INTERSPEECH 2009 BRIGHTON

Online Discriminative Training for Grapheme-to-Phoneme Conversion

Sittichai Jiampojamarn, Grzegorz Kondrak

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

Abstract

We present an online discriminative training approach to grapheme-to-phoneme (g2p) conversion. We employ a many-to-many alignment between graphemes and phonemes, which overcomes the limitations of widely used one-to-one alignments. The discriminative structure-prediction model incorporates input segmentation, phoneme prediction, and sequence modeling in a unified dynamic programming framework. The learning model is able to capture both local context features in inputs, as well as non-local dependency features in sequence outputs. Experimental results show that our system surpasses the state-of-the-art on several data sets.

Index Terms: grapheme-to-phoneme conversion, speech synthesis, discriminative training

1. Introduction

The objective of grapheme-to-phoneme (g2p) conversion is to generate phonemes that correspond to a given written word. The phonemes represent the standard pronunciation of the word. Many data-driven techniques have been proposed for the task. The systems can generally be divided into two categories: those that consider the task as a multi-class classification problem, and those that treat it as a sequence prediction problem. In either case, training data generally contain word-phoneme pairs in which alignments between graphemes and phonemes are not present. Most supervised learning algorithms require these hidden structures to be discovered before they can learn phoneme generation models.

We present a phoneme generation model based on online discriminative training that incorporates many-to-many alignments. We define our many-to-many alignment algorithm in Section 2. The discriminative approach described in Section 3 allows us to utilize a large number of features, including the context features used in the classification approaches, and the output sequence features that aid our system to generate cohesive phoneme sequence outputs. In Section 4, we evaluate our approach using several data sets and compare the results to the state-of-the art joint-sequence model [1]. Our system reduces the relative error rate by as much as 40% in comparison with the best reported results.

2. Grapheme-to-phoneme alignment

Generally, in the g2p task, training data are available in the form of word-pronunciation pairs. Each pair is composed of a sequence of graphemes and their corresponding phonemes, which are not aligned; there is no explicit information indicating individual grapheme-phoneme relationships. While the relationships are hidden in the training data, humans naturally have an intuition of grapheme-phoneme relationships given a word-pronunciation pair. To simplify the task, these grapheme-phoneme relationships called "alignments" must be discovered, so that phoneme generation models can infer the production between each grapheme (or substring of graphemes) in the input word and phoneme (or substring of phonemes) in the output. For example, the word *phoenix*, pronounced [f i n ɪ k s], can be aligned as follows:

ph	*oe*	*n*	*i*	*x*
f	i	n	ɪ	ks

2.1. One-to-one alignment

In the earliest systems, graphemes and phonemes were aligned manually. Nowadays, the alignments are usually constructed in an automatic fashion, using unsupervised Expectation Maximization (EM) algorithms. Previous systems generally assumed one-to-one alignments for simplicity. They require each grapheme to align to a single phoneme or to a null phoneme (ϵ). Even though these automatic generated alignments are not always identical to human-generated alignments, they provide enough information for training a phoneme generation model. A one-to-one alignment of the word *phoenix* is:

p	*h*	*o*	*e*	*n*	*i*	*x*	-
f	-	-	i	n	ɪ	k	s

Although the one-to-one alignment assumption maintains the simplicity of the process, it creates at least two problems. The first problem occurs when two graphemes should be aligned with one phoneme; for example, *ph* - [f]. Under the one-to-one assumptions, a null phoneme has to be aligned with one of the graphemes, which can potentially confuse the generation model. The second problem occurs when two phonemes should be aligned to one grapheme; for example, *x* - [ks]. One possible solution is to align a null grapheme with one of the phonemes (as in the example above), which introduces complexities into the generation phase. Another possible solution is to create a new "super-phoneme" by merging the two phonemes in question, which requires an expert to construct a new phoneme list.

