Natural Language Engineering **22** (6): 907–938. © Cambridge University Press 2015
doi:10.1017/S1351324915000315

Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the WFST framework

JOSEF ROBERT NOVAK, NOBUAKI MINEMATSU
and KEIKICHI HIROSE

The University of Tokyo, Graduate School of Information Science and Technology, Tokyo, Japan
e-mails: novakj@gavo.t.u-tokyo.ac.jp, mine@gavo.t.u-tokyo.ac.jp,
hirose@gavo.t.u-tokyo.ac.jp

(*Received 26 April 2014; revised 25 July 2015; accepted 27 July 2015; first published online 7 September 2015*)

Abstract

This paper provides an analysis of several practical issues related to the theory and implementation of Grapheme-to-Phoneme (G2P) conversion systems utilizing the Weighted Finite-State Transducer paradigm. The paper addresses issues related to system accuracy, training time and practical implementation. The focus is on joint n-gram models which have proven to provide an excellent trade-off between system accuracy and training complexity. The paper argues in favor of simple, productive approaches to G2P, which favor a balance between training time, accuracy and model complexity. The paper also introduces the first instance of using joint sequence RnnLMs directly for G2P conversion, and achieves new state-of-the-art performance via ensemble methods combining RnnLMs and n-gram based models. In addition to detailed descriptions of the approach, minor yet novel implementation solutions, and experimental results, the paper introduces *Phonetisaurus*, a fully-functional, flexible, open-source, BSD-licensed G2P conversion toolkit, which leverages the OpenFst library. The work is intended to be accessible to a broad range of readers.

1 Introduction

The phrase Grapheme-to-Phoneme (G2P) conversion is typically used to refer to the process of automatically generating pronunciation candidates for previously unseen words, or generating alternative pronunciations for known words.

G2P conversion is an important problem in both the areas of Automatic Speech Recognition and Text-to-Speech synthesis. In the case of Automatic Speech Recognition, the true vocabulary is often dynamic in nature. This means that new words, or new pronunciation candidates for existing words may need to be added to the system on a regular basis. Analogous problems arise in the case of Text-to-Speech synthesis. In both cases, building accurate G2P systems is typically both difficult and resource intensive. This difficulty derives from the fact that the letter-to-sound rules for natural spoken languages tend to be rife with special rules, inconsistencies and conflicts.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Table 1. *Sample entries from the CMU pronunciation dictionary*

Word	Pronunciation
AARONSON	AA R AH N S AH N
AARONSON	EH R AH N S AH N
BRANDISHING	B R AE N D IH SH IH NG
CHAPPLE	CH AE P AH L
DETRACTORS	D AH T R AE K T ER Z
KRONER	K R OW N ER
TASTE	T EY S T
TEST	T EH S T
TEXTBOOK	T EH K S T B UH K
VAPID	V AE P AH D
WATERPROOF	W AO T ER P R UW F

Table 2. *Example alignments for '*`TEST→T EH S S T`*' and '*`TASTE→T EY S T`*', the latter of which requires a phonemic null*

T	E	S	T		T	A	S	T	E
|	|	|	|		|	|	|	|	|
T	EH	S	T		T	EY	S	T	ϵ

The Phonetisaurus G2P approach, which is the subject of this work, is another variation on the well-known joint multigram approach, and can be summarized in four steps. The first step is data preparation, which involves collecting a suitable pronunciation lexicon for training. This should include a list of known words and their corresponding pronunciations. The second step is to align the training lexicon, so as to approximate a mapping between the graphemes and phonemes in the lexicon. In the third step, the aligned corpus is utilized as the input to estimate a standard n-gram model, which is subsequently converted into a Weighted Finite-State Transducer (WFST). In the fourth and final step, pronunciations for previously unseen words are predicted by using weighted composition (Mohri, Pereira and Riley 2002) to compute the intersection of the WFST representation of the target word and the joint n-gram model. The most likely pronunciation is determined by extracting the shortest-path through the combined machine. In depth discussion of each of these steps is provided in the following pages, starting with the pronunciation dictionary below.

The pronunciation dictionary is typically constructed by expert linguists with experience in phonetics and phonology. Table 1 provides an example of several entries taken from the English-language, open source CMU pronunciation dictionary (Weide 1998). Once a suitable training dictionary has been obtained or created, the next step towards training a G2P model is to *align* the graphemes and phonemes in the dictionary. In some cases, if the number of graphemes and phonemes is the same, as depicted in Table 2, this may be very simple.

