Letter-Phoneme Alignment: An Exploration

Sittichai Jiampojamarn and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
{sj,kondrak}@cs.ualberta.ca

Abstract

Letter-phoneme alignment is usually generated by a straightforward application of the EM algorithm. We explore several alternative alignment methods that employ phonetics, integer programming, and sets of constraints, and propose a novel approach of refining the EM alignment by aggregation of best alignments. We perform both intrinsic and extrinsic evaluation of the assortment of methods. We show that our proposed EM-Aggregation algorithm leads to the improvement of the state of the art in letter-to-phoneme conversion on several different data sets.

1 Introduction

Letter-to-phoneme (L2P) conversion (also called grapheme-to-phoneme conversion) is the task of predicting the pronunciation of a word given its orthographic form by converting a sequence of letters into a sequence of phonemes. The L2P task plays a crucial role in speech synthesis systems (Schroeter et al., 2002), and is an important part of other applications, including spelling correction (Toutanova and Moore, 2001) and speech-to-speech machine translation (Engelbrecht and Schultz, 2005). Many data-driven techniques have been proposed for letter-to-phoneme conversion systems, including neural networks (Sejnowski and Rosenberg, 1987), decision trees (Black et al., 1998), pronunciation by analogy (Marchand and Damper, 2000), Hidden Markov Models (Taylor, 2005), and constraint satisfaction (Bosch and Canisius, 2006).

Letter-phoneme alignment is an important step in the L2P task. The training data usually consists of pairs of letter and phoneme sequences, which are not aligned. Since there is no explicit information indicating the relationships between individual letter and phonemes, these must be inferred by a letter-phoneme alignment algorithm before a prediction model can be trained. The quality of the alignment affects the accuracy of L2P conversion. Letter-phoneme alignment is closely related to transliteration alignment (Pervouchine et al., 2009), which involves graphemes representing different writing scripts. Letter-phoneme alignment may also be considered as a task in itself; for example, in the alignment of speech transcription with text in spoken corpora.

Most previous L2P approaches induce the alignment between letters and phonemes with the expectation maximization (EM) algorithm. In this paper, we propose a number of alternative alignment methods, and compare them to the EM-based algorithms using both intrinsic and extrinsic evaluations. The intrinsic evaluation is conducted by comparing the generated alignments to a manually-constructed gold standard. The extrinsic evaluation uses two different generation techniques to perform letter-to-phoneme conversion on several different data sets. We discuss the advantages and disadvantages of various methods, and show that better alignments tend to improve the accuracy of the L2P systems regardless of the actual technique. In particular, one of our proposed methods advances the state of the art in L2P conversion. We also examine the relationship between alignment entropy and alignment quality.

This paper is organized as follows. In Section 2, we enumerate the assumptions that the alignment methods commonly adopt. In Section 3, we review previous work that employs the EM approach. In Sections 4, 5 and 6, we describe alternative approaches based on phonetics, manually-constructed constraints, and Integer Programming, respectively. In Section 7, we propose an algorithm to refine the alignments produced by EM. Sections 8 and 9 are devoted to the intrinsic and extrinsic evaluation of various approaches. Section 10 concludes the paper.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 780–788,
Uppsala, Sweden, 11-16 July 2010.

2 Background

We define the letter-phoneme alignment task as the problem of inducing *links* between units that are related by pronunciation. Each link is an instance of a specific *mapping* between letters and phonemes. The leftmost example alignment of the word *accuse* [@kjuz] below includes 1-1, 1-0, 1-2, and 2-1 links. The letter e is considered to be linked to special *null phoneme*.

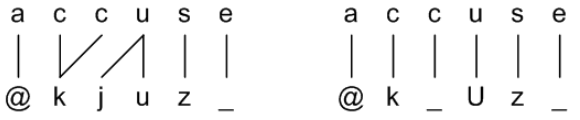

Figure 1: Two alignments of *accuse*.

The following constraints on links are assumed by some or all alignment models:

- the *monotonicity* constraint prevents links from crossing each other;
- the *representation* constraint requires each phoneme to be linked to at least one letter, thus precluding nulls on the letter side;
- the *one-to-one* constraint stipulates that each letter and phoneme may participate in at most one link.

