Joint Processing and Discriminative Training for Letter-to-Phoneme Conversion

Sittichai Jiampojamarn[†] **Colin Cherry**[‡] **Grzegorz Kondrak**[†]

[†]Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
{sj,kondrak}@cs.ualberta.ca

[‡]Microsoft Research
One Microsoft Way
Redmond, WA, 98052
colinc@microsoft.com

Abstract

We present a discriminative structure-prediction model for the letter-to-phoneme task, a crucial step in text-to-speech processing. Our method encompasses three tasks that have been previously handled separately: input segmentation, phoneme prediction, and sequence modeling. The key idea is online discriminative training, which updates parameters according to a comparison of the current system output to the desired output, allowing us to train all of our components together. By folding the three steps of a pipeline approach into a unified dynamic programming framework, we are able to achieve substantial performance gains. Our results surpass the current state-of-the-art on six publicly available data sets representing four different languages.

1 Introduction

Letter-to-phoneme (L2P) conversion is the task of predicting the pronunciation of a word, represented as a sequence of phonemes, from its orthographic form, represented as a sequence of letters. 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).

Converting a word into its phoneme representation is not a trivial task. Dictionary-based approaches cannot achieve this goal reliably, due to unseen words and proper names. Furthermore, the construction of even a modestly-sized pronunciation dictionary requires substantial human effort for each new language. Effective rule-based approaches can be designed for some languages such as Spanish. However, Kominek and Black (2006) show that in languages with a less transparent relationship between spelling and pronunciation, such as English, Dutch, or German, the number of letter-to-sound rules grows almost linearly with the lexicon size. Therefore, most recent work in this area has focused on machine-learning approaches.

In this paper, we present a joint framework for letter-to-phoneme conversion, powered by online discriminative training. By updating our model parameters online, considering only the current system output and its feature representation, we are able to not only incorporate overlapping features, but also to use the same learning framework with increasingly complex search techniques. We investigate two online updates: averaged perceptron and Margin Infused Relaxed Algorithm (MIRA). We evaluate our system on L2P data sets covering English, French, Dutch and German. In all cases, our system outperforms the current state of the art, reducing the best observed error rate by as much as 46%.

2 Previous work

Letter-to-phoneme conversion is a complex task, for which a number of diverse solutions have been proposed. It is a structure prediction task; both the input and output are structured, consisting of sequences of letters and phonemes, respectively. This makes L2P a poor fit for many machine-learning techniques that are formulated for binary classification.

The L2P task is also characterized by the existence of a hidden structure connecting input to output. The training data consists of letter strings paired with phoneme strings, without explicit links connecting individual letters to phonemes. The subtask of inserting these links, called letter-to-phoneme alignment, is not always straightforward. For example, consider the word "phoenix" and its corresponding phoneme sequence [f i n ɪ k s], where we encounter cases of two letters generating a single phoneme (ph→f), and a single letter generating two phonemes (x→k s). Fortunately, alignments between letters and phonemes can be discovered reliably with unsupervised generative models. Originally, L2P systems assumed one-to-one alignment (Black et al., 1998; Damper et al., 2005), but recently many-to-many alignment has been shown to perform better (Bisani and Ney, 2002; Jiampojamarn et al., 2007). Given such an alignment, L2P can be viewed either as a sequence of classification problems, or as a sequence modeling problem.

In the classification approach, each phoneme is predicted independently using a multi-class classifier such as decision trees (Daelemans and Bosch, 1997; Black et al., 1998) or instance-based learning (Bosch and Daelemans, 1998). These systems predict a phoneme for each input letter, using the letter and its context as features. They leverage the structure of the input but ignore any structure in the output.

L2P can also be viewed as a sequence modeling, or tagging problem. These approaches model the structure of the output, allowing previously predicted phonemes to inform future decisions. The supervised Hidden Markov Model (HMM) applied by Taylor (2005) achieved poor results, mostly because its maximum-likelihood emission probabilities cannot be informed by the emitted letter's context. Other approaches, such as those of Bisani and Ney (2002) and Marchand and Damper (2000), have shown that better performance can be achieved by pairing letter substrings with phoneme substrings, allowing context to be captured implicitly by these groupings.

