Failure transitions for Joint n-gram Models and G2P Conversion

Josef R. Novak, Nobuaki Minematu, 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 work investigates two related issues in the area of WFST-based G2P conversion. The first is the impact that the approach utilized to convert a target word to an equivalent finite-state machine has on downstream decoding efficiency. The second issue considered is the impact that the approach utilized to represent the joint n-gram model via the WFST framework has on the speed and accuracy of the system. In the latter case two novel algorithms are proposed, which extend the work from [1] to enable the use of failure-transitions with joint n-gram models. All solutions presented in this work are available as part of the open-source, BSD-licensed Phonetisaurus G2P toolkit [2].
Index Terms: G2P, WFST, model conversion

1. Introduction

Grapheme-to-Phoneme (G2P) modeling is an important topic in both Automatic Speech Recognition (ASR) and Text-to-Speech Synthesis (TTS). Joint n-gram models have proven to be a popular and effective approach to G2P conversion [3, 4, 5]. This approach also affords an efficient representation via the Weighted Finite-State Transducer (WFST) framework, however several caveats apply. In particular, while both approximate and exact methods have been proposed to represent standard statistical language models via the WFST framework [1], these exact algorithms are incompatible with joint n-gram models.

This work investigates two related issues in the area of WFST-based G2P conversion. The first is the impact that the approach utilized to convert a target word to an equivalent finite-state machine has on downstream decoding efficiency. The second issue considered is the impact that the approach utilized to represent the joint n-gram model via the WFST framework has on the speed and accuracy of the system. In the latter case two novel algorithms are proposed, which extend the work from [1] to enable the use of failure-transitions with joint n-gram models.

All solutions presented in this work are available as part of the open-source, BSD-licensed Phonetisaurus G2P toolkit [2], which is itself based on OpenFst [6]. This toolkit has recently been independently evaluated on a variety of large-scale corpora and languages, and shown to perform very competitively with other state-of-the-art solutions from both industry and academia [7].

2. WFST-based G2P Conversion

It will be useful to first briefly outline the WFST-based G2P conversion approach utilized in this work and implemented in Phonetisaurus [2]. The approach consists of loosely-coupled a three-step process. First the target pronunciation dictionary is aligned using an expectation-maximization training procedure based on [8] and detailed in [9], the result of which is a corpus of aligned, joint sequences. Before and after examples are depicted in Table 1 and Table 2, respectively. In the second step, the corpus of joint sequences is used as the input to train a standard joint n-gram model. The final preparatory step is then to convert the resulting n-gram model to an equivalent WFST. This step follows the same basic approach described in [10].

Word	Pronunciation
BRANDISHING	B R AE N D IH SH IH NG
CHAPPLE	CH AE P AH L

Table 1: Samples from the CMU pronunciation dictionary.

Aligned Entry
B:B R:R A:AE N:N D:D I:IH S,H:SH I:IH N,G:NG C,H:CH A:AE P,P:P L:L E:ϵ

Table 2: Results of aligning and reformatting the dictionary as a corpus of joint sequences. Here the "," indicates a one-to-many or man-to-one relationship, while "ϵ" represents a deletion or insertion.

The WFST-based model may then be utilized to produce pronunciation hypotheses for novel words by first transforming the target word into an equivalent finite-state machine and composing it with the model. The general decoding procedure is summarized in Equation 1.

$$pron_{best} = ShortestPath(Project_o(W \circ M)) \quad (1)$$

Here "$pron_{best}$" refers to the most likely pronunciation given the input word, "W" and the model, "M"; "$\circ$" refers to weighted composition; "$Project_o(\cdot)$" means to project the output labels. Finally, the "$ShortestPath(\cdot)$" refers to the shortest-path algorithm. Details on these and many other WFST-based algorithms may be found in [11]. Some form of beam pruning could also be applied to reduce computation time for large inputs, at the potential cost of some accuracy.

This basic decoder formulation is utilized in all following experiments, however the methods used to transform the target word into an equivalent finite-state machine, as well as the approach used to represent the G2P model, have a significant impact on the efficiency and accuracy of the resulting system. These two issues are discussed in the following sections. In general the approach adopted here, and implemented in Phonetisaurus [2] may be summarized a synthesis of minor, incremental improvements to the excellent work from [1, 5, 8, 10].

3. Word-to-FSM Conversion

There are several valid approaches that may be utilized to convert a target word to an equivalent finite-state machine. This choice and the approach utilized to represent the G2P model under the WFST framework are interdependent.