These constraints increasingly reduce the search space and facilitate the training process for the L2P generation models.

We refer to an alignment model that assumes all three constraints as a pure one-to-one (1-1) model. By allowing only 1-1 and 1-0 links, the alignment task is thus greatly simplified. In the simplest case, when the number of letters is equal to the number of phonemes, there is only one possible alignment that satisfies all three constraints. When there are more letters than phonemes, the search is reduced to identifying letters that must be linked to null phonemes (the process referred to as "epsilon scattering" by Black et al. (1998)). In some words, however, one letter clearly represents more than one phoneme; for example, u in Figure 1. Moreover, a pure 1-1 approach cannot handle cases where the number of phonemes exceeds the number of letters. A typical solution to overcome this problems is to introduce so-called *double phonemes* by merging adjacent phonemes that could be represented as a single letter. For example, a double phoneme U would replace a sequence of the phonemes j and u in Figure 1. This solution requires a manual extension of the set of phonemes present in the data. By convention, we regard the models that include a restricted set of 1-2 mappings as 1-1 models.

Advanced L2P approaches, including the joint n-gram models (Bisani and Ney, 2008) and the joint discriminative approach (Jiampojamarn et al., 2007) eliminate the one-to-one constraint entirely, allowing for linking of multiple letters to multiple phonemes. We refer to such models as many-to-many (M-M) models.

3 EM Alignment

Early EM-based alignment methods (Daelemans and Bosch, 1997; Black et al., 1998; Damper et al., 2005) were generally pure 1-1 models. The 1-1 alignment problem can be formulated as a dynamic programming problem to find the maximum score of alignment, given a probability table of aligning letter and phoneme as a mapping function. The dynamic programming recursion to find the most likely alignment is the following:

$$C_{i,j} = \max \begin{cases} C_{i-1,j-1} + \delta(x_i, y_j) \\ C_{i-1,j} + \delta(x_i, \epsilon) \\ C_{i,j-1} + \delta(\epsilon, y_j) \end{cases} \quad (1)$$

where $\delta(x_i, \epsilon)$ denotes a probability that a letter x_i aligns with a null phoneme and $\delta(\epsilon, y_j)$ denotes a probability that a null letter aligns with a phoneme y_j. In practice, the latter probability is often set to zero in order to enforce the representation constraint, which facilitates the subsequent phoneme generation process. The probability table $\delta(x_i, y_j)$ can be initialized by a uniform distribution and is iteratively re-computed (M-step) from the most likely alignments found at each iteration over the data set (E-step). The final alignments are constructed after the probability table converges.

M2M-aligner (Jiampojamarn et al., 2007) is a many-to-many (M-M) alignment algorithm based on EM that allows for mapping of multiple letters to multiple phonemes. Algorithm 1 describes the E-step of the many-to-many alignment algorithm. γ represents partial counts collected over all possible mappings between substrings of letters and phonemes. The maximum lengths of letter and phoneme substrings are controlled by the

Algorithm 1: Many-to-many alignment

Input: $x, y, maxX, maxY, \gamma$
Output: γ

1 $\alpha :=$ FORWARD-M2M $(x, y, maxX, maxY)$
2 $\beta :=$ BACKWARD-M2M $(x, y, maxX, maxY)$
3 $T = |x| + 1$, $V = |y| + 1$
4 **if** $(\alpha_{T,V} = 0)$ **then**
5 return
6 **for** $t = 1..T$, $v = 1..V$ **do**
7 **for** $i = 1..maxX$ **st** $t - i \geq 0$ **do**
8 $\gamma(x_{t-i+1}^{t}, \epsilon) \mathrel{+}= \frac{\alpha_{t-i,v}\delta(x_{t-i+1}^{t}, \epsilon)\beta_{t,v}}{\alpha_{T,V}}$
9 **for** $i = 1..maxX$ **st** $t - i \geq 0$ **do**
10 **for** $j = 1..maxY$ **st** $v - j \geq 0$ **do**
11 $\gamma(x_{t-i+1}^{t}, y_{v-j+1}^{v}) \mathrel{+}= \frac{\alpha_{t-i,v-j}\delta(x_{t-i+1}^{t}, y_{v-j+1}^{v})\beta_{t,v}}{\alpha_{T,V}}$

