Improving WFST-based G2P Conversion with Alignment Constraints and RNNLM N-best Rescoring

Josef R. Novak[1], *Paul R. Dixon*[2], *Nobuaki Minematsu*[1], *Keikichi Hirose*[1], *Chiori Hori*[2], *Hideki Kashioka*[2]

[1]Graduate School of Information Science and Technology, The University of Tokyo, Japan
[2]National Institute of Communication Technology, Kyoto, Japan

`{novakj,mine,hirose}@gavo.t.u-tokyo.ac.jp, {paul.dixon,chori.hori,hideki.kashioka}@nict.go.jp`

Abstract

This work introduces a modified WFST-based multiple to multiple EM-driven alignment algorithm for Grapheme-to-Phoneme (G2P) conversion, and preliminary experimental results applying a Recurrent Neural Network Language Model (RNNLM) as an N-best rescoring mechanism for G2P conversion. The alignment algorithm leverages the WFST framework and introduces several simple structural constraints which yield a small but consistent improvement in Word Accuracy (WA) on a selection of standard baselines. The RNNLM rescoring further extends these gains and achieves state-of-the-art performance on four standard G2P datasets. The system is also shown to be significantly faster than existing solutions. Finally, the complete WFST-based G2P framework is provided as an open-source toolkit.

Index Terms: G2P, Alignment, RNNLM, WFST

1. Introduction

Grapheme-to-Phoneme (G2P) conversion is an important topic related to both Automatic Speech Recognition (ASR) and Text-To-Speech synthesis (TTS), and which also plays a role in Spoken Dialog Systems (SDS) and Natural Language Processing (NLP).

The primary goal of G2P conversion is to accurately predict the pronunciation of a novel word given only the orthography. For languages like French and English, designing robust systems has proven to be a surprisingly difficult challenge due inconsistencies and competing rules.

One of the most popular approaches to this problem is the joint sequence model [1]. In the simplest case this amounts to training an N-gram model from an aligned corpus of G⇔P sequences. Bisani and Ney [1] proposed a method for joint sequence modeling that utilizes a discounted Expectation Maximization (EM) scheme to simultaneously align and segment a pronunciation dictionary into a successful G2P model. This approach currently represents the gold-standard in joint sequence approaches to G2P conversion. More recently, Jiampojamarn [2] proposed an approach combining a multiple-to-multiple (M2M) sequence alignment algorithm [3] with an online discriminative training framework that consistently outperforms the joint sequence model. Many related methods have been proposed in past, and the reader is referred to [1] for a comprehensive review.

This work employs a modified, loosely coupled approach to joint sequence modeling which introduces several minor structural improvements to the M2M alignment algorithm of [3], and utilizes a standard joint N-gram model. The Weighted Finite-State Transducer (WFST) framework is utilized throughout, following the approach outlined in [4, 5].

This work also investigates N-best rescoring with a Recurrent Neural Network Language Model (RNNLM) [6], which consistently advances the state-of-the-art in G2P conversion across a range of standard test sets. In addition to the modest accuracy improvements, the system is shown to be much more efficient. Finally, the proposed system is released as an open source software project [7].

The remainder of the paper is structured as follows. Section 2 describes the alignment sub-problem in detail, and proposes an improved m-to-one/one-to-m alignment algorithm leveraging the WFST framework. Section 3 summarizes the joint N-gram modeling approach. Section 4 describes the WFST-based G2P decoding framework. Section 5 details the RNN-LM based N-best rescoring approach. Section 6 presents experimental results and related analysis. Section 7 summarizes the results and discusses future work.

2. Grapheme-to-Phoneme Alignment

Yianolos and Ristad [8] described the theory and implementation of a one-to-one stochastic transducer which could be trained to model string edit distance using the EM framework. Jiampojamarn [3, 2] extended this to model multiple-to-multiple alignments and close analysis of their reference implementation [9] indicated several potential areas for improvement. In the proposed implementation we cast the problem in the WFST framework as described in Algorithm 1. We make three modifications to the