In the simplest case, the target word may be represented as a linear chain automaton, w. An example of this is depicted in Figure 1 for the word SIXTH. In practice the alignment process may learn more complex one-to-many or many-to-one relationships, implying a machine w', like that depicted in Figure 4. Here w' may be generated explicitly, or constructed via composition with a machine C, that encodes all valid subsequences. An example of a possible C machine is depicted in Figure 2. The process may be summarized, $w' = Project_o(w \circ C)$. The machine w' is compatible with a model representation scheme that utilizes standard ϵ-arcs to represent back-off transitions in the LM.

The second model modification algorithm proposed in this work, which is described in 4.1, requires that the target word be transformed into an FST. In this case new transitions must be added to w', one for each allowable G-P correspondence. Again, this may be achieved explicitly, or via composition. A suitable example machine is depicted in Figures 3, 5.

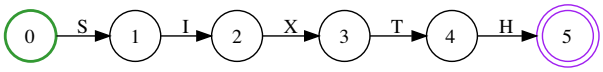

Figure 1: Example result of converting the word "SIXTH" to an equivalent linear FSA w.

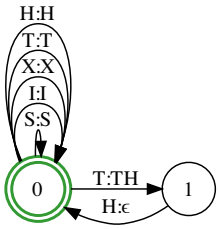

Figure 2: Single state FST C, suitable for expanding clusters.

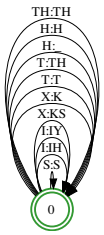

Figure 3: Single state FST F, mapping G↔P correspondences.

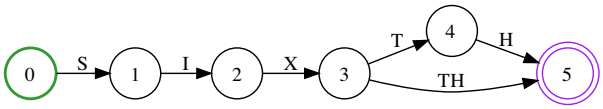

Figure 4: FSA w', result of computing $RmEps(Project_o(w \circ C))$.

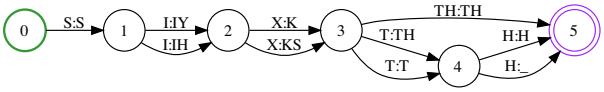

Figure 5: FST w'', result of computing $RmEps(Min(Det(w' \circ F)))$.

3.1. Word-to-FSM Experiments

The preceding section described four possible alternative methods for converting a target word into an equivalent finite-state machine: two that result in an FSA w', and two that result in an FST w''. Here we present experimental results that illustrate the run-time profiles and characteristics of these four methods, using two standard pronunciation dictionaries of varying size. The first dictionary is based on the NETtalk [12] dataset and contains 15k training entries and a 5k test set. The second is based on the CMUdict [13] dataset and contains 113k training entries and a 13k test set. The dictionary profiles are described in Table 3.

Dictionary	# Symbols		Number of words	
	G	P Train	Test	
NETtalk 15k/5k	26	50	14851	4951
CMUdict 113k/13k	27	39	106837	12000

Table 3: Pronunciation dictionary characteristics overview.

In order to determine which construction approach is most suitable, a series of experiments were run evaluating the run time performance of each of the four algorithms, using the 13k CMUdict test set. The results of these evaluations are described in Table 4. As described in the table, the explicit approaches,

FSA/T	∘	Avg #States	Avg #Arcs	Time (nsec)
FSA	N	8.50	10.25	98323
FSA	Y	11.25	13.25	861695
FST	N	8.50	136.40	191556
FST	Y	11.25	188.60	1101230

Table 4: CMUdict word-to-FSM conversion characteristics for 4 different algorithms.

while slightly more complex, are consistently more efficient for both w' and w''. These versions were therefore utilized for all subsequent experiments.

4. ARPA-to-WFST Conversion for joint n-gram models

The simplest approach to model conversion using the WFST framework represents back-off transitions in the LM as standard ϵ-transitions. It is possible to use the model in this approximate form to compute model probabilities for novel sequences, however it tends to generate redundant paths, and it may in some cases compute incorrect probabilities [1].

In [1] the authors describe two methods for achieving exact, correct behavior for a WFSA-based representation of a word n-gram model. The first method utilizes the special semantics of ϕ-transitions (failure), which encode the idea that an arc should be traversed only in the event that no valid normal transition exists leaving the state in question. This requires making several modifications to the set of final states, and the final state weights, but does not require the creation of new states or transitions. The second method creates an exact, offline, WFSA-based representation using ϵ-transitions, but requires the modification and creation of sometimes numerous new states and arcs, thus we focus on the ϕ-based method in here.