$maxX$ and $maxY$ parameters. The forward probability α is estimated by summing the probabilities from left to right, while the backward probability β is estimated in the opposite direction. The FORWARD-M2M procedure is similar to line 3 to 10 of Algorithm 1, except that it uses Equation 2 in line 8 and 3 in line 11. The BACKWARD-M2M procedure is analogous to FORWARD-M2M.

$$\alpha_{t,v} \mathrel{+}= \delta(x_{t-i+1}^{t}, \epsilon)\alpha_{t-i,v} \quad (2)$$

$$\alpha_{t,v} \mathrel{+}= \delta(x_{t-i+1}^{t}, y_{v-j+1}^{v})\alpha_{t-i,v-j} \quad (3)$$

In M-step, the partial counts are normalized by using a conditional distribution to create the mapping probability table δ. The final many-to-many alignments are created by finding the most likely paths using the Viterbi algorithm based on the learned mapping probability table. The source code of M2M-aligner is publicly available.[1]

Although the many-to-many approach tends to create relatively large models, it generates more intuitive alignments and leads to improvement in the L2P accuracy (Jiampojamarn et al., 2007). However, since many links involve multiple letters, it also introduces additional complexity in the phoneme prediction phase. One possible solution is to apply a letter segmentation algorithm at test time to cluster letters according to the alignments in the training data. This is problematic because of error propagation inherent in such a process. A better solution is to combine segmentation and decoding using a phrasal decoder (e.g. (Zens and Ney, 2004)).

[1] `http://code.google.com/p/m2m-aligner/`

4 Phonetic alignment

The EM-based approaches to L2P alignment treat both letters and phonemes as abstract symbols. A completely different approach to L2P alignment is based on the phonetic similarity between phonemes. The key idea of the approach is to represent each letter by a phoneme that is likely to be represented by the letter. The actual phonemes on the phoneme side and the phonemes representing letters on the letter side can then be aligned on the basis of phonetic similarity between phonemes. The main advantage of the phonetic alignment is that it requires no training data, and so can be readily be applied to languages for which no pronunciation lexicons are available.

The task of identifying the phoneme that is most likely to be represented by a given letter may seem complex and highly language-dependent. For example, the letter a can represent no less than 12 different English vowels. In practice, however, absolute precision is not necessary. Intuitively, the letters that had been chosen (often centuries ago) to represent phonemes in any orthographic system tend to be close to the prototype phoneme in the original script. For example, the letter 'o' represented a mid-high rounded vowel in Classical Latin and is still generally used to represent similar vowels.

The following simple heuristic works well for a number of languages: treat every letter as if it were a symbol in the International Phonetic Alphabet (IPA). The set of symbols employed by the IPA includes the 26 letters of the Latin alphabet, which tend to correspond to the phonemes that they represent in the Latin script. For example, the IPA symbol [m] denotes a voiced bilabial nasal consonant, which is the phoneme represented by the letter m in most languages that utilize Latin script.

ALINE (Kondrak, 2000) performs phonetic alignment of two strings of phonemes. It combines a dynamic programming alignment algorithm with an appropriate scoring scheme for computing phonetic similarity on the basis of multivalued features. The example below shows the alignment of the word *sheath* to its phonetic transcription [ʃiθ]. ALINE correctly links the most similar pairs of phonemes (s:ʃ, e:i, t:θ).[2]

[2] ALINE can also be applied to non-Latin scripts by replacing every grapheme with the IPA symbol that is phonetically closest to it.

s	h	e	a	t	h
|	|	|	|	|	|
ʃ	-	i	-	θ	-

Since ALINE is designed to align phonemes with phonemes, it does not incorporate the representation constraint. In order to avoid the problem of unaligned phonemes, we apply a post-processing algorithm, which also handles 1-2 links. The algorithm first attempts to remove 0-1 links by merging them with the adjacent 1-0 links. If this is not possible, the algorithm scans a list of valid 1-2 mappings, attempting to replace a pair of 0-1 and 1-1 links with a single 1-2 link. If this also fails, the entire entry is removed from the training set. Such entries often represent unusual foreign-origin words or outright annotation errors. The number of unaligned entries rarely exceeds 1% of the data.

