WFST-based Grapheme-to-Phoneme Conversion: Open Source Tools for Alignment, Model-Building and Decoding

Josef R. Novak, Nobuaki Minematsu, Keikichi Hirose
Graduate School of Information Science and Technology
The University of Tokyo, Japan
{novakj,mine,hirose}@gavo.t.u-tokyo.ac.jp

Abstract

This paper introduces a new open source, WFST-based toolkit for Grapheme-to-Phoneme conversion. The toolkit is efficient, accurate and currently supports a range of features including EM sequence alignment and several decoding techniques novel in the context of G2P. Experimental results show that a combination RNNLM system outperforms all previous reported results on several standard G2P test sets. Preliminary experiments applying Lattice Minimum Bayes-Risk decoding to G2P conversion are also provided. The toolkit is implemented using OpenFst.

1 Introduction

Grapheme-to-Phoneme (G2P) conversion is an important problem related to Natural Language Processing, Speech Recognition and Spoken Dialog Systems development. The primary goal of G2P conversion is to accurately predict the pronunciation of a novel input word given only the spelling. For example, we would like to be able to predict,

PHOENIX → /f i n I k s/

given only the input spelling and a G2P model or set of rules. This problem is straightforward for some languages like Spanish or Italian, where pronunciation rules are consistent. For languages like English and French however, inconsistent conventions make the problem much more challenging.

In this paper we present a fully data-driven, state-of-the-art, open-source toolkit for G2P conversion, *Phonetisaurus* [1]. It includes a novel modified Expectation-Maximization (EM)-driven G2P sequence alignment algorithm, support for joint-sequence language models, and several decoding solutions. The paper also provides preliminary investigations of the applicability of Lattice Minimum Bayes-Risk (LMBR) decoding [2; 3] and N-best rescoring with a Recurrent Neural Network Language Model (RNNLM) [4; 5] to G2P conversion. The Weighted Finite-State Transducer (WFST) framework is used throughout, and the open source implementation relies on OpenFst [6]. Experimental results are provided illustrating the speed and accuracy of the proposed system.

The remainder of the paper is structured as follows. Section 2 provides background, Section 3 outlines the alignment approach, Section 4 describes the joint-sequence LM. Section 5 describes decoding approaches. Section 6 discusses preliminary experiments, Section 7 provides simple usage commands and Section 8 concludes the paper.

2 G2P problem outline

Grapheme-to-Phoneme conversion has been a popular research topic for many years. Many different approaches have been proposed, but perhaps the most popular is the joint-sequence model [6]. Most joint-sequence modeling techniques focus on producing an initial alignment between corresponding grapheme and phoneme sequences, and then modeling the aligned dictionary as a series of joint tokens. The gold standard in this area is the EM-driven joint-sequence modeling approach described in [6] that simultaneously infers both alignments and subsequence chunks. Due to space constraints the reader is referred to [6] for a detailed background of previous research.

The G2P conversion problem is typically broken down into several sub-problems: (1) Sequence alignment, (2) Model training and, (3) Decoding. The goal of (1) is to align the grapheme and phoneme sequence pairs in a training dictionary. The goal of (2) is to produce a model able to generate new pronunciations for novel words, and the

Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing, pages 45–49,
Donostia–San Sebastián, July 23–25, 2012.

goal of (3) is to find the most likely pronunciation given the model.

3 Alignment

The proposed toolkit implements a modified WFST-based version of the EM-driven multiple-to-multiple alignment algorithm proposed in [7] and elaborated in [8]. This algorithm is capable of learning natural G-P relationships like `igh`→/AY/ which were not possible with previous 1-to-1 algorithms like [9].

The proposed alignment algorithm includes three modifications to [7]: (1) A constraint is imposed such that only `m-to-one` and `one-to-m` arcs are considered during training. (2) During initialization a joint alignment lattice is constructed for each input entry, and any unconnected arcs are deleted. (3) All arcs, including deletions and insertions are initialized to and constrained to maintain a non-zero weight.