Recently, two hybrid methods have attempted to capture the flexible context handling of classification-based methods, while also modeling the sequential nature of the output. The constraint satisfaction inference (CSInf) approach (Bosch and Canisius, 2006) improves the performance of instance-based classification (Bosch and Daelemans, 1998) by predicting for each letter a trigram of phonemes consisting of the previous, current and next phonemes in the sequence. The final output sequence is the sequence of predicted phonemes that satisfies the most unigram, bigram and trigram agreement constraints. The second hybrid approach (Jiampojamarn et al., 2007) also extends instance-based classification. It employs a many-to-many letter-to-phoneme alignment model, allowing substrings of letters to be classified into substrings of phonemes, and introducing an input segmentation step before prediction begins. The method accounts for sequence information with post-processing: the numerical scores of possible outputs from an instance-based phoneme predictor are combined with phoneme transition probabilities in order to identify the most likely phoneme sequence.

3 A joint approach

By observing the strengths and weaknesses of previous approaches, we can create the following prioritized desiderata for any L2P system:

1. The phoneme predicted for a letter should be informed by the letter's context in the input word.
2. In addition to single letters, letter substrings should also be able to generate phonemes.
3. Phoneme sequence information should be included in the model.

Each of the previous approaches focuses on one or more of these items. Classification-based approaches such as the decision tree system (Black et al., 1998) and instance-based learning system (Bosch and Daelemans, 1998) take into account the letter's context (#1). By pairing letter substrings with phoneme substrings, the joint n-gram approach (Bisani and Ney, 2002) accounts for all three desiderata, but each operation is informed only by a limited amount of left context. The many-to-many classifier of Jiampojamarn et al. (2007) also attempts to account for all three, but it adheres

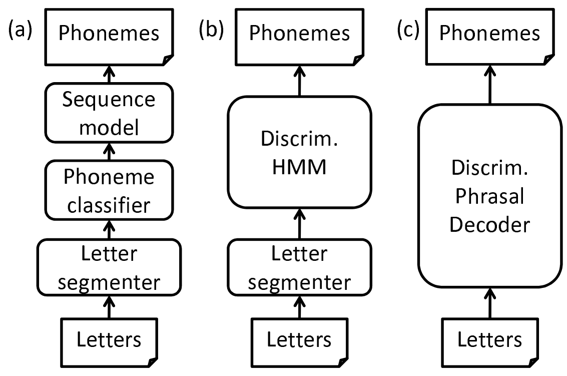

Figure 1: Collapsing the pipeline.

strictly to the pipeline approach illustrated in Figure 1a. It applies in succession three separately trained modules for input segmentation, phoneme prediction, and sequence modeling. Similarly, the CSInf approach modifies independent phoneme predictions (#1) in order to assemble them into a cohesive sequence (#3) in post-processing.

The pipeline approaches are undesirable for two reasons. First, when decisions are made in sequence, errors made early in the sequence can propagate forward and throw off later processing. Second, each module is trained independently, and the training methods are not aware of the tasks performed later in the pipeline. For example, optimal parameters for a phoneme prediction module may vary depending on whether or not the module will be used in conjunction with a phoneme sequence model.

We propose a joint approach to L2P conversion, grounded in dynamic programming and online discriminative training. We view L2P as a tagging task that can be performed with a discriminative learning method, such as the Perceptron HMM (Collins, 2002). The Perceptron HMM naturally handles phoneme prediction (#1) and sequence modeling (#3) simultaneously, as shown in Figure 1b. Furthermore, unlike a generative HMM, it can incorporate many overlapping source n-gram features to represent context. In order to complete the conversion from a pipeline approach to a joint approach, we fold our input segmentation step into the exact search framework by replacing a separate segmentation module (#2) with a monotone phrasal decoder (Zens and Ney, 2004). At this point all three of our desiderata are incorporated into a single module, as shown in Figure 1c.

Algorithm 1 Online discriminative training.

```
1: $\alpha = \vec{0}$
2: for $K$ iterations over training set do
3:     for all letter-phoneme sequence pairs $(x, y)$ in the training set do
4:         $\hat{y} = \arg\max_{y' \in Y} [\alpha \cdot \Phi(x, y')]$
5:         update weights $\alpha$ according to $\hat{y}$ and $y$
6:     end for
7: end for
8: return $\alpha$
```

Our joint approach to L2P lends itself to several refinements. We address an underfitting problem of the perceptron by replacing it with a more robust Margin Infused Relaxed Algorithm (MIRA), which adds an explicit notion of margin and takes into account the system's current n-best outputs. In addition, with all of our features collected under a unified framework, we are free to conjoin context features with sequence features to create a powerful linear-chain model (Sutton and McCallum, 2006).