The post-processing algorithm produces an alignment that contains 1-0, 1-1, and 1-2 links. The list of valid 1-2 mappings must be prepared manually. The length of such lists ranges from 1 for Spanish and German (x:[ks]) to 17 for English. This approach is more robust than the double-phoneme technique because the two phonemes are clustered only if they can be linked to the corresponding letter.

5 Constraint-based alignment

One of the advantages of the phonetic alignment is its ability to rule out phonetically implausible letter-phoneme links, such as o:p. We are interested in establishing whether a set of allowable letter-phoneme mappings could be derived directly from the data without relying on phonetic features.

Black et al. (1998) report that constructing lists of possible phonemes for each letter leads to L2P improvement. They produce the lists in a "semi-automatic", interactive manner. The lists constrain the alignments performed by the EM algorithm and lead to better-quality alignments.

We implement a similar interactive program that incrementally expands the lists of possible phonemes for each letter by refining alignments constrained by those lists. However, instead of employing the EM algorithm, we induce alignments using the standard edit distance algorithm with substitution and deletion assigned the same cost. In cases when there are multiple alternative alignments that have the same edit distance, we randomly choose one of them. Furthermore, we extend this idea also to many-to-many alignments. In addition to lists of phonemes for each letter (1-1 mappings), we also construct lists of many-to-many mappings, such as ee:i, sch:ʃ, and ew:ju. In total, the English set contains 377 mappings, of which more than half are of the 2-1 type.

6 IP Alignment

The process of manually inducing allowable letter-phoneme mappings is time-consuming and involves a great deal of language-specific knowledge. The Integer Programming (IP) framework offers a way to induce similar mappings without a human expert in the loop. The IP formulation aims at identifying the smallest set of letter-phoneme mappings that is sufficient to align all instances in the data set.

Our IP formulation employs the three constraints enumerated in Section 2, except that the one-to-one constraint is relaxed in order to identify a small set of 1-2 mappings. We specify two types of binary variables that correspond to local alignment links and global letter-phoneme mappings, respectively. We distinguish three types of local variables, X, Y, and Z, which correspond to 1-0, 1-1, and 1-2 links, respectively. In order to minimize the number of global mappings, we set the following objective that includes variables corresponding to 1-1 and 1-2 mappings:

$$\text{minimize} : \sum_{l,p} G(l,p) + \sum_{l,p_1,p_2} G(l,p_1p_2) \quad (4)$$

We adopt a simplifying assumption that any letter can be linked to a null phoneme, so no global variables corresponding to 1-0 mappings are necessary.

In the lexicon entry k, let l_{ik} be the letter at position i, and p_{jk} the phoneme at position j. In order to prevent the alignments from utilizing letter-phoneme mappings which are not on the global list, we impose the following constraints:

$$\forall_{i,j,k} Y(i,j,k) \leq G(l_{ik}, p_{jk}) \quad (5)$$

$$\forall_{i,j,k} Z(i,j,k) \leq G(l_{ik}, p_{jk}p_{(j+1)k}) \quad (6)$$

For example, the local variable $Y(i,j,k)$ is set if l_{ik} is linked to p_{jk}. A corresponding global variable $G(l_{ik}, p_{jk})$ is set if the list of allowed letter-phoneme mappings includes the link (l_{ik}, p_{jk}). Activating the local variable implies activating the corresponding global variable, but not vice versa.

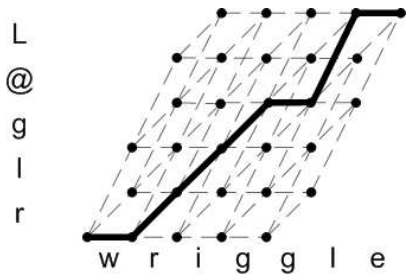

Figure 2: A network of possible alignment links.

We create a network of possible alignment links for each lexicon entry k, and assign a binary variable to each link in the network. Figure 2 shows an alignment network for the lexicon entry k: *wriggle* [r I g @ L]. There are three 1-0 links (level), three 1-1 links (diagonal), and one 1-2 link (steep). The local variables that receive the value of 1 are the following: X(1,0,k), Y(2,1,k), Y(3,2,k), Y(4,3,k), X(5,3,k), Z(6,5,k), and X(7,5,k). The corresponding global variables are: G(r,r), G(i,I), G(g,g), and G(l,@L).