These minor modifications appear to result in a small but consistent improvement in terms of Word Accuracy (WA) on G2P tasks. The Expectation and Maximization steps for the EM training procedure are outlined in Algorithms 2, 3. The EM algorithm

Algorithm 1: EM-driven M2One/One2M

Input: x^T, y^V, mX, mY, dX, dY
Output: γ, AlignedLattices
1 **foreach** *sequence pair* (x^T, y^V) **do**
2 | InitFSA(x^T, y^V, mX, mY, dX, dY)
3 **foreach** *sequence pair* (x^T, y^V) **do**
4 | Expectation(x^T, y^V, mX, mY, γ)
5 Maximization(γ)

is initialized by generating an alignment FSA for each dictionary entry, which encodes all valid G-P alignments, given max subsequence parameters supplied by the user. Any unconnected arcs are deleted and all remaining arcs are initialized with a non-zero weight. In Algorithm 2 lines 2-3 compute the forward and backward probabilities. Lines 4-8 compute the arc posteriors and update the current model. In Algorithm 3 lines 1-2 normalize the probability distribution. Lines 3-6 update the alignment lattice arc weights with the new model.

Algorithm 2: Expectation step

Input: $AlignedLattices$
Output: γ, $total$
1 **foreach** *FSA alignment lattice* F **do**
2 | $\alpha \leftarrow$ ShortestDistance(F)
3 | $\beta \leftarrow$ ShortestDistance(F^R)
4 | **foreach** *state* $q \in Q[F]$ **do**
5 | | **foreach** *arc* $e \in E[q]$ **do**
6 | | | $v \leftarrow ((\alpha[q] \otimes w[e]) \otimes \beta[n[e]]) \oslash \beta[0]$;
7 | | | $\gamma[i[e]] \leftarrow \gamma[i[e]] \oplus v$;
8 | | | $total \leftarrow total \oplus v$;

Algorithm 3: Maximization step

Input: γ, $total$
Output: AlignedLattices
1 **foreach** *arc* e *in* $E[\gamma]$ **do**
2 | $\gamma_{new}[i[e]] \leftarrow w[e]/total$; $\gamma[i[e]] \leftarrow 0$;
3 **foreach** *FSA alignment lattice* F **do**
4 | **foreach** *state* $q \in Q[F]$ **do**
5 | | **foreach** *arc* $e \in E[q]$ **do**
6 | | | $w[e] \leftarrow \gamma_{new}[i[e]]$;

4 Joint Sequence N-gram model

The pronunciation model implemented by the toolkit is a straightforward joint N-gram model. The training corpus is constructed by extracting the best alignment for each entry, e.g.:

```
a}x b}b a}@ c|k}k
a}x b}b a}@ f}f t}t
```

The training procedure is then, (1) Convert aligned sequence pairs to sequences of aligned joint label pairs, $(g_1{:}p_1, g_2{:}p_2, ..., g_n{:}p_n)$; (2) Train an N-gram model from (1); (3) Convert the N-gram model to a WFST. Step (3) may be performed with any language modeling toolkit. In this paper *mitlm* [11] is utilized.

5 Decoding

The proposed toolkit provides varying support for three different decoding schemes. The default decoder provided by the distribution simply extracts the shortest path through the phoneme lattice created via composition with the input word,

$$H_{best} = ShortestPath(Project_o(w \circ M)) \quad (1)$$

where H_{best} refers to the lowest cost path, $Project_o$ refers to projecting the output labels, w refers to the input word, M refers to the G2P model, and $\circ$ indicates composition.