4 Online discriminative training

In this section, we describe our entire L2P system. An outline of our discriminative training process is presented in Algorithm 1. An online process repeatedly finds the best output(s) given the current weights, and then updates those weights to make the model favor the correct answer over the incorrect ones.

The system consists of the following three main components, which we describe in detail in Sections 4.1, 4.2 and 4.3, respectively.

1. A scoring model, represented by a weighted linear combination of features ($\alpha \cdot \Phi(x, y)$).
2. A search for the highest scoring phoneme sequence for a given input word (Step 4).
3. An online update equation to move the model away from incorrect outputs and toward the correct output (Step 5).

4.1 Model

Given an input word x and an output phoneme sequence y, we define $\Phi(x, y)$ to be a feature vector representing the evidence for the sequence y found in x, and α to be a feature weight vector providing a weight for each component of $\Phi(x, y)$. We assume that both the input and output consist of m substrings, such that x_i generates y_i, $0 \leq i < m$. At training time, these substrings are taken from a many-to-many letter-to-phoneme alignment. At test time, input segmentation is handled by either a segmentation module or a phrasal decoder.

Table 1 shows our feature template that we include in $\Phi(x, y)$. We use only indicator features; each feature takes on a binary value indicating whether or not it is present in the current (x, y) pair. The context features express letter evidence found in the input string x, centered around the generator x_i of each y_i. The parameter c establishes the size of the context window. Note that we consider not only letter unigrams but all n-grams that fit within the window, which enables the model to assign phoneme preferences to contexts containing specific sequences, such as *ing* and *tion*. The transition features are HMM-like sequence features, which enforce cohesion on the output side. We include only first-order transition features, which look back to the previous phoneme substring generated by the system, because our early development experiments indicated that larger histories had little impact on performance; however, the number of previous substrings that are taken into account could be extended at a polynomial cost. Finally, the linear-chain features (Sutton and McCallum, 2006) associate the phoneme transitions between y_{i-1} and y_i with each n-gram surrounding x_i. This combination of sequence and context data provides the model with an additional degree of control.

context	x_{i-c}, y_i
	$\ldots$
	x_{i+c}, y_i
	$x_{i-c}x_{i-c+1}, y_i$
	$\ldots$
	$x_{i+c-1}x_{i+c}, y_i$
	$\ldots\ldots$
	$x_{i-c} \ldots x_{i+c}, y_i$
transition	y_{i-1}, y_i
linear chain	x_{i-c}, y_{i-1}, y_i
	$\ldots$
	x_{i+c}, y_{i-1}, y_i
	$x_{i-c}x_{i-c+1}, y_{i-1}, y_i$
	$\ldots$
	$x_{i+c-1}x_{i+c}, y_{i-1}, y_i$
	$\ldots\ldots$
	$x_{i-c} \ldots x_{i+c}, y_{i-1}, y_i$

Table 1: Feature template.

4.2 Search

Given the current feature weight vector α, we are interested in finding the highest-scoring phoneme sequence $\hat{y}$ in the set Y of all possible phoneme sequences. In the pipeline approach (Figure 1b), the input word is segmented into letter substrings by an instance-based classifier (Aha et al., 1991), which learns a letter segmentation model from many-to-many alignments (Jiampojamarn et al., 2007). The search for the best output sequence is then effectively a substring tagging problem, and we can compute the $\arg\max$ operation in line 4 of Algorithm 1 with the standard HMM Viterbi search algorithm.

In the joint approach (Figure 1c), we perform segmentation and L2P prediction simultaneously by applying the monotone search algorithm developed for statistical machine translation (Zens and Ney, 2004). Thanks to its ability to translate phrases (in our case, letter substrings), we can accomplish the $\arg\max$ operation without specifying an input segmentation in advance; the search enumerates all possible segmentations. Furthermore, the language model functionality of the decoder allows us to keep benefiting from the transition and linear-chain features, which are explicit in the previous HMM approach.