We create constraints to ensure that the link variables receiving a value of 1 form a left-to-right path through the alignment network, and that all other link variables receive a value of 0. We accomplish this by requiring the sum of the links entering each node to equal the sum of the links leaving each node.

$$\forall_{i,j,k} \quad X(i,j,k) + Y(i,j,k) + Z(i,j,k) = X(i+1,j,k) + Y(i+1,j+1,k) + Z(i+1,j+2,k)$$

We found that inducing the IP model with the full set of variables gives too much freedom to the IP program and leads to inferior results. Instead, we first run the full set of variables on a subset of the training data which includes only the lexicon entries in which the number of phonemes exceeds the number of letters. This generates a small set of plausible 1-2 mappings. In the second pass, we run the model on the full data set, but we allow only the 1-2 links that belong to the initial set of 1-2 mappings induced in the first pass.

6.1 Combining IP with EM

The set of allowable letter-phoneme mappings can also be used as an input to the EM alignment algorithm. We call this approach *IP-EM*. After inducing the minimal set of letter-phoneme mappings, we constrain EM to use only those mappings with the exclusion of all others. We initialize the probability of the minimal set with a uniform distribution, and set it to zero for other mappings. We train the EM model in a similar fashion to the many-to-many alignment algorithm presented in Section 3, except that we limit the letter size to be one letter, and that any letter-phoneme mapping that is not in the minimal set is assigned zero count during the E-step. The final alignments are generated after the parameters converge.

7 Alignment by aggregation

During our development experiments, we observed that the technique that combines IP with EM described in the previous section generally leads to alignment quality improvement in comparison with the IP alignment. Nevertheless, because EM is constrained not to introduce any new letter-phoneme mappings, many incorrect alignments are still proposed. We hypothesized that instead of pre-constraining EM, a post-processing of EM's output may lead to better results.

M2M-aligner has the ability to create precise links involving more than one letter, such as *ph*:f. However, it also tends to create non-intuitive links such as *se*:z for the word *phrase* [f r e z], where e is clearly a case of a "silent" letter. We propose an alternative EM-based alignment method that instead utilizes a list of alternative *one-to-many* alignments created with M2M-aligner and aggregates 1-M links into M-M links in cases when there is a disagreement between alignments within the list. For example, if the list contains the two alignments shown in Figure 3, the algorithm creates a single many-to-many alignment by merging the first pair of 1-1 and 1-0 links into a single *ph:f* link. However, the two rightmost links are *not* merged because there is no disagreement between the two initial alignments. Therefore, the resulting alignment reinforces the *ph*:f mapping, but avoids the questionable *se*:z link.

```
p h r a s e      p h r a s e
| | | | | |      | | | | | |
f - r e z -      - f r e z -
```

Figure 3: Two alignments of *phrase*.

In order to generate the list of best alignments, we use Algorithm 2, which is an adaptation of the standard Viterbi algorithm. Each cell $Q_{t,v}$ contains a list of n-best scores that correspond to al-

Algorithm 2: Extracting n-best alignments

Input: x, y, δ
Output: $Q_{T,V}$

```
1  T = |x| + 1 , V = |y| + 1
2  for t = 1..T do
3      Q_{t,v} = ∅
4      for v = 1..V do
5          for q ∈ Q_{t-1,v} do
6              append q · δ(x_t, ϵ) to Q_{t,v}
7          for j = 1..maxY st v − j ≥ 0 do
8              for q ∈ Q_{t-1,v-j} do
9                  append q · δ(x_t, y^v_{v-j+1}) to Q_{t,v}
10         sort Q_{t,v}
11         Q_{t,v} = Q_{t,v}[1 : n]
```

ternative alignments during the forward pass. In line 9, we consider all possible 1-M links between letter x_t and phoneme substring y^v_{v-j+1}. At the end of the main loop, we keep at most n best alignments in each $Q_{t,v}$ list.

Algorithm 2 yields n-best alignments in the $Q_{T,V}$ list. However, in order to further restrict the set of high-quality alignments, we also discard the alignments with scores below threshold R with respect to the best alignment score. Based on the experiments with the development set, we set $R = 0.8$ and $n = 10$.