Algorithm 1: Outline of the *proposed* Multiple-to-Multiple EM alignment algorithm.

```
Input: $x^T$, $y^V$, $maxX$, $maxY$, $delX$, $delY$
Output: $\gamma$, AlignedLattices
foreach sequence pair ($x^T$, $y^V$) do
    InitM2MFSA($x^T$, $y^V$, $maxX$, $maxY$, $delX$, $delY$)
foreach sequence pair ($x^T$, $y^V$) do
    Exp-M2M($x^T$, $y^V$, $maxX$, $maxY$, $\gamma$)
Max-M2M($\gamma$)
```

initialization function compared to [3, 2]. ❶ Only m-to-one and one-to-m arcs are trained, in contrast to [3] where m-to-m arcs are trained but two-to-two arcs are excluded by default during decoding, and [2] where only one-to-m arcs are allowed. The EM-training tends to heavily favor m-to-m links, however when using 1-best alignments to train a joint N-gram model, the larger chunks result in lower precision. In practice the m-to-one, one-to-m constraints appear to achieve a better balance and result in higher WA scores, while the smaller number of arcs leads to shorter training times. ❷ A joint WFSA alignment lattice is built from each sequence pair using the log semiring, input/output labels are encoded as joint labels, and any arcs that are not on a valid path are deleted. The latter step is important because unless both *delX* and *delY* are true, unconnected arcs may be generated. ❸ All remaining arcs, including deletion/substitution arcs are initialized to and constrained to maintain a non-zero weight. This helps to ensure that EM training produces valid estimates for all possible transitions.

Once the corpus of sequences is transformed into a set of FSA alignment lattices EM training proceeds straightforwardly. No further book-keeping or string manipulation is required, which simplifies and speeds up subsequent expectation and maximization steps in comparison to [9]. The WFST-based Expectation step is described in Algorithm 2, where the ShortestDistance in the log semiring is equivalent to the forward algorithm. The maximization step is described in Algorithm 3, and simply renormalizes the result of the E-step. Once the EM process terminates, the resulting γ table is applied to the alignment lattices. A shortest path algorithm can then be used to produce the 1-best alignment for each entry in the training corpus, or the lattices can be used directly to train a joint N-gram model. Finally a length-normalization penalty is applied during decoding. This is defined as the length of the longest subsequence for a given arc, similar to [9].

Algorithm 2: Outline of the proposed WFST-based Expectation algorithm. Here $a.w$ refers to the “arc weight”, $a.ns$ refers to the “next state” and $a.il$ refers to an “input label”.

```
Input: $AlignedLattices$
Output: $\gamma$, $total$
foreach FSA alignment lattice F do
    $\alpha$ = ShortestDistance($F$)
    $\beta$ = ShortestDistance($F^R$)
    foreach State q in F do
        foreach Arc a in q do
            $val = ((\alpha[q] \otimes a.w) \otimes \beta[a.ns]) \oslash \beta[0]$
            $\gamma[a.il] = \gamma[a.il] \oplus val$
            $total = total \oplus val$
```

Algorithm 3: Outline of the proposed WFST-based Maximization routine.

```
Input: $\gamma$, $total$
Output: AlignedLattices
foreach Label pair p in $\gamma$ do
    $\gamma\_new[p.il] = p.w/total$ $\gamma[p.il] = 0$
foreach FSA alignment lattice F do
    foreach State q in F do
        foreach Arc a in q do
            $a.w = \gamma\_new[a.il]$
```

3. WFST-based Joint N-gram model

The proposed toolkit [7] utilizes a simple joint N-gram formulation which is trained on an aligned pronunciation lexicon following [5]. The training corpus is generated directly from the shortest path through each joint WFSA alignment lattice:

```
<s> a}x b}b a}@ c|k}k </s>
<s> a}x b}b a}@ f}f t}t </s>
```

and this is used to train a statistical language model in the standard way. The complete training procedure is outlined below.

1. Convert aligned sequence pairs, $(g_1, g_2, ..., g_n)$, $(p_1, p_2, ..., p_n)$ to sequences of aligned pairs, $(g_1{:}p_1, g_2{:}p_2, ..., g_n{:}p_n)$.
2. Generate an N-gram model from (1).
3. Convert the N-gram model to a WFST with grapheme input and phoneme output labels.