The search can be efficiently performed by the dynamic programming recurrence shown below. We define $Q(j, p)$ as the maximum score of the phoneme sequence ending with the phoneme p generated by the letter sequence $x_1 \ldots x_j$. Since we are no longer provided an input segmentation in advance, in this framework we view x as a sequence of J letters, as opposed to substrings. The phoneme p' is the phoneme produced in the previous step. The expression $\phi(x_{j'+1}^{j}, p', p)$ is a convenient way to express the subvector of our complete feature vector $\Phi(x, y)$ that describes the substring pair (x_i, y_{i-1}^{i}), where $x_i = x_{j'+1}^{j}$, $y_{i-1} = p'$ and $y_i = p$. The value N limits the size of the dynamically created substrings. We use $N = 2$, which reflects a similar limit in our many-to-many aligner. The special symbol \$ represents a starting phoneme or ending phoneme. The value in $Q(I+1, \$)$ is the score of highest scoring phoneme sequence corresponding to the input word. The actual sequence can be retrieved by backtracking through the table Q.

$$\begin{aligned} &Q(0, \$) = 0 \\ &Q(j, p) = \max_{\substack{p', p, \\ j-N \leq j' < j}} \{\alpha \cdot \phi(x^{j}_{j'+1}, p', p) + Q(j', p')\} \\ &Q(J+1, \$) = \max_{p'} \{\alpha \cdot \phi(\$, p', \$) + Q(J, p')\} \end{aligned}$$

4.3 Online update

We investigate two model updates to drive our online discriminative learning. The simple perceptron update requires only the system's current output, while MIRA allows us to take advantage of the system's current n-best outputs.

Perceptron

Learning a discriminative structure prediction model with a perceptron update was first proposed by Collins (2002). The perceptron update process is relatively simple, involving only vector addition. In line 5 of Algorithm 1, the weight vector α is updated according to the best output $\hat{y}$ under the current weights and the true output y in the training data. If $\hat{y} = y$, there is no update to the weights; otherwise, the weights are updated as follows:

$$\alpha = \alpha + \Phi(x, y) - \Phi(x, \hat{y}) \quad (1)$$

We iterate through the training data until the system performance drops on a held-out set. In a separable case, the perceptron will find an α such that:

$$\forall \hat{y} \in Y - \{y\} : \alpha \cdot \Phi(x, y) > \alpha \cdot \Phi(x, \hat{y}) \quad (2)$$

Since real-world data is not often separable, the average of all α values seen throughout training is used in place of the final α, as the average generalizes better to unseen data.

MIRA

In the perceptron training algorithm, no update is derived from a particular training example so long as the system is predicting the correct phoneme sequence. The perceptron has no notion of margin: a slim preference for the correct sequence is just as good as a clear preference. During development, we observed that this lead to underfitting the training examples; useful and consistent evidence was ignored because of the presence of stronger evidence in the same example. The MIRA update provides a principled method to resolve this problem.

The Margin Infused Relaxed Algorithm or MIRA (Crammer and Singer, 2003) updates the model based on the system's n-best output. It employs a margin update which can induce an update even when the 1-best answer is correct. It does so by finding a weight vector that separates incorrect sequences in the n-best list from the correct sequence by a variable width margin.

The update process finds the smallest change in the current weights so that the new weights will separate the correct answer from each incorrect answer by a margin determined by a structured loss function. The loss function describes the distance between an incorrect prediction and the correct one; that is, it quantifies just how wrong the proposed sequence is. This update process can be described as an optimization problem:

$$\begin{aligned} &\min_{\alpha_n} \parallel \alpha_n - \alpha_o \parallel \\ &\text{subject to } \forall \hat{y} \in Y_n : \\ &\alpha_n \cdot (\Phi(x, y) - \Phi(x, \hat{y})) \geq \ell(y, \hat{y}) \end{aligned} \quad (3)$$

where Y_n is a set of n-best outputs found under the current model, y is the correct answer, α_o is the current weight vector, α_n is the new weight vector, and $\ell(y, \hat{y})$ is the loss function.

Since our direct objective is to produce the correct phoneme sequence for a given word, the most intuitive way to define the loss function $\ell(y, \hat{y})$ is binary: 0 if $\hat{y} = y$, and 1 otherwise. We refer to this as *0-1* loss. Another possibility is to base the loss function on the phoneme error rate, calculated as the Levenshtein distance between y and $\hat{y}$. We can also compute a combined loss function as an equally-weighted linear combination of the *0-1* and phoneme loss functions.