8 Intrinsic evaluation

For the intrinsic evaluation, we compared the generated alignments to gold standard alignments extracted from the the core vocabulary of the Combilex data set (Richmond et al., 2009). Combilex is a high quality pronunciation lexicon with explicit expert manual alignments. We used a subset of the lexicon composed of the core vocabulary containing 18,145 word-phoneme pairs. The alignments contain 550 mappings, which include complex 4-1 and 2-3 types.

Each alignment approach creates alignments from unaligned word-phoneme pairs in an unsupervised fashion. We distinguish between the 1-1 and M-M approaches. We report the alignment quality in terms of precision, recall and F-score. Since the gold standard includes many links that involve multiple letters, the theoretical upper bound for recall achieved by a one-to-one approach is 90.02%. However, it is possible to obtain the perfect precision because we count as correct all 1-1 links that are *consistent* with the M-M links in the gold standard. The F-score corresponding to perfect precision and the upper-bound recall is 94.75%.

Alignment entropy is a measure of alignment quality proposed by Pervouchine et al. (2009) in the context of transliteration. The entropy indicates the uncertainty of mapping between letter l and phoneme p resulting from the alignment: We compute the alignment entropy for each of the methods using the following formula:

$$H = -\sum_{l,p} P(l,p) \log P(l|p) \qquad (7)$$

Table 1 includes the results of the intrinsic evaluation. (the two rightmost columns are discussed in Section 9). The baseline *BaseEM* is an implementation of the one-to-one alignment method of (Black et al., 1998) *without* the allowable list. *ALINE* is the phonetic method described in Section 4. *SeedMap* is the hand-seeded method described in Section 5. *M-M-EM* is the M2M-aligner approach of Jiampojamarn et al. (2007). *1-M-EM* is equivalent to *M-M-EM* but with the restriction that each link contains exactly one letter. *IP-align* is the alignment generated by the IP formulation from Section 6. *IP-EM* is the method that combines IP with EM described in Section 6.1. *EM-Aggr* is our final many-to-many alignment method described in Section 7. *Oracle* corresponds to the gold-standard alignments from Combilex.

Overall, the M-M models obtain lower precision but higher recall and F-score than 1-1 models, which is to be expected as the gold standard is defined in terms of M-M links. *ALINE* produces the most accurate alignments among the 1-1 methods, with the precision and recall values that are very close to the theoretical upper bounds. Its precision is particularly impressive: on average, only one link in a thousand is not consistent with the gold standard. In terms of word accuracy, 98.97% words have no incorrect links. Out of 18,145 words, only 112 words contain incorrect links, and further 75 words could not be aligned. The ranking of the 1-1 methods is quite clear: *ALINE* followed by *IP-EM*, *1-M-EM*, *IP-align*, and BaseEM. Among the M-M methods, EM-Aggr has slightly better precision than M-M-EM, but its recall is much worse. This is probably caused by the aggregation strategy causing EM-Aggr to "lose" a significant number of correct links. In general, the entropy measure does not mirror the quality of the alignment.

Aligner	Precision	Recall	F_1 score	Entropy	L2P 1-1	L2P M-M
BaseEM	96.54	82.84	89.17	0.794	50.00	65.38
ALINE	**99.90**	**89.54**	**94.44**	0.672	**54.85**	68.74
1-M-EM	99.04	89.15	93.84	**0.636**	53.91	**69.13**
IP-align	98.30	88.49	93.14	0.706	52.66	68.25
IP-EM	99.31	89.40	94.09	0.651	53.86	68.91
M-M-EM	96.54	97.13	96.83	0.655	—	68.52
EM-Aggr	96.67	93.39	95.00	0.635	—	**69.35**
SeedMap	**97.88**	**97.44**	**97.66**	**0.634**	—	68.69
Oracle	100.0	100.0	100.0	0.640	—	69.35

Table 1: Alignment quality, entropy, and L2P conversion accuracy on the Combilex data set.

Aligner	Celex-En	CMUDict	NETtalk	OALD	Brulex
BaseEM	75.35	60.03	54.80	67.23	81.33
ALINE	**81.50**	66.46	54.90	**72.12**	**89.37**
1-M-EM	80.12	66.66	**55.00**	71.11	88.97
IP-align	78.88	62.34	53.10	70.46	83.72
IP-EM	80.95	**67.19**	54.70	71.24	87.81

Table 2: L2P word accuracy using the TiMBL-based generation system.