In the case of a joint n-gram model however, the algorithm proposed in [1] produces incorrect results. This is because an input grapheme may map to multiple output phonemes, but only a subset of output phonemes may be represented explicitly via outgoing transitions at a particular state. In such cases, the ϕ transition will not be traversed at all, resulting in only a small subset of the potentially valid set of pronunciation hypotheses.

This problem and its solution are best described via graphical examples. Figure 6 depicts a standard WFST-based representation of a joint n-gram model. In this case back-off transitions are represented as standard ϵ-transitions. In most cases this will be a good-enough solution, however strictly speaking it is inexact, and incorrect.

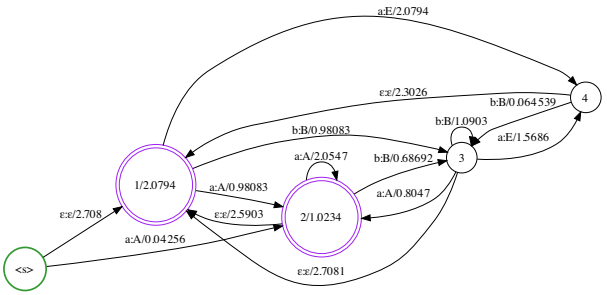

Figure 6: WFST-based representation of a simple G2P joint n-gram model. Here back-off transitions are represented using standard ϵ-transitions.

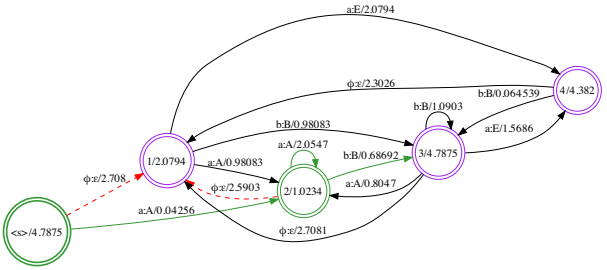

Figure 7: Illustration of attempting to compose linear FSA "aab" with a WFST representation of a joint n-gram model while interpreting back-off transitions as ϕ arcs. Green arrows indicate arcs that are traversed, while red, dashed arcs indicate arcs that are incorrectly ignored.

The ϕ-based method from [1] may be applied directly to a G2P joint n-gram model, however composition with novel words will produce largely incorrect results. An illustration of the model from Figure 6, this time modified to support ϕ-transitions, is depicted in Figure 7. This also illustrates the result of attempting to compose a short example word "aab" with the model. Composition produces one hypothesis "AAB", but the valid alternatives, "EAB", "EEB", "AEB" will not be considered, because the dashed red arcs will not be traversed. Two solutions to this problem are proposed in the following subsections. Explicit, fully-functional, open-source, BSD licensed solutions are provided in [2].

4.1. Encode-based solution

One solution to the above problem involves *encoding* the n-gram model, and generating WFST-based, encoded versions of the input words. In this case, generation of the test-word WFST is slightly more complex, but the standard ϕ-based approach from [1] may be used to generate the correct result from the encoded machines. The encoded, ϕ-ready version of the toy model is depicted in Figure 8.

4.2. Transition modification solution

A second possible approach starts by applying the standard algorithm from [1]. This is augmented by adding, for each state, new outgoing transitions to ensure that, for any input label, there is an explicit transition for each valid grapheme-phoneme correspondence that was learned during the alignment process.

A graphical example using the same toy model is depicted in Figure 9. Note that several new transitions have been added. These added transitions guarantee that all valid pronunciation hypotheses will be considered during composition.

```
def fsa_phiify( fst, all_io_labels ):
    fst = generic_phiify( fst )
    for state in fst:
        io_labels = {}
        for arc in fst.Arcs( state ):
            io_labels[arc.il].append( arc.ol )
        for il in io_labels:
            for ol in get_missing(
                io_labels[il],
                all_io_labels ):
                add_explicit_arc(
                    state, phi, il, ol, bo_w )
```

Listing 1: Python pseudocode for the `fsa_phi` algorithm. `add_explicit_arc()` recursively traverses the back-off arcs until an arc with the missing il/ol pair is found. Back-off costs are accumulated, and a new arc connecting the original state to the destination state of the found arc is created.