If the number of letters or phonemes does not match, it may be necessary to map letters or phonemes to a null symbol (ϵ) such as depicted in the second example

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Table 3. *Two example alignments for the word 'TEXTBOOK→T EH K S T B UH K'. The first depicts a naive one-to-one alignment, the second an arguably more natural one-to-many alignment*

T	E	X	T	B	O	O	K		T	E	X	T	B	O,O	K
|	|	|	|	|	|	|	|		|	|	|	|	|	|	|
T	EH	K	S	T	B	UH	K		T	EH	K,S	T	B	UH	K

Table 4. *Results of aligning and reformatting the dictionary as a corpus of joint sequences. A ',' indicates a one-to-many relationship, while 'ϵ' represents a null*

Aligned Entry
A,A:AA R:R O:AH N:N S:S O:AH N:N
A,A:EH R:R O:AH N:N S:S O:AH N:N
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
D:D E:AH T:T R:R A:AE C:K T:T O,R:ER S:Z
K:K R:R O:OW N:N E,R:ER
T:T A:EY S:S T:T E:ϵ
T:T E:EH S:S T:T
T:T E:EH X:K,S T:T B:B O,O:UH K:K
V:V A:AE P:P I:AH D:D
W:W A:AO T:T E,R:ER P:P R:R O,O:UW F:F

from Table 2. In yet other instances, the number of graphemes and phonemes may be equal, but a one-to-one alignment may still be incorrect. An example of this situation is depicted on the left-hand side of Table 3. In the TEXTBOOK example, graphemic and phonemic nulls could be used to produce a reasonable alignment, however it will almost certainly be more intuitive to instead permit one-to-many alignments between the grapheme and phoneme sequences. An example of this is depicted on the right-hand side of Table 3.

Section 3 describes a fully automated approach to acheive this sort of flexible alignment result utilizing the Expectation–Maximisation (EM) framework based on Yianilos and Ristad (1998), and Jiampojamarn, Kondrak and Sherif (2007), and the WFST paradigm. The result of the alignment step then, is an aligned dictionary, as depicted in Table 4.

The corpus of aligned joint sequences may be used directly to train a joint n-gram model. The first variation of this approach was proposed in Deligne, Yvon and Bimbot (1995) and various extensions and reformulations were proposed in Galescu and Allen (2001), Caseiro and Trancoso (2002), Chen (2003), and Damper *et al.* (2005). In Bisani and Ney (2008), the authors propose another joint n-gram approach which combines the alignment, segmentation and model estimation procedures into one integrated framework based on maximum-likelihood EM. In the latter case, the authors also released their toolkit, Sequitur G2P (Bisani and Ney 2008). This has represented the gold standard in joint n-gram models for G2P for some time. A detailed and comprehensive review of further related work in this area is provided

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

in Bisani and Ney (2008). The central advantage of the joint n-gram approach is that is both fairly simple, and facilitates the use of research and tools developed by the Statistical Language Modeling community over the past twenty years.

The final step in most G2P solutions is the decoding step. Here, the model trained on the input corpus is utilized to produce pronunciation hypotheses for input words. The simplest-possible solution is to compose an Finite-State Acceptor (FSA) representation of the target word with the G2P model, and extract the shortest-path through the resulting lattice of pronunciations. This is the same basic approach described in Caseiro and Trancoso (2002). More sophisticated decoding procedures may also be applied, some of which are investigated in Section 6.

The main contribution of the current work can be summarised as the synthesis of the most effective components of previously proposed solutions in the literature, with a clear focus on achieving a balance between speed, accuracy and flexibility. In contrast to Bisani and Ney (2008), the proposed toolkit opts to decouple the the alignment and joint multigram training stages, in the interest of increasing flexibility and reducing training time. This also makes it possible to compare several of the previously proposed approaches on an equal footing within the same framework. In contrast to earlier approaches (Galescu and Allen 2001; Caseiro and Transcoso 2002; Chen 2003; Damper *et al.* 2005), the current work provides support for training multiple-to-multiple, multiple-to-one / one-to-multiple alignments based on Jiampojamarn *et al.* (2007), as well as the ability to optionally utilise full alignment lattices for the purpose of training models based on fractional counts. In contrast to Galescu and Allen (2001), Chen (2003), Damper *et al.* (2005), and Bisani and Ney (2008), the proposed toolkit employs the WFST framework in order to leverage the flexibiility and practical efficiency that this provides. It also implements a novel solution for the handling of back-off transitions in a joint n-gram model under the WFST framework. Finally, the toolkit provides native support for direct decoding of joint multigram-based Recurrent Neural Network Language Models, which represents a novel contribution in this area.