2.2. Many-to-many alignment

We employ a many-to-many alignment algorithm for the g2p conversion task, which is similar to the forward-backward algorithm proposed in [2]. The training process follows the expectation maximization paradigm. In the expectation step (Algorithm 1), partial counts of possible grapheme-phoneme mappings, γ, are collected from word-pronunciation pairs (x^T, y^V) using the forward-backward algorithms to estimate the forward probability α and backward probability β. In the algorithm,

Algorithm 1: Many-to-many alignment

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

$\alpha :=$ *Forward-many-to-many* $(x^T, y^V, maxX, maxY)$
$\beta :=$ *Backward-many-to-many* $(x^T, y^V, maxX, maxY)$

if $(\alpha_{T,V} = 0)$ **then**
  return
for $t = 0...T$ **do**
  for $v = 0...V$ **do**
    if $(t > 0)$ **then**
      for $i = 1...maxX$ **st** $t - i \geq 0$ **do**

$$\gamma(x^t_{t-i+1}, \epsilon) += \frac{\alpha_{t-i,v}\delta(x^t_{t-i+1}, \epsilon)\beta_{t,v}}{\alpha_{T,V}}$$

    if $(v > 0 \wedge t > 0)$ **then**
      for $i = 1...maxX$ **st** $t - i \geq 0$ **do**
        for $j = 1...maxY$ **st** $v - j \geq 0$ **do**

$$\gamma(x^t_{t-i+1}, y^v_{v-j+1}) += \frac{\alpha_{t-i,v-j}\delta(x^t_{t-i+1}, y^v_{v-j+1})\beta_{t,v}}{\alpha_{T,V}}$$

we allow a substring of graphemes x to be aligned with a null phoneme (ϵ) or a substring of phonemes y. The size of x ranges from 1 to $maxX$, while the size of y ranges from 0 to $maxY$. In the maximization step, the counts γ are normalized to the grapheme-phoneme mapping probabilities δ using the conditional probability distribution. The forward algorithm is shown in Algorithm 2, and the backward algorithm is an analogue of the forward algorithm. In the backward algorithm, we sum all paths from the end of the string pair to the beginning of the string pair (x^T, y^V). The EM process iteratively trains over the word-pronunciation pairs until the mapping probability δ converges. Finally, many-to-many alignments are produced using the Viterbi algorithm which finds the most likely alignment path based on the learned probabilities.

Many-to-many alignments overcome the one-to-one mapping limitations by providing more accurate grapheme-to-phoneme alignments for a phoneme generation model. However, since the alignments are not restricted to a single grapheme token, the exact segmentation of the input word has to be discovered at test time. One possible solution is to greedily merge graphemes to form substrings that occurred in the alignments during training. Another solution is to use a grapheme chunking model [3] which is trained on the generated alignments. In both approaches, the errors committed during the grapheme chunking phase are carried forward to the phoneme generation model that is unlikely to correct them. Instead, our online discriminative training algorithm directly incorporates the many-to-many alignments by using a monotone phrasal decoder [4]. The phoneme generation model simultaneously finds the most likely phoneme sequence outputs and input structures (grapheme chunks) given input words.

In contrast with the estimation model of [1], our many-to-many alignment algorithm is based on a unigram estimation without evidence trimming and smoothing. In addition, our approach allows for nulls on the phoneme side, which results in more natural alignments (e.g., in English words ending with letter e). Our algorithm is less complex than the estimation model of the joint n-gram model, yet powerful enough to guide the g2p generation model.

3. Phoneme generation

Once the alignments are discovered in the training data, we can use them to explicitly express grapheme-to-phoneme productions. The grapheme-to-phoneme conversion task can be viewed either as a multi-class classification problem, or as a sequence prediction problem. In the multi-class classification problem, each sub-phoneme output is generated directly from the focus grapheme (or substring of graphemes) and its context (surrounding graphemes) without considering the sequence of generated phonemes. Many classification algorithms have been proposed for the task including neural networks [5], instance-based learning [6], and decision trees [7]. These methods leverage the structure of the input but ignore any structure of the output.

Algorithm 2: Forward-many-to-many

Input: $(x^T, y^V, maxX, maxY)$
Output: α

$\alpha_{0,0} := 1$
for $t = 0...T$ **do**
  for $v = 0...V$ **do**
    if $(t > 0 \vee v > 0)$ **then**
      $\alpha_{t,v} = 0$
    if $(t > 0)$ **then**
      for $i = 1...maxX$ **st** $t - i \geq 0$ **do**
        $\alpha_{t,v} += \delta(x^t_{t-i+1}, \epsilon)\alpha_{t-i,v}$
    if $(v > 0 \wedge t > 0)$ **then**
      for $i = 1...maxX$ **st** $t - i \geq 0$ **do**
        for $j = 1...maxY$ **st** $v - j \geq 0$ **do**
          $\alpha_{t,v} += \delta(x^t_{t-i+1}, y^v_{v-j+1})\alpha_{t-i,v-j}$

A natural way to view grapheme-to-phoneme conversion is to consider it as a sequence modeling or tagging task. However, a standard supervised Hidden Markov Model (HMM) for the task proposed by [8] achieves poor results, which are mainly caused by the fact that the HMM framework lacks the capability of using the grapheme context information directly. These features have been shown to be important in classification-based approaches.