9 Extrinsic evaluation

In order to investigate the relationship between the alignment quality and L2P performance, we feed the alignments to two different L2P systems. The first one is a classification-based learning system employing TiMBL (Daelemans et al., 2009), which can utilize either 1-1 or 1-M alignments. The second system is the state-of-the-art online discriminative training for letter-to-phoneme conversion (Jiampojamarn et al., 2008), which accepts both 1-1 and M-M types of alignment. Jiampojamarn et al. (2008) show that the online discriminative training system outperforms a number of competitive approaches, including joint n-grams (Demberg et al., 2007), constraint satisfaction inference (Bosch and Canisius, 2006), pronunciation by analogy (Marchand and Damper, 2006), and decision trees (Black et al., 1998). The decoder module uses standard Viterbi for the 1-1 case, and a phrasal decoder (Zens and Ney, 2004) for the M-M case. We report the L2P performance in terms of word accuracy, which rewards only the completely correct output phoneme sequences. The data set is randomly split into 90% for training and 10% for testing. For all experiments, we hold out 5% of our training data to determine when to stop the online training process.

Table 1 includes the results on the Combilex data set. The two rightmost columns correspond to our two test L2P systems. We observe that although better alignment quality does not always translate into better L2P accuracy, there is nevertheless a strong correlation between the two, especially for the weaker phoneme generation system. Interestingly, *EM-Aggr* matches the L2P accuracy obtained with the gold standard alignments. However, there is no reason to claim that the gold standard alignments are optimal for the L2P generation task, so that result should not be considered as an upper bound. Finally, we note that alignment entropy seems to match the L2P accuracy better than it matches alignment quality.

Tables 2 and 3 show the L2P results on several evaluation sets: English Celex, CMUDict, NETTalk, OALD, and French Brulex. The training sizes range from 19K to 106K words. We follow exactly the same data splits as in Bisani and Ney (2008).

The TiMBL L2P generation method (Table 2) is applicable only to the 1-1 alignment models. *ALINE* produces the highest accuracy on four out of six datasets (including Combilex). The performance of *IP-EM* is comparable to *1-M-EM*, but not consistently better. *IP-align* does not seem to measure up to the other algorithms.

The discriminative approach (Table 3) is flexible enough to utilize all kinds of alignments. However, the M-M models perform clearly better than 1-1 models. The only exception is NetTalk, which

Aligner	Celex-En	CMUDict	NETTalk	OALD	Brulex
BaseEM	85.66	71.49	68.60	80.76	88.41
ALINE	87.96	75.05	69.52	81.57	94.56
1-M-EM	88.08	75.11	70.78	81.78	94.54
IP-EM	88.00	75.09	70.10	81.76	94.96
M-M-EM	88.54	75.41	70.18	82.43	95.03
EM-Aggr	**89.11**	**75.52**	**71.10**	**83.32**	**95.07**
joint n-gram	88.58	75.47	69.00	82.51	93.75

Table 3: L2P word accuracy using the online discriminative system.

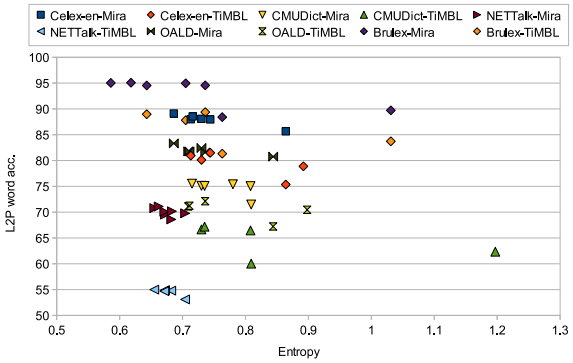

Figure 4: L2P word accuracy vs. alignment entropy.

can be attributed to the fact that NetTalk already includes double-phonemes in its original formulation. In general, the *1-M-EM* method achieves the best results among the 1-1 alignment methods, Overall, *EM-Aggr* achieves the best word accuracy in comparison to other alignment methods including the *joint n-gram* results, which are taken directly from the original paper of Bisani and Ney (2008). Except the Brulex and CMUDict data sets, the differences between *EM-Aggr* and *M-M-EM* are statistically significant according to McNemar's test at 90% confidence level.