4. WFST-based G2P Decoding

Generating a pronunciation for a new word is achieved by compiling the word into an FSA and composing it with the pronunciation model. In the case of an m-to-one/one-to-m model, a list of grapheme subsequences generated during alignment are utilized to

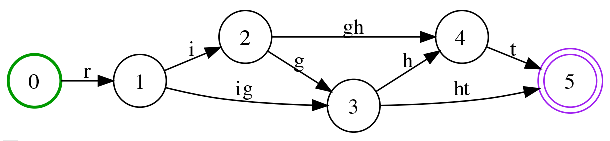

Figure 1: A multiple-to-multiple alignment example. Arc choices like, $i \rightarrow g \rightarrow h$ versus $i \rightarrow gh$ will be determined at runtime.

build the input FSA. The maximum size of the subsequences is determined by the alignment parameters. An example of such a test FSA for the word 'right', is depicted in Figure 1.

The decoding procedure is defined by a cascade of WFST operations and a final formatting step described in Equation 1.

$$H_{list} = ShortestPath(Det(Proj_o(W \circ M))) \quad (1)$$

where H_{list} refers to the weighted list of pronunciation hypotheses. W refers to the FSA constructed from the input test word, and M refers to the WFST constructed from the joint G2P N-gram model. The $\circ$ operator denotes composition, the $Proj_o$ operator indicates that the output labels only are projected, thereby creating an FSA containing just the hypothesized phoneme sequences. The Det operator refers to an optional determinization step, and $ShortestPath$ denotes the global shortest path, or N-shortest paths. Here the best hypothesis is just the single shortest path through the composition result. A major advantage of this decoupled approach is that each component of the final model can usually be trained in a matter of minutes.

5. N-best rescoring with RNNLM

Recurrent Neural Network Language Models have recently enjoyed a resurgence in popularity, achieving notable improvements in the area of N-best rescoring of ASR lattices [6]. In [6] the authors make the argument that the neural network language models have memory, are able to cluster similar words and histories, and provide complementary information to standard back-off models. This work has recently been made available as an open source toolkit, RNNLM [10]. This suggested that the approach might also be applicable to our G2P system.

As an additional set of experiments we explored extending the proposed WFST-based G2P system with RNNLM-based N-best rescoring. In order to train each RNNLM the aligned corpus of joint G$\Leftrightarrow$P sequences was utilized as input. The RNNLMs required a considerable amount of parameter tuning and experimentation in order to find the right number of hidden-layers, order and direct connections. A crude form of interpolation was utilized whereby the optimal N produced by each baseline model was also tuned prior to the testing phase. RNNLM rescoring produced consistent improvements to the state-of-the-art on all evaluated test sets.

6. Experiments and Results

A range of experiments were conducted exploring the impact of the modified alignment algorithm and the impact of RNNLM-based N-best rescoring. Specifically the NETtalk-15k, NETTalk-19k, CMUdict and OALD tests were replicated exactly as in [1, 2]. We investigate the relative performance of the m2m-alignment algorithm from [3], our proposed modifications, and RNNLM rescoring. For the alignment algorithms, parameters were set as described in Table 2. For the N-gram models N was set to 7 for the NETtalk evaluations and 11 for the larger CMUdict and OALD evaluations. Kneser-Ney smoothing was used for all N-gram models.

Parameter tuning for the RNNLMs was somewhat involved. First each training set was randomly partitioned into a 96.5% training and 3.5% validation set - this corresponded to roughly 500 entries in the case of the NETTalk-15k dataset. Parameters for all models were set as follows. Hidden layers: 300, direct-order: 4, direct-connections:2M, bptt: 10, bptt-block: 10. Finally the optimal N was selected by similarly tuning on a separate 5% held-out test set. For NETTalk N=10, CMUdict N=3 and for OALD N=5. This parameter can be thought of as a crude interpolation weight. WA results for the RNNLM rescoring were computed by averaging over 5 separate randomized trials.

Results for the four different test sets including previous reported results from [1, 2], and three variations of the proposed setup are described in Table 1. WA scores are computed as C/N where C represents the total number of correct hypotheses and N the number of unique words in the test set. Test words with more than one pronunciation variant were counted only once, and counted correct if the G2P hypothesis matched one of the variants. The improvements to the alignment algorithm clearly yield a small but consistent increase versus [3]. This may be attributed to ❶ elimination of m-to-m arcs during training, whereas in [9] m-to-m arcs are trained but ignored during decoding and ❷ the constraint enforcing non-zero weights for deletion arcs. For the NETTalk experiments the improved alignment algorithm alone is enough to produce consistent improvements over [1]. Adding the RNNLM rescoring further produces improvements to the state-of-the-art in all evaluations.