Joint n-gram models [9, 10] achieve good g2p results by training the models on grapheme-phoneme substring pairs, so that sequence information in both grapheme and phoneme sides directly contributes to the learning models. [1] improve the joint n-gram model with a new estimation algorithm, called "joint-sequence model". Similarly, Pronunciation by Analogy (PbA) [11] also operates on substrings of graphemes and phonemes. The method finds the least number of segmented grapheme sequences in the training examples that can form the input word, and produces the phoneme sequence that corresponds to the path with the highest score. The constraint satisfaction inference (CSInf) approach [12] is a hybrid method that combines a local classifier and a sequence prediction model. It improves the performance of the instance-based classification [13] by predicting a trigram of phonemes for each grapheme. The output is the phoneme sequence that satisfies the most of unigram, bigram, and trigram agreement constraints. Similarly, a post-processing language model [3] also improves the accuracy of a local classification model. However, all these methods are examples of a pipeline approach which encounters the error propagation problem.

3.1. Online discriminative training

The online discriminative training approach to the g2p task can include an arbitrary number of features to express relationships between grapheme sequence inputs and phoneme sequence outputs. The phrasal decoder [4] simultaneously finds most likely

Algorithm 3: Online discriminative training.

Input: Training examples x and y
Output: ψ

$\psi := \vec{0}$

for K *iterations over training set* **do**
  forall *word-phoneme pairs* (x, y) *in training set* **do**
    $\hat{Y} = \arg\max_{y'}[\psi \cdot \Phi(x, y')]$
    update weights ψ according to $\hat{Y}$ and y
return ψ

sequence of phoneme outputs and input structures.

We outline the process in Algorithm 3. The model is based on the averaged Perceptron algorithm [14], but differs in the feature representation, search, and update method. The training process repeatedly finds the n-best outputs $\hat{Y}$ given the current weights ψ, and updates the weights to make the model favors the correct answer over the incorrect ones. The number of iterations K is determined on a development set. There are three components in the algorithm: (1) the feature vector $\Phi(x, y)$ that describes evidence for the sequence output y found in x, (2) the $\arg\max$ operation that finds the n-best sequence outputs $\hat{Y}$ given the current weight vector ψ and feature vector $\Phi(x, y)$, and (3) the update method that optimizes the weight vector ψ. Unlike the averaged Perceptron model, our $\arg\max$ operation finds the n-best outputs instead of the most likely output to optimize the weight vector ψ.

In the first component, we define our feature vector $\Phi(x, y)$ to include (1) context features, (2) transition features and (3) linear-chain features. These features are indicator features representing whether or not they are present in the current (x, y). We assume that both input and output consist of m substrings, such that x_i generates y_i. At training time, these substrings can be taken from our many-to-many alignments. At test time, input segmentations are handled by the phrasal decoder. The context features express grapheme evidence surrounding the generator x_i of each y_i. We consider all n-grams that fit within the context window of size c. The transition features are HMM-like sequence features that allow the model to express the cohesion of phoneme sequence outputs. In the first Markov order model, these features are basically bi-grams of sequence output $(y_{i-1}y_i)$. Finally, the linear-chain features are those widely used in Conditional Random Fields (CRFs) [15] that associate the phoneme transition features $y_{i-1}y_i$ and grapheme n-grams. Each context feature has a corresponding linear-chain feature.

Our $\arg\max$ operation is implemented using a phrasal decoder [4], which is able to find the sequence output y without specifying an input segmentation in advance. The search algorithm uses dynamic programming recurrence equations that enumerate all possible input segmentations:

$$Q(0, \$) = 0$$
$$Q(j, p) = \max_{\substack{p', p, \\ j-N \le j' < j}} \{\psi \cdot \phi(x_{j'+1}^{j}, p', p) + Q(j', p')\}$$
$$Q(J+1, \$) = \max_{p'} \{\psi \cdot \phi(\$, p', \$) + Q(J, p')\}$$

The Q table keeps the n-best scores of the phoneme sequence ending with p generated by the grapheme sub-sequence $x_1 \ldots x_j$ of the input word x consisting of J graphemes. The term $\phi(x_{j'+1}^{j}, p', p)$ is a convenient way to express the sub-feature vector of $\Phi(x, y)$, where p' denotes the generated phoneme in the previous step. N is the maximum size of the input substring that can generate an output (phoneme) substring.