5.1 RNNLM N-best rescoring

Recurrent Neural Network Language Models have recently enjoyed a resurgence in popularity in the context of ASR applications [4]. In another recent publication we investigated the applicability of this approach to G2P conversion with joint sequence models by providing support for the `rnnlm toolkit` [5]. The training corpus for the G2P LM is a corpus of joint sequences, thus it can be used without modification to train a parallel RNNLM. N-best reranking is then accomplished with the proposed toolkit by causing the decoder to output the N-best joint G-P sequences, and employing `rnnlm` to rerank the the N-best joint sequences,

$$\begin{aligned} H_{Nbest} &= NShortestPaths(w \circ M) \\ H_{best} &= Project_o(Rescore_{rnn}(H_{Nbest})). \end{aligned} \tag{2}$$

In practice the `rnnlm` models require considerable tuning, and somewhat more time to train, but provide a consistent WA boost. For further details on algorithm as well as tuning for G2P see [4; 10].

5.2 Lattice Minimum Bayes-Risk decoding for G2P

In [2] the authors note that the aim of MBR decoding is to find the hypothesis that has the "least expected loss under the model". MBR decoding was successfully applied to Statistical Machine Translation (SMT) lattices in [2], and significantly improved in [3]. Noting the similarities between G2P conversion and SMT, we have begun work implementing an integrated LMBR decoder for the proposed toolkit.

Our approach closely follows that described in [3], and the algorithm implementation is summarized in Algorithm 4. The inputs are the full phoneme lattice that results from composing the input word with the G2P model and projecting output labels, an exponential scale factor α, and N-gram precision factors θ_{0-N}. The θ_n are computed using a linear corpus BLEU [2] N-gram precision p, and a match ratio r using the following equations, $\theta_0 = -1/T;\ \ \theta_n = 1/(NTpr^{n-1})$. T is a constant

Algorithm 4: G2P Lattice MBR-Decode

Input: $\mathcal{E} \leftarrow Project_o(w \circ M)$, α, θ_{0-n}

1 $\mathcal{E} \leftarrow$ScaleLattice$(\alpha \times \mathcal{E})$
2 $\mathcal{N}_N \leftarrow$ExtractN-grams$(\mathcal{E})$
3 **for** $n \leftarrow 1$ **to** N **do**
4 $\quad \Phi_n \leftarrow$ MakeMapper$(\mathcal{N}_n)$
5 $\quad \Psi_n^R \leftarrow$ MakePathCounter$(\mathcal{N}_n)$
6 $\quad \mathcal{U}_n \leftarrow Opt((\mathcal{E} \circ \Phi_n) \circ \Psi_n^R)$
7 $\quad \Omega_n = \Phi_n$
8 $\quad$ **for** *state* $q \in Q[\Omega_n]$ **do**
9 $\quad\quad$ **for** *arc* $e \in E[q]$ **do**
10 $\quad\quad\quad w[e] \leftarrow \theta_n \times \mathcal{U}(o[e])$
11 $\mathcal{P} \leftarrow Project_{input}(\mathcal{E}_{\theta_0} \circ \Omega_1)$
12 **for** $n \leftarrow 2$ **to** N **do**
13 $\quad \mathcal{P} \leftarrow Project_{input}(\mathcal{P} \circ \Omega_n)$
14 $\mathcal{H}_{best} = ShortestPath(\mathcal{P})$

which does not affect the MBR decision [2]. Line 1 applies α to the raw lattice. In effect this controls how much we trust the raw lattice weights. After applying α, $\mathcal{E}$ is normalized by pushing weights to the final state and removing any final weights. In line 2 all unique N-grams up to order N are extracted from the lattice. Lines 4-10 create, for each order, a context-dependency FST (Φ_n) and a special path-posterior counting WFST (Ψ_n^R), which are then used to compute N-gram posteriors ($\mathcal{U}_n$), and finally to create a decoder WFST (Ω_n). The full MBR decoder is then computed by first making an unweighted copy of $\mathcal{E}$, applying θ_0 uniformly to all arcs, and iteratively composing and input-projecting with each Ω_n. The MBR hypothesis is then the best path through the result $\mathcal{P}$. See [2; 3] for further details.