One of the major advantages of the proposed system is its speed. Table 3 compares total training times for the various systems. The proposed align-

Table 1: Word Accuracy (WA) results on four standard test sets for previous systems, and variations of the proposed system [7]. Here *m2m-P* refers to [7] utilizing m2m-aligner, *m2m-fst-P* refers to [7] using the proposed FST alignment algorithm, and *rnnlm-P* includes the application of RNNLM-based n-best rescoring.

System	NETtalk-15k	NETtalk-19k	CMUdict	OALD
Sequitur [1]	66.20	69.00	75.47	82.51
direcTL+ [2]	~	71.10	75.52	83.32
m2m-P	66.39	68.90	75.08	81.20
m2m-fst-P	66.50	69.50	75.25	81.86
rnnlm-P	**67.77**	**71.14**	**75.56**	**83.52**

Table 2: Optimal alignment parameters for *m2m-aligner* and the *proposed* aligner.

Aligner	maxX	maxY	delX	delY
m2m-aligner	2	2	True	False
Proposed	2	2	True	True

Table 3: Training times for the smallest (15k entries) and largest (112k entries) training sets.

System	NETtalk-15k	CMUdict
Sequitur [1]	Hours	Days
direcTL+ [2]	Hours	Days
m2m-P	2m56s	21m58s
m2m-fst-P	1m43s	13m06s
rnnlm-P	20m	2h

ment algorithm is typically 40% faster than *m2m-aligner* even though more complex parameters are being utilized. For comparison both the *DirecTL+* and *SequiturG2P* approaches require anywhere from hours to days to train models of similar quality [1, 2]. The complementary RNNLM models typically required 20 minutes to 2 hours to train.

7. Conclusions and Future Work

This work presented a novel, modified WFST-based m-to-one/one-to-m alignment algorithm which achieves a small but consistent improvement over previous proposals through the use of simple constraints. It also explored the application of RNNLM-based N-best rescoring to G2P conversion, and introduced a new open source WFST-based G2P conversion framework [7] which is extremely fast and, when combined with rescoring, consistently improved state-of-the-art G2P performance on a range of standard test sets.

We note that the proposed alignment algorithm may be applicable to *DirecTL+*, and that RNNLM rescoring should be applicable to both *DirecTL+* and *SequiturG2P*, although at the expense of increased training time and complexity. In future we plan to give a more detailed analysis of the improved alignment algorithm, and investigate a more general regularized or penalized EM approach. We also plan to extend the G2P framework to train N-gram models directly from the alignment lattices rather than just the one-best alignments. Finally minimum bayes risk decoding has proven effective for machine translation applications and we plan to extend the current framework to support this. On the RNNLM side we plan to implement a more integrated interpolation solution to replace the current N-best approach, simplify to eliminate tuning requirements, and integrate the solution into the existing open source project.

8. Acknowledgements

The authors would like to thank Tomáš Mikolov and Stefan Kombrink for providing valuable feedback on the theory and use of the RNNLM toolkit.

9. References

[1] M. Bisani, H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion", Speech Communication 50, 2008, pp. 434-451.

[2] S. Jiampojamarn, G. Kondrak, "Letter-to-Phoneme Alignment: an Exploration", Proc. ACL, pp. 780-788, 2010.

[3] S. Jiampojamarn, et. al, "Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion", NAACL HLT, pp. 372-379, 2007.

[4] D. Caseiro, I. Trancosoo, "Grapheme-to-Phoneme using finite state transducers", Proceedings 2002 IEEE Workshop on Speech Synthesis.

[5] D. Yang, P. Dixon, S. Furui, "Rapid development of a G2P system based on WFST framework", ASJ 2009, pp. 111-112.

[6] T. Mikolov, M. Karafiat, et. al, "Recurrent Neural Network based Language Model", Proc. InterSpeech, 2010.

[7] J. Novak, `http://code.google.com/p/phonetisaurus`

[8] E. Ristad and P. Yianilos, "Learning String Edit Distance", IEEE Trans. PRMI, pp. 522-532, 1998.

[9] S. Jiampojamarn, `http://code.google.com/p/m2m-aligner`

[10] M. Tomas, S. Kombrink, et. al, "RNNLM - Recurrent Neural Network Language Modeling Toolkit," ASRU 2011, demo session.