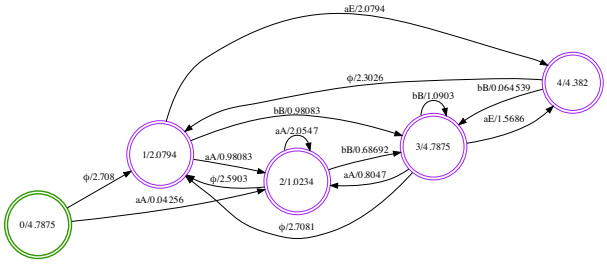

Figure 8: Example of ϕ-enabled WFST version of the toy G2P joint n-gram model. Note that the input-output labels have been *encoded*, resulting in a modified *acceptor* with joint labels. The same encoder must be used to generate the FSA version of a new input word.

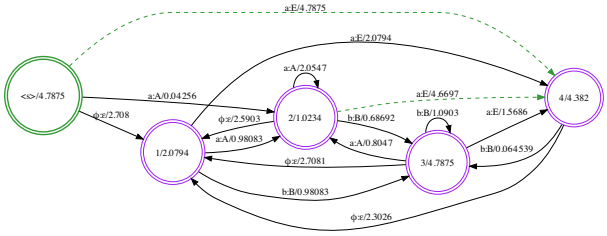

Figure 9: Example of modified ϕ-enabled WFSA version of the toy G2P joint n-gram model. In this case the model has been modified to add explicit transitions for each non-explicit grapheme-phoneme correspondence for each state. Transitions that have been added are green and dashed.

4.3. Performance evaluations

The three alternatives listed above entail different size, speed, and potentially accuracy characteristics.

The performance profiles for the three representation approaches for the CMUdict and NETtalk dictionaries are described in Table 5. Note that both ϕ-transition approaches produce the exact same PER/WER results. The `fst_phi` approach is notably slower than the other two approaches, however this is due primarily to the multiple *encode* and *decode* operations required for each word. The `fsa_phi` approach is only marginally slower than the ϵ-based approach, but the transition modifications impose a significant penalty in terms of model size. In particular, both the `fsa_eps` and `fst_phi` ap-

Decoder type	PER/WER[%]	Model Parameters			Time[s]	Avg. $w \circ M$ Parameters
(CMUdict)	(1-best)	#States	#Arcs	Size[M]	n=1/n=5	#States / Arcs / Epsilons
`fsa_eps`	8.24/33.55	128557	282168	5.8	1.95 / 129.42	154 / 292 / 157
`fsa_phi`	8.24/33.59	128557	882312	15	1.91 / 3.81	132 / 664 / 58
`fst_phi`	8.24/33.59	128557	282168	5.8	3.73 / 6.75	132 / 664 / 58
(NETtalk)	(1-best)	#States	#Arcs	Size[M]		#States / Arcs / Epsilons
`fsa_eps`	5.85/24.43	779024	1657271	34	13.59 / 354.60	338 / 660 / 359
`fsa_phi`	5.85/24.42	779024	13296429	212	16.83 / 42.75	308 / 4582 / 232
`fst_phi`	5.85/24.42	779024	1657271	34	45.79 / 74.24	308 / 4582 / 232

Table 5: CMUdict and NETtalk PER/WER results, model parameter information, average decoding speed for n=1 and n=5, and corpus averaged lattice characteristics for the $w \circ M$ composition for three decoder configurations. Times were averaged over 5 runs.

proaches result in a model size of 34M while the `fsa_phi` approach results in a final model size of 212M. The results for the smaller NETtalk dictionary show similar relative performance characteristics.

4.4. Generating n-best candidates

In the case of the 1-best hypothesis there is little variation among the different approaches in terms of accuracy, and the ϵ-based solution tends to be the fastest for larger dictionaries. This situation is notably different however, when generating n-best hypotheses. This is due to the fact that the inexact ϵ-based solution generates multiple paths for each hypothesis, because the back-off transition is traversed regardless of the presence or absence of a higher-order n-gram in the model. This means that often, in order to obtain a fixed number of unique n-best hypotheses, a considerably larger percentage of each lattice must be traversed in order to filter redundant paths that differ only in terms of ϵ-arc placement. The present implementation of this is in the epsilon case is somewhat crude, in that the user sets a heuristic n-best beam for the entire test set, and this is used as the basis for pruning the raw lattice.