MIRA training is similar to averaged perceptron training, but instead of finding the single best answer, we find the n-best answers (Y_n) and update weights according to Equation 3. To find the n-best answers, we modify the HMM and monotone search algorithms to keep track of the n-best phonemes at

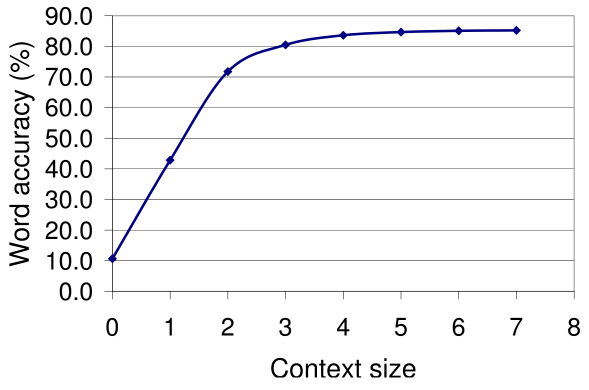

Figure 2: Perceptron update with different context size.

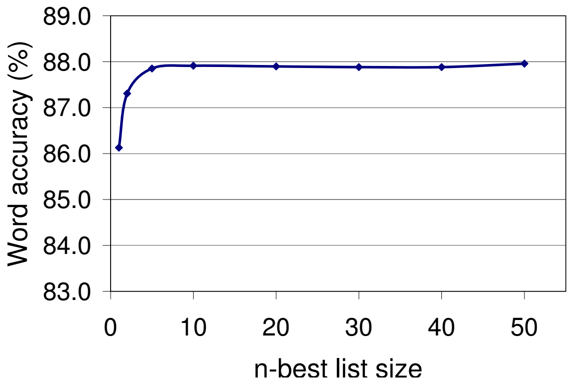

Figure 3: MIRA update with different size of n-best list.

each cell of the dynamic programming matrix. The optimization in Equation 3 is a standard quadratic programming problem that can be solved by using Hildreth's algorithm (Censor and Zenios, 1997). The details of our implementation of MIRA within the SVM^{light} framework (Joachims, 1999) are given in the Appendix A. Like the perceptron algorithm, MIRA returns the average of all weight vectors produced during learning.

5 Evaluation

We evaluated our approach on English, German and Dutch CELEX (Baayen et al., 1996), French Brulex, English Nettalk and English CMUDict data sets. Except for English CELEX, we used the data sets from the PRONALSYL letter-to-phoneme conversion challenge[1]. Each data set is divided into 10 folds: we used the first one for testing, and the rest for training. In all cases, we hold out 5% of our training data to determine when to stop perceptron or MIRA training. We ignored one-to-one alignments included in the PRONALSYL data sets, and instead induced many-to-many alignments using the method of Jiampojamarn et al. (2007).

Our English CELEX data set was extracted directly from the CELEX database. After removing duplicate words, phrases, and abbreviations, the data set contained 66,189 word-phoneme pairs, of which 10% was designated as the final test set, and the rest as the training set. We performed our development experiments on the latter part, and then used the final test set to compare the performance of our system to other results reported in the literature.

We report the system performance in terms of word accuracy, which rewards only completely correct phoneme sequences. Word accuracy is more demanding than phoneme accuracy, which considers the number of correct phonemes. We feel that word accuracy is a more appropriate error metric, given the quality of current L2P systems. Phoneme accuracy is not sensitive enough to detect improvements in highly accurate L2P systems: Black et al. (1998) report 90% phoneme accuracy is equivalent to approximately 60% word accuracy, while 99% phoneme accuracy corresponds to only 90% word accuracy.

5.1 Development Experiments

We began development with a zero-order Perceptron HMM with an external segmenter, which uses only the context features from Table 1. The zero-order Perceptron HMM is equivalent to training a multi-class perceptron to make independent substring-to-phoneme predictions; however, this framework allows us to easily extend to structured models. We investigate the effect of augmenting this baseline system in turn with larger context sizes, the MIRA update, joint segmentation, and finally sequence features. We report the impact of each contribution on our English CELEX development set.