In the present work, we make the argument that the simple, decoupled G2P approach, based on the joint n-gram model has not yet been thoroughly exploited, and more generally that simple solutions can often provide surprisingly good results, sufficient to encourage their adoption in business and industry contexts, and to suggest avenues for further research in academic circles, especially in the case of large lexica.

The remainder of the paper is structured as follows. Section 2 introduces the WFST framework. Section 3 describes the alignment training procedure. Section 4 provides a synopsis of several Statistical Language Modeling techniques, WFST-based representations of G2P models, and descriptions of joint sequence RnnLMs for G2P conversion. Section 6 explains the model decoding processes for joint n-gram models and joint sequence RnnLMs. Section 7 provides a wide range of experiments using the proposed toolkit with n-gram based models, RnnLMs and ensemble methods. Section 8 provides final analysis, concluding remarks and possible directions for future work. Appendix A provides several basic examples of the available tools and their usage.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

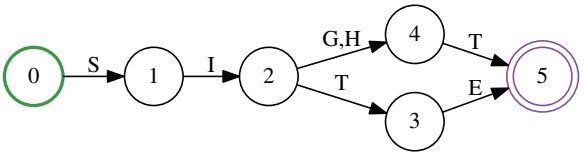


Fig. 1. (Colour online) Example FSA depicting possible graphemic representations of the homonyms 'SIGHT' and 'SITE'.

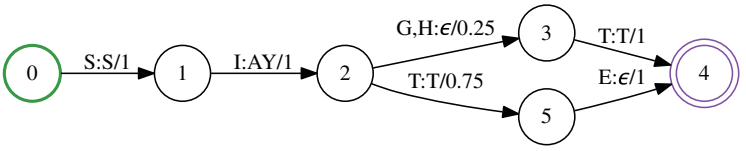


Fig. 2. (Colour online) Example WFST mapping the homophones 'SIGHT' and 'SITE' to the pronunciation 'S AY T'. The 'ϵ' symbol indicates null output.

2 Weighted finite-state transducer preliminaries

The WFST framework has gained considerable popularity in the speech and NLP communities in recent years, in part because it provides a unified representational framework applicable to a wide range of different modeling and decoding techniques. Casting the G2P problem in the language of WFST provides us with access to a broad array of tools suitable for combination, optimization, training and decoding. Before proceeding further, we present a short introduction to the WFST framework including several important definitions and frequently used algorithms.

The starting point for the WFST framework is the familiar unweighted FSA. An example of an FSA describing possible graphemic representations of two words is depicted in Figure 1. Valid paths through the machine are defined as the concatenation of labels connecting the start state to one of the final states in the machine. In the case of Figure 1, there are two valid paths: one for 'S I G,H T', where the 'G' and 'H' are encoded as a cluster, and one that represents the homonym 'S I T E'. The language represented by an FSA may be informally described as the set of valid paths that it encodes.

WFST extend the concept of the acceptor in two important ways. Namely, each arc or transition in the machine is extended to encode both a weight, and an output label, in addition to the input labels in the original FSA. The weight of a complete, valid path through the WFST is then determined by computing the product of the arc weights along the path. In a WFST, the output labels describe a second output language comprising again valid paths through the machine. The transducer itself then encodes a relation that maps paths from the input language to paths in the output language. Figure 2 illustrates another example relevant to the present work. In this example, the input language corresponds to the words 'SIGHT' and 'SITE', and the output language to the pronunciation 'S AY T'. The 'ϵ' symbol indicates null-output.