6 Experimental results

Experimental evaluations were conducted utilizing three standard G2P test sets. These included replications of the NetTalk, CMUdict, and OALD English language dictionary evaluations described in detail in [6]. Results comparing various configuration of the proposed toolkit to the joint sequence model *Sequitur* [6] and an alternative discriminative training toolkit *direcTL+* [8] are described in Table 1. Here *m2m-P* indicates the proposed toolkit using the alignment algorithm from [7], *m2m-fst-P*

System	NT15k	CMUdict	OALD
Sequitur [6]	66.20	75.47	82.51
direcTL+ [8]	∼	75.52	83.32
m2m-P	66.39	75.08	81.20
m2m-fst-P	66.41	75.25	81.86
rnnlm-P	**67.77**	**75.56**	**83.52**

Table 1: Comparison of G2P WA(%) for previous systems and variations of the proposed toolkit.

indicates the alternative FST-based alignment algorithm, and *rnnlm-P* indicates the use of RNNLM N-best reranking.

The results show that the improved alignment algorithm contributes a small but consistent improvement to WA, while RNNLM reranking contributes a further small but significant boost to WA which produces state-of-the-art results on all three test sets.

The WA gains are interesting, however a major plus point for the toolkit is speed. Table 2 compares training times for the proposed toolkit with previously reported results. The *m2m-fst-P* for system for

System	NETtalk-15k	CMUdict
Sequitur [6]	Hours	Days
direcTL+ [8]	Hours	Days
m2m-P	2m56s	21m58s
m2m-fst-P	1m43s	13m06s
rnnlm-P	20m	2h

Table 2: Training times for the smallest (15k entries) and largest (112k entries) training sets.

CMUdict performs %0.27 worse than the state-of-the-art, but requires just a tiny fraction of the training time. This turn-around time may be very important for rapid system development. Finally, Figure. 1 plots WA versus decoding time for *m2m-fst-P* on the largest test set, further illustrating the speed of the decoder, and the impact of using larger models.

Preliminary experiments with the LMBR decoder were also carried out using the smaller NT15k dataset. The θ_n values were computed using p, r, and T from [2] while α was tuned to 0.6. Results are described in Table 3. The system matched the basic WA for N=6, and achieved a small improvement in PA over *m2m-fst-P* (%91.80 versus %91.82). Tuning the loss function for the G2P task should improve performance.

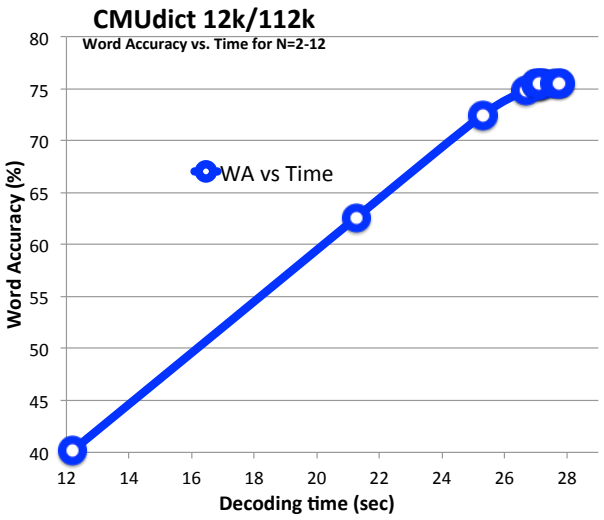

Figure 1: Decoding speed vs. WA plot for various N-gram orders for the CMUdict 12k/112k test/train set. Times averaged over 5 run using `ctime`.

NT15k	N=1	N=2	N=3	N=4	N=5	N=6
WA	28.88	65.48	66.03	66.41	66.37	**66.50**
PA	83.17	91.74	91.79	91.87	91.82	**91.82**

Table 3: LMBR decoding Word Accuracy (WA) and Phoneme Accuracy (PA) for order N=1-6.

7 Toolkit distribution and usage

The preceding sections introduced various theoretical aspects of the toolkit as well as preliminary experimental results. The current section provides several introductory usage commands.