Figure 2 shows the performance of our baseline L2P system with different context size values (c). Increasing the context size has a dramatic effect on accuracy, but the effect begins to level off for context sizes greater than 5. Henceforth, we report the

[1]Available at `http://www.pascal-network.org/Challenges/PRONALSYL/`. The results have not been announced.

	Perceptron	MIRA
Separate segmentation	84.5%	85.8%
Phrasal decoding	**86.6%**	**88.0%**

Table 2: Separate segmentation versus phrasal decoding in terms of word accuracy.

results with context size $c = 5$.

Figure 3 illustrates the effect of varying the size of n-best list in the MIRA update. $n = 1$ is equivalent to taking into account only the best answer, which does not address the underfitting problem. A large n-best list makes it difficult for the optimizer to separate the correct and incorrect answers, resulting in large updates at each step. We settle on $n = 10$ for the subsequent experiments.

The choice of MIRA's loss function has a minimal impact on performance, probably because our baseline system already has a very high phoneme accuracy. We employ the loss function that combines *0-1* and phoneme error rate, due to its marginal improvement over *0-1* loss on the development set.

Looking across columns in Table 2, we observe over 8% reduction in word error rate when the perceptron update is replaced with the MIRA update. Since the perceptron is a considerably simpler algorithm, we continue to report the results of both variants throughout this section.

Table 2 also shows the word accuracy of our system after adding the option to conduct joint segmentation through phrasal decoding. The 15% relative reduction in error rate in the second row demonstrates the utility of folding the segmentation step into the search. It also shows that the joint framework enables the system to reduce and compensate for errors that occur in a pipeline. This is particularly interesting because our separate instance-based segmenter is highly accurate, achieving 98% segmentation accuracy. Our experiments indicate that the application of joint segmentation recovers more than 60% of the available improvements, according to an upper bound determined by utilizing perfect segmentation.[2]

Table 3 illustrates the effect of our sequence features on both the perceptron and MIRA systems.

[2]Perfect with respect to our many-to-many alignment (Jiampojamarn et al., 2007), but not necessarily in any linguistic sense.

Feature	Perceptron	MIRA
zero order	86.6%	88.0%
+ 1^{st} order HMM	87.1%	88.3%
+ linear-chain	87.5%	89.3%
All features	**87.8%**	**89.4%**

Table 3: The effect of sequence features on the joint system in terms of word accuracy.

Replacing the zero-order HMM with the first-order HMM makes little difference by itself, but combined with the more powerful linear-chain features, it results in a relative error reduction of about 12%. In general, the linear-chain features make a much larger difference than the relatively simple transition features, which underscores the importance of using source-side context when assessing sequences of phonemes.

The results reported in Tables 2 and 3 were calculated using cross validation on the training part of the CELEX data set. With the exception of adding the 1^{st} order HMM, the differences between versions are statistically significant according to McNemar's test at 95% confidence level. On one CPU of AMD Opteron 2.2GHz with 6GB of installed memory, it takes approximately 32 hours to train the MIRA model with all features, compared to 12 hours for the zero-order model.

5.2 System Comparison

Table 4 shows the comparison between our approach and other systems on the evaluation data sets. We trained our system using n-gram context, transition, and linear-chain features. All parameters, including the size of n-best list, size of letter context, and the choice of loss functions, were established on the English CELEX development set, as presented in our previous experiments. With the exception of the system described in (Jiampojamarn et al., 2007), which we re-ran on our current test sets, the results of other systems are taken from the original papers. Although these comparisons are necessarily indirect due to different experimental settings, they strongly suggest that our system outperforms all previous published results on all data sets, in some case by large margins. When compared to the current state-of-the-art performance of each data set, the relative reductions in error rate range from 7% to 46%.