Finally, our last component, the update method is based on the Margin Infused Relaxed Algorithm (MIRA) [16]. It updates the feature weights using the system's n-best outputs and the true output y in the training data. The update process finds the smallest change in the current weights ψ such that the new weights separate the correct answer y from each incorrect answer $\hat{y} \in \hat{Y}$ by a margin of at least as much as a structured loss between the correct and incorrect answer. We can describe the update process as an optimization problem:

$$\begin{aligned} &\min_{\psi_n} \parallel \psi_n - \psi_o \parallel \\ &\text{subject to } \forall \hat{y} \in \hat{Y} : \\ &\psi_n \cdot (\Phi(x, y) - \Phi(x, \hat{y})) \ge \ell(y, \hat{y}) \end{aligned} \quad (1)$$

We define our structured loss function to be 0 if $\hat{y} = y$; otherwise we set it to $1 + d$, where d is the Levenshtein distance between y and $\hat{y}$. This optimization equation is a standard quadratic programming problem that can be solved by using Hildreth's algorithm [17]. In practice, we employ the optimization function implemented in the SVM^{light} framework [18] to solve this optimization problem.

4. Experiments and results

We evaluated several components of our model in terms of performance on our CELEX development set [19]. First, our full system outperforms the model that includes only the context features. This indicates that the transition and linear-chain features are able to capture cohesion in the phoneme sequence outputs. Second, the phrasal decoder yields better results than the letter chunking model [3], which segments the words at test time in a separate preprocessing step. This demonstrates the value of the phrasal decoder which incorporates the many-to-many alignments with no separate segmentation steps. Finally, when the MIRA update algorithm replaces a simpler averaged Perceptron algorithm the performance improves again. The Perceptron update algorithm has no notion of margin separation: it greedily updates the feature weights toward the correct answers. The result indicates that the MIRA update is more appropriate for the g2p task.

We conducted the final comparison experiments on several English and French data sets which were used in [1]. With the exception of OALD [1] and English CELEX [20], the remaining data sets (NETtalk, Beep, CMUdict, and Brulex) were obtained from the Pascal Letter-to-Phoneme Conversion Challenge[2]. In data preparation, we followed the conventions described in [1], excluding homographs, one-letter words, punctuation, phrases and abbreviations. We used 10% of each data set for testing, 85% for training, and 5% as our held-out set for setting the model's parameters. We report the results in term of word accuracy, which considers only completely correct phoneme sequences. In other words, it is simply one minus word error rate (WER).

Table 1 shows the comparison between our approach and the joint-sequence approach [1], which was shown to outperform previous systems, including the joint n-gram [10], Pronunciation by Analogy [11], M-M HMM [3], CSInf [12], and decision trees [7]. Although we were not able to use exactly the

[1] ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/

[2] http://pascallin.ecs.soton.ac.uk/Challenges/PRONALSYL/Datasets/

Data set	This work	Joint-sequence
Eng. CELEX	91.0%	88.6%
Beep	81.6%	79.9%
OALD	89.6%	82.5%
CMUDict	72.5%	75.5%
NETtalk	68.1%	69.0%
Brulex	94.6%	93.8%

Table 1: Word accuracy results of our system on several data sets compared to the **joint-sequence** results reported in [1].

same splits of the data sets for training and testing, we strived to faithfully replicate the evaluated sets with respect to the data sources and sizes. Our system outperforms the joint-sequence model on four out of six data sets, with the relative error reduction ranging from 8% to 40%. This also implies better performance than all other systems reported in the literature for those data sets.

On the CMUdict and NETtalk data sets, our system's results are slightly lower than the results reported by [1]. However, it is worth noting that both sets appear to be of inferior quality in comparison to the remaining data sets, as evidenced by generally lower accuracy numbers. CMUdict is partly composed of pronunciations extracted from speech transcriptions, and partly from automatically generated examples, while NETtalk annotations were shown to be inconsistent in [21].

