the input string in other dialects or languages. What distinguishes our solution to integrating supplemental information strings in string transduction settings from other research (e.g., (Bhargava and Kondrak, 2011; Bhargava and Kondrak, 2012)) is that rather than integrating systems on the *global* level of *strings*, we integrate them on the *local* level of smaller units, namely, *substrings* appropriated to the domain of application (e.g., in our context, phonemes/grapheme substructures). Both approaches may be considered complementary. Finally, another important contribution of our work is to outline an 'approximation algorithm' to inducing multiple many-to-many alignments of strings, which is otherwise an NP-hard problem for which (most likely) no efficient exact solutions exist, and to investigate its suitability for the problem task. In particular, we have seen that exact alignments lead to better overall model performance, but that the margin over the approximation is not wide.

The scope for future research of our modeling is huge: multiple many-to-many alignments may be useful in aligning cognates in linguistic research; they may be the first necessary step for many other ensemble techniques in string transduction as we have considered (Cortes et al., 2014), and they may allow, on a large scale, to boost G2P (transliteration, lemmatization, etc.) systems by integrating them with many traditional (or modern) knowledge resources such as rule- and dictionary-based lemmatizers, crowd-sourced phonetic transcriptions (e.g., based on Wiktionary), etc., with the outlook of significantly outperforming current state-of-the-art models which are based solely on input string information.

Finally, we note that we have thus far shown that supplemental information strings may be beneficial in case of overall little training data and that improvements decrease with data size. Further investigating this relationship will be of importance. Morevoer, it will be insightful to compare the exact and approximate alignment algorithms presented here with other (heuristic) alignment methods, such as iterative pairwise alignments as employed in machine translation, and to investigate how alignment quality of multiple strings impacts overall G2P performance in the setup of additional information strings.

References

S. Bangalore, G. Bodel, and G. Riccardi. 2001. Computing consensus translation from multiple machine translation systems. In *In Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU-2001*, pages 351–354.

Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. 2008. Automatic syllabification with structured svms for letter-to-phoneme conversion. In Kathleen McKeown, Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, editors, *ACL*, pages 568–576. The Association for Computer Linguistics.

Aditya Bhargava and Grzegorz Kondrak. 2009. Multiple word alignment with Profile Hidden Markov Models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, pages

43–48, Boulder, Colorado, June. Association for Computational Linguistics.

Aditya Bhargava and Grzegorz Kondrak. 2011. How do you pronounce your name?: Improving g2p with transliterations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 399–408, Stroudsburg, PA, USA. Association for Computational Linguistics.

Aditya Bhargava and Grzegorz Kondrak. 2012. Leveraging supplemental representations for sequential transduction. In *HLT-NAACL*, pages 396–406. The Association for Computational Linguistics.

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

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 286–293, Stroudsburg, PA, USA. Association for Computational Linguistics.

Corinna Cortes, Vitaly Kuznetsov, and Mehryar Mohri. 2014. Ensemble methods for structured prediction. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1134–1142.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic FSTs. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Baltimore, June. 6 pages.

Michael A. Covington. 1998. Alignment of multiple languages for historical comparison. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 275–279, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Sabine Deligne, Franois Yvon, and Frédéric Bimbot. 1995. Variable-length sequence matching for phonetic transcription using joint multigrams. In *EUROSPEECH*. ISCA.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *EMNLP*, pages 1080–1089. ACL.

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

Steffen Eger. 2013. Sequence segmentation by enumeration: An exploration. *Prague Bull. Math. Linguistics*, 100:113–132.

D. F. Feng and R. F. Doolittle. 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of molecular evolution*, 25(4):351–360.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press.

Kenneth Heafield, Greg Hanneman, and Alon Lavie. 2009. Machine translation system combination with flexible word ordering. In *Proceedings of the EACL 2009 Fourth Workshop on Statistical Machine Translation*, pages 56–60, Athens, Greece, March.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York, April. Association for Computational Linguistics.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*, pages 905–913, Columbus, Ohio, June. Association for Computational Linguistics.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2010. Integrating joint n-gram features into a discriminative training framework. In *HLT-NAACL*, pages 697–700. The Association for Computational Linguistics.

Xiaoyi Jiang, Jran Wentker, and Miquel Ferrer. 2012. Generalized median string computation by means of string embedding in vector spaces. *Pattern Recognition Letters*, 33(7):842–852.

T. Kohonen. 1985. Median strings. *Pattern Recognition Letters*, 3:309–313.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

VI Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Wolfgang Macherey and Franz Josef Och. 2007. An empirical study on computing consensus translations from multiple machine translation systems. In *EMNLP-CoNLL*, pages 986–995. ACL.

Urs-Viktor Marti and Horst Bunke. 2001. Use of positional information in sequence alignment for multiple classifier combination. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume

2096 of *Lecture Notes in Computer Science*, pages 388–398. Springer.

Evgeny Matusov, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Conference of the European Chapter of the Association for Computational Linguistics*, pages 33–40, Trento, Italy, April.

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March.

Nam Nguyen and Yunsong Guo. 2007. Comparisons of sequence labeling algorithms and extensions. In Zoubin Ghahramani, editor, *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 681–688. ACM.

Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012. WFST-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 45–49, Donostia–San Sebastin, July. Association for Computational Linguistics.

Michael J. Paul and Jason Eisner. 2012. Implicitly intersecting weighted automata using dual decomposition. In *HLT-NAACL*, pages 232–242. The Association for Computational Linguistics.

Korin Richmond, Robert A. J. Clark, and Susan Fitt. 2009. Robust LTS rules with the Combilex speech technology lexicon. In *INTERSPEECH*, pages 1295–1298. ISCA.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):522–532.

Tarek Sherif and Grzegorz Kondrak. 2007. Substring-based transliteration. In John A. Carroll, Antal van den Bosch, and Annie Zaenen, editors, *ACL*. The Association for Computational Linguistics.

Jeong Seop Sim and Kunsoo Park. 2003. The consensus string problem for a metric is np-complete. *J. of Discrete Algorithms*, 1(1):111–117, February.

Esko Ukkonen. 1985. Algorithms for approximate string matching. *Information and Control*, 64:100–118.

Jean Véronis. 1988. Computerized correction of phonographic errors. *Computers and the Humanities*, 22(1):43–56.