Figure 4 contains a plot of alignment *entropy* values vs. L2P word accuracy. Each point represent an application of a particular alignment method to a different data sets. It appears that there is only weak correlation between alignment entropy and L2P accuracy. So far, we have been unable to find either direct or indirect evidence that alignment entropy is a reliable measure of letter-phoneme alignment quality.

10 Conclusion

We investigated several new methods for generating letter-phoneme alignments. The phonetic alignment is recommended for languages with little or no training data. The constraint-based approach achieves excellent accuracy at the cost of manual construction of seed mappings. The IP alignment requires no linguistic expertise and guarantees a minimal set of letter-phoneme mappings. The alignment by aggregation advances the state-of-the-art results in L2P conversion. We thoroughly evaluated the resulting alignments on several data sets by using them as input to two different L2P generation systems. Finally, we employed an independently constructed lexicon to demonstrate the close relationship between alignment quality and L2P conversion accuracy.

One open question that we would like to investigate in the future is whether L2P conversion accuracy could be improved by treating letter-phoneme alignment links as latent variables, instead of committing to a single best alignment.

Acknowledgments

This research was supported by the Alberta Ingenuity, Informatics Circle of Research Excellence (iCORE), and Natural Science of Engineering Research Council of Canada (NSERC).

References

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

Alan W. Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *The Third ESCA Workshop in Speech Synthesis*, pages 77–80.

Antal Van Den Bosch and Sander Canisius. 2006. Improved morpho-phonological sequence processing with constraint satisfaction inference. *Proceedings of the Eighth Meeting of the ACL Special Interest Group in Computational Phonology, SIGPHON '06*, pages 41–49.

Walter Daelemans and Antal Van Den Bosch. 1997. Language-independent data-oriented grapheme-to-phoneme conversion. In *Progress in Speech Synthesis*, pages 77–89. New York, USA.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2009. *TiMBL: Tilburg Memory Based Learner, version 6.2, Reference Guide.* ILK Research Group Technical Report Series no. 09-01.

Robert I. Damper, Yannick Marchand, John DS. Marsters, and Alexander I. Bazin. 2005. Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology*, 8(2):147–160.

Vera Demberg, Helmut Schmid, and Gregor Möhler. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 96–103, Prague, Czech Republic.

Herman Engelbrecht and Tanja Schultz. 2005. Rapid development of an afrikaans-english speech-to-speech translator. In *International Workshop of Spoken Language Translation (IWSLT)*, Pittsburgh, PA, USA.

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, USA.

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.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of NAACL 2000: 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 288–295.

Yannick Marchand and Robert I. Damper. 2000. A multistrategy approach to improving pronunciation by analogy. *Computational Linguistics*, 26(2):195–219.

Yannick Marchand and Robert I. Damper. 2006. Can syllabification improve pronunciation by analogy of English? *Natural Language Engineering*, 13(1):1–24.

Vladimir Pervouchine, Haizhou Li, and Bo Lin. 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, Suntec, Singapore, August. Association for Computational Linguistics.

Korin Richmond, Robert A. J. Clark, and Sue Fitt. 2009. Robust LTS rules with the Combilex speech technology lexicon. In *Proceedings od Interspeech*, pages 1295–1298.

Juergen Schroeter, Alistair Conkie, Ann Syrdal, Mark Beutnagel, Matthias Jilka, Volker Strom, Yeon-Jun Kim, Hong-Goo Kang, and David Kapilow. 2002. A perspective on the next challenges for TTS research. In *IEEE 2002 Workshop on Speech Synthesis*.

Terrence J. Sejnowski and Charles R. Rosenberg. 1987. Parallel networks that learn to pronounce English text. In *Complex Systems*, pages 1:145–168.

Paul Taylor. 2005. Hidden Markov Models for grapheme to phoneme conversion. In *Proceedings of the 9th European Conference on Speech Communication and Technology*.

Kristina Toutanova and Robert C. Moore. 2001. Pronunciation modeling for improved spelling correction. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 144–151, Morristown, NJ, USA.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 257–264, Boston, Massachusetts, USA.