Our system is fast in terms of phoneme sequence generation. Using a single CPU of AMD Opteron 2.2GHz with 6GB of installed memory, it takes approximately 60 hours to train on the CMUDict data set (95K words), and approximately 15 minutes to produce outputs for 11K words.

5. Conclusion

We presented an online discriminative training approach to g2p, which makes it possible to incorporate a large number of features including grapheme context, phoneme transitions, and linear-chain features. Our many-to-many alignments overcome the limitations of one-to-one alignments. The phrasal-based decoder smoothly integrates many-to-many alignments into the model, eliminating the need for a separate grapheme sequence segmentation step. The MIRA update algorithm is very effective in updating feature weights for distinguishing between correct and incorrect structured outputs. Our experimental results on several data sets compare favorably to the state of the art results of the joint-sequence approach [1], and indicate the success of our approach in terms of alignment quality and phoneme generation accuracy.

6. Acknowledgements

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

7. References

[1] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434–451, 2008.

[2] T. H. Sherif, "Substring-based transliteration," Master's thesis, University of Alberta, 2007.

[3] S. Jiampojamarn, G. Kondrak, and T. Sherif, "Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion," in *HLT-NAACL 2007: Main Proceedings*, 2007, pp. 372–379.

[4] R. Zens and H. Ney, "Improvements in phrase-based statistical machine translation," in *HLT-NAACL 2004: Main Proceedings*, 2004, pp. 257–264.

[5] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce English text," in *Complex Systems*, 1987, pp. 1:145–168.

[6] W. Daelemans and A. V. D. Bosch, "Language-independent data-oriented grapheme-to-phoneme conversion," in *Progress in Speech Synthesis*, 1997, pp. 77–89.

[7] A. W. Black, K. Lenzo, and V. Pagel, "Issues in building general letter to sound rules," in *The Third ESCA Workshop in Speech Synthesis*, 1998, pp. 77–80.

[8] P. Taylor, "Hidden Markov Models for grapheme to phoneme conversion," in *The 9th European Conference on Speech Communication and Technology*, 2005.

[9] M. Bisani and H. Ney, "Investigations on joint-multigram models for grapheme-to-phoneme conversion," in *The 7th International Conference on Spoken Language Processing*, 2002, pp. 105–108.

[10] S. F. Chen, "Conditional and joint models for grapheme-to-phoneme conversion," in *Eurospeech-2003*, 2003.

[11] Y. Marchand and R. I. Damper, "A multistrategy approach to improving pronunciation by analogy," *Computational Linguistics*, vol. 26, no. 2, pp. 195–219, 2000.

[12] A. V. D. Bosch and S. Canisius, "Improved morpho-phonological sequence processing with constraint satisfaction inference," *The Eighth Meeting of the ACL Special Interest Group in Computational Phonology, SIGPHON*, pp. 41–49, 2006.

[13] A. V. D. Bosch and W. Daelemans, "Do not forget: Full memory in memory-based learning of word pronunciation." in *NeMLaP3/CoNLL98*, 1998, pp. 195–204.

[14] M. Collins, "Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms," in *EMNLP '02: the ACL-02 conference on Empirical methods in natural language processing*, 2002, pp. 1–8.

[15] C. Sutton and A. McCallum, "An introduction to conditional random fields for relational learning," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. MIT Press, 2006.

[16] K. Crammer and Y. Singer, "Ultraconservative online algorithms for multiclass problems," *The Journal of Machine Learning Research*, vol. 3, pp. 951–991, 2003.

[17] Y. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1997.

[18] T. Joachims, "Making large-scale support vector machine learning practical." Cambridge, MA: MIT Press, 1999, pp. 169–184.

[19] S. Jiampojamarn, C. Cherry, and G. Kondrak, "Joint processing and discriminative training for letter-to-phoneme conversion," in *Proceedings of ACL-08: HLT*, 2008, pp. 905–913.

[20] H. Baayen, R. Piepenbrock, and L. Gulikers, "The CELEX2 lexical database," LDC96L14, 1996.

[21] S. Bartlett, G. Kondrak, and C. Cherry, "Automatic syllabification with structured SVMs for letter-to-phoneme conversion," in *Proceedings of ACL-08: HLT*, 2008, pp. 568–576.