The toolkit is open source and released under the liberal BSD license. It is available for download from [1], which also includes detailed compilation instructions, tutorial information and additional examples. The examples that follow utilize the NETTalk dictionary.

Align a dictionary:

```
$ phonetisaurus-align --input=test.dic \
   --ofile=test.corpus
```

Train a 7-gram model with mitlm:

```
$ estimate-ngram -o 7 -t test.corpus \
   -wl test.arpa
```

Convert the model to a WFSA

```
$ phonetisaurus-arpa2fst --input=test.arpa \
   --prefix=test
```

Apply the default decoder

```
$ phonetisaurus-g2p --model=test.fst \
   --input=abbreviate --nbest=3 --words
abbreviate 25.66 @ b r i v i e t
```

```
abbreviate 28.20 @ b i v i e t
abbreviate 29.03 x b b r i v i e t
```

Apply the LMBR decoder

```
$ phonetisaurus-g2p --model=test.fst \
   --input=abbreviate --nbest=3 --words \
   --mbr --order=7
abbreviate 1.50 @ b r i v i e t
abbreviate 2.62 x b r i v i e t
abbreviate 2.81 a b r i v i e t
```

8 Conclusion and Future work

This work introduced a new Open Source WFST-driven G2P conversion toolkit which is both highly accurate as well as efficient to train and test. It incorporates a novel modified alignment algorithm. To our knowledge the RNNLM N-best reranking and LMBR decoding are also novel applications in the context of G2P.

Both the RNNLM N-best reranking and LMBR decoding are promising but further work is required to improve usability and performance. In particular RNNLM training requires considerable tuning, and we would like to automate this process. The provisional LMBR decoder achieved a small improvement but further work will be needed to tune the loss function. Several known optimizations are also planned to speed up the LMBR decoder.

Nevertheless the current release of the toolkit provides several novel G2P solutions, achieves state-of-the-art WA on several test sets and is efficient for both training and decoding.

References

J. Novak, et al. [1]. `http://code.google.com/p/phonetisaurus`

R. Tromble and S. Kumar and F. Och and W. Macherey. [2]. *Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation*, Proc. EMNLP 2007, pp. 620-629.

G. Blackwood and A. Gispert and W. Byrne. [3]. *Efficient path counting transducers for minimum bayes-risk decoding of statistical machine translation lattices*, Proc. ACL 2010, pp. 27-32.

T. Mikolov and M. Karafiat and L. Burget and J. Černocký and S. Khundanpur. [4]. *Recurrent Neural Network based Language Model*, Proc. InterSpeech, 2010.

T. Mikolov and S. Kombrink and D. Anoop and L. Burget and J. Černocký. [5]. *RNNLM - Recurrent Neural Network Language Modeling Toolkit*, ASRU 2011, demo session.

C. Allauzen and M. Riley and J. Schalkwyk and W. Skut and M. Mohri. [6]. *OpenFST: A General and Efficient Weighted Finite-State Transducer Library*, Proc. CIAA 2007, pp. 11-23.

M. Bisani and H. Ney. [6]. *Joint-sequence models for grapheme-to-phoneme conversion*, Speech Communication 50, 2008, pp. 434-451.

S. Jiampojamarn and G. Kondrak and T. Sherif. [7]. *Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion*, NAACL HLT 2007, pp. 372-379.

S. Jiampojamarn and G. Kondrak. [8]. *Letter-to-Phoneme Alignment: an Exploration*, Proc. ACL 2010, pp. 780-788.

E. Ristad and P. Yianilos. [9]. *Learning String Edit Distance*, IEEE Trans. PRMI 1998, pp. 522-532.

J. Novak and P. Dixon and N. Minematsu and K. Hirose and C. Hori and H. Kashioka. [10]. *Improving WFST-based G2P Conversion with Alignment Constraints and RNNLM N-best Rescoring*, Interspeech 2012 (Accepted).

B. Hsu and J. Glass. [11]. *Iterative Language Model Estimation: Efficient Data Structure & Algorithms*, Proc. Interspeech 2008.