Corpus	MIRA	Perceptron	M-M HMM	Joint n-gram*	CSInf*	PbA*	CART*
Eng. CELEX	**90.51%**	88.44%	84.81%	76.3%	84.5%	-	-
Dutch CELEX	**95.32%**	95.13%	91.69%	-	94.5%	-	-
German CELEX	**93.61%**	92.84%	90.31%	92.5%	-	-	89.38%
Nettalk	**67.82%**	64.87%	59.32%	64.6%	-	65.35%	-
CMUDict	**71.99%**	71.03%	65.38%	-	-	-	57.80%
Brulex	**94.51%**	93.89%	89.77%	89.1%	-	-	-

Table 4: Word accuracy on the evaluated data sets. **MIRA**, **Perceptron**: our systems. **M-M HMM**: Many-to-Many HMM system (Jiampojamarn et al., 2007). **Joint n-gram**: Joint n-gram model (Demberg et al., 2007). **CSInf**: Constraint satisfaction inference (Bosch and Canisius, 2006). **PbA**: Pronunciation by Analogy (Marchand and Damper, 2006). **CART**: CART decision tree system (Black et al., 1998). The columns marked with * contain results reported in the literature. "-" indicates no reported results. We have underlined the best previously reported results.

6 Conclusion

We have presented a joint framework for letter-to-phoneme conversion, powered by online discriminative training. We introduced two methods to convert multi-letter substrings into phonemes: one relying on a separate segmenter, and the other incorporating a unified search that finds the best input segmentation while generating the output sequence. We investigated two online update algorithms: the perceptron, which is straightforward to implement, and MIRA, which boosts performance by avoiding underfitting. Our systems employ source n-gram features and linear-chain features, which substantially increase L2P accuracy. Our experimental results demonstrate the power of a joint approach based on online discriminative training with large feature sets. In all cases, our MIRA-based system advances the current state of the art by reducing the best reported error rate.

Appendix A. MIRA Implementation

We optimize the objective shown in Equation 3 using the SVM^{light} framework (Joachims, 1999), which provides the quadratic program solver shown in Equation 4.

$$\begin{aligned} &\min_{w,\xi} \tfrac{1}{2} \parallel w \parallel^2 + C \textstyle\sum_i \xi_i \\ &\text{subject to } \forall i, \\ &w \cdot t_i \geq rhs_i - \xi_i \end{aligned} \qquad (4)$$

In order to approximate a hard margin using the soft-margin optimizer of SVM^{light}, we assign a very large penalty value to C, thus making the use of any slack variables (ξ_i) prohibitively expensive. We define the vector w as the difference between the new and previous weights: $w = \alpha_n - \alpha_o$. We constrain w to mirror the constraints in Equation 3. Since each $\hat{y}$ in the n-best list (Y_n) needs a constraint based on its feature difference vector, we define a t_i for each:

$$\forall \hat{y} \in Y_n : t_i = \Phi(x,y) - \Phi(x,\hat{y})$$

Substituting that equation along with the inferred equation $a_n = a_o + w$ into our original MIRA constraints yields:

$$(\alpha_o + w) \cdot t_i \geq \ell(y,\hat{y})$$

Moving α_o to the right-hand-side to isolate $w \cdot t_i$ on the left, we get a set of mappings that implement MIRA in SVM^{light}'s optimizer:

w	$\alpha_n - \alpha_o$
t_i	$\Phi(x,y) - \Phi(x,\hat{y})$
rhs_i	$\ell(y,\hat{y}) - \alpha_o \cdot t_i$

The output of the SVM^{light} optimizer is an update vector w to be added to the current α_o.

Acknowledgements

This research was supported by the Alberta Ingenuity Fund, and the Natural Sciences and Engineering Research Council of Canada.

References

David W. Aha, Dennis Kibler, and Marc K. Albert. 1991. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Harald Baayen, Richard Piepenbrock, and Leon Gulikers. 1996. The CELEX2 lexical database. LDC96L14.

Maximilian Bisani and Hermann Ney. 2002. Investigations on joint-multigram models for grapheme-to-phoneme conversion. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 105–108.

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.

Antal Van Den Bosch and Walter Daelemans. 1998. Do not forget: Full memory in memory-based learning of word pronunciation. In *Proceedings of NeMLaP3/CoNLL98*, pages 195–204, Sydney, Australia.

Yair Censor and Stavros A. Zenios. 1997. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8, Morristown, NJ, USA.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991.

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.

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.

Thorsten Joachims. 1999. Making large-scale support vector machine learning practical. pages 169–184. MIT Press, Cambridge, MA, USA.

John Kominek and Alan W Black. 2006. Learning pronunciation dictionaries: Language complexity and word selection strategies. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 232–239, New York City, USA.

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.

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*.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

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.