Depending on the structure of the relation between the input and output languages, the transducer may produce a single output, or it may produce multiple different outputs for a given input. The example depicted in Figure 2 illustrates the

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Table 5. *Definition of the* log *and* tropical *semirings utilized in this work. Note that* $\oplus_{log}$ *is defined as:* $x \oplus_{log} y = -log(e^{-x} + e^{-y})$

Name	Set	$\oplus$ (Plus)	$\otimes$(Times)	$\bar{0}$(Zero)	$\bar{1}$(One)
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Log	$\mathbb{R} \cup \{-\infty, +\infty\}$	$\oplus_{log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R} \cup \{-\infty, +\infty\}$	*min*	$+$	$+\infty$	0

former case. Both input orthographies, '`S I G,H T`', and '`S I T E`' will produce the same pronunciation output, '`S AY T`'. In the example, the weights of the two outputs are differentiated by their relative unigram frequency in English. In the event that an input path is not contained in the input language of the WFST, the transducer will not produce any output; rather it will reject the input. An FSA or Weighted Finite-State Acceptor (WFSA), on the other hand does not perform any mapping, but is said to represent, or accept a particular input language.

The WFST framework provides a wide range of known operations for manipulating machines. Many of these, including determinization, minimization and epsilon-removal, alter the structure of the WFST (e.g. the number of states, arcs, epsilon transitions, or the positions of weights and labels) without changing the language that the machine recognizes, the relation that it encodes or the associated total path weights. Others such as shortest-path and composition result in the creation of new machines.

All of these operations are underpinned by algebraic structures called semirings (Mohri 2002). Informally, these structures govern the way that path weights are computed by overloading the properties of the Plus ($\oplus$), Times ($\otimes$), Zero / annihilator ($\bar{0}$) and One / identity ($\bar{1}$) operators, as well as the set of values over which the operations are valid for a given semiring. The example in Figure 2 employs the familiar *probability* semiring. The current work relies primarily on two particular semirings: the *log* semiring and the *tropical* semiring, which are defined, along with the *probability* semiring in Table 5. The particular properties of each semiring influence its suitability for different applications. In particular, the $\oplus$-operation for the *log* semiring indicates a summation of partial path weights. This makes it the right choice for the EM driven alignment algorithm described in Section 3. The $\oplus$-operation in the *tropical* semiring, by contrast, considers only the least costly partial path, making it suitable for the decoding stage described in Section 6, where the goal is to find the shortest-path through a lattice of pronunciation hypotheses. It is also suitable for using negative log probabilities, which matches with the language model representations employed throughout the current work. A detailed formal discussion of semirings is provided in Mohri (2002).

There exists a wide variety of different algorithms to combine, manipulate and optimise WFSTs. It is beyond the scope of this work to discuss these algorithms in detail, however there are three in particular that bear mention, as they are utilised frequently by the proposed system. In the current work, the most fundamental of these operations is *weighted composition*, which is used to hierarchically cascade

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

multiple WFSTs. Informally, given two input WFSTs, A and B, the composition algorithm works by matching paths in the output language of A with paths in the input language of B. The result of composition, $C=A \circ B$, is a new WFST, C where each input path u mapping to output path w in C is determined by matching input path u mapping to output path v in A, to input path v mapping to output path w in B. The weights of paths in the composition result, C are determined by computing the $\otimes$-product of the corresponding paths in A and B, according to the specified semiring (Mohri *et al.* 2002). The process may be applied to an acceptor by simply mirroring the input labels to output labels to produce an equivalent transducer.

Other operations ultized in the current work include *projection*, which is used to transform a WFST into a WFSA representing just the input language or just the output language of the original machine. In the current work, this operation is utilized to obtain pronunciation lattices during the decoding stage. The *shortest distance* operation, which is used to compute the shortest distance from the start state to every other state, is utilized by the EM-alignment stage in the current work. Finally, the *shortest-path* operation, which is used to compute the shortest-path through the input machine starting from the start to a final state, is utilized during both the alignment and decoding stages. Formal definitions and in-depth discussions of these algorithms and many more can be found in Mohri *et al.* (2002) and Mohri (2002). Similarly, robust open-source implementations and examples of these algorithms are provided in the *OpenFst* toolkit (Allauzen *et al.* 2007).

3 Grapheme-to-Phoneme alignment

3.1 Algorithm motivation and overview

The typical first step in training a G2P system involves aligning the corresponding grapheme and phoneme sequences in the input training dictionary. The approach adopted in this work is based on the EM driven multiple-to-multiple alignment algorithm proposed in Jiampojamarn *et al