The two right-most columns in Table 5 illustrate the relative run times for the three decoding approaches in the case of 1-best and 5-best hypotheses, as well as the average number of states, transitions, and ϵ-transitions contained in the pronunciation lattices that result from computing $w \circ M$, and projecting the output labels. The simple ϵ-based solution is clearly much slower, and on average generates significantly more ϵ-transitions. This is largely due to the number of redundant paths generated by this approach, but also due to the implementation.

The `fsa_eps` approach could conceivably be sped up by implementing a specialized n-shortest paths algorithm that filters paths differing only based on ϵ-placement. Note however that ϵ-transitions do occur in the lattices generated by the `fsa_phi` and `fst_phi` solutions. These correspond to *deletions*, transitions where the input grapheme mapped to the empty label. Although much rarer, this can also be a source of path redundancy, and a modified n-shortest paths implementation would benefit these solutions to a small degree as well. One could also remove epsilons prior to applying shortest path, however in practice this tends to be much slower and more expensive, this is also true for weighted determinization in this case.

5. Conclusions and Future work

This paper proposed several new, minor algorithmic developments related to WFST-based G2P conversion using joint n-gram models. It profiled four different approaches to converting input target words to finite-state machines, showing that, in this case explicit conversion algorithms, rather than WFST-based methods, produce more efficient results.

It also proposed two new algorithms, which extend the work in [1], and make it possible to convert joint n-gram models to equivalent Weighted Finite-State Transducers in such a way as to make them compatible with exact evaluation using the special semantics of failure-transitions. These two algorithms were then pitted against the default, epsilon-based method and their performance characteristics compared using two well known standardized pronunciation dictionaries of varying size and complexity. The results of these evaluations show that the proposed approaches are comparably efficient and achieve accuracy equal to that of the ϵ-based approach for 1-best results, but are significantly faster when generation of n-best hypotheses is required.

A final implication of this work is that, as in so many other areas of experience, an ensemble solution provides the best balance between competing interests. In this case, this means combining both WFST-based and non-WFST-based methods into a hybrid system, thereby producing a result that jointly optimizing competing concerns of speed, accuracy, and intelligibility.

In future it may be interesting to consider other transformation approaches, or look in more detail at the impact that these choices have on specific downstream lattice rescoring methods. Furthermore, all three approaches would benefit from a modified n-shortest path implementation that filters redundant paths.

Finally, the source code, test sets, models and scripts specific to this paper are all available as a bundle on the downloads page of the project website.

6. References

[1] C. Allauzen, M. Mohri, and B. Roark, "Generalized algorithms for constructing statistical language models," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, 2003, pp. 40–47.

[2] J. Novak, "Phonetisaurus g2p," 2012. [Online]. Available: http://code.google.com/p/phonetisaurus

[3] L. Galescu and J. Allen, "Bi-directional conversion between graphemes and phonemes using a joint n-gram model," 2001.

[4] S. F. Chen, "Conditional and joint models for grapheme-to-phoneme conversion," in *Proc. of Interspeech*, 2003.

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

[6] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "Openfst: A general and efficient weighted finite-state trans-

ducer library." in *CIAA*, ser. Lecture Notes in Computer Science, J. Holub and J. Zd?rek, Eds., vol. 4783. Springer, 2007, pp. 11–23.

[7] S. Hahn, P. Vozila, and M. Bisani, "Comparison of grapheme-to-phoneme methods on large pronunciation dictionaries and lvcsr tasks," in *Proc. INTERSPEECH*, 2012.

[8] S. Jiampojamarn, G. Kondrak, and T. Sherif, "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*. Association for Computational Linguistics, 2007, pp. 372–379.

[9] J. Novak, P. Dixon, M. Minematsu, K. Hirose, C. Horie, and H. Kashioka, "Improving wfst-based g2p conversion with alignment constraints and rnnlm n-best rescoring," in *INTERSPEECH*, 2012.

[10] D. Caseiro, I. Trancoso, L. Oliveira, and C. Viana, "Grapheme-to-phone using finite-state transducers," in *In: Proc. 2002 IEEE Workshop on Speech Synthesis. Volume*, 2002, pp. 1349–1360.

[11] M. Mohri, *Weighted Automata Algorithms*. springer, 2009.

[12] T. Sejnowski and C. Rosenberg, "Nettalk corpus," 1993. [Online]. Available: ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/beep.tar.gz

[13] R. L. Weide, "The carnegie mellon pronouncing dictionary," 1998. [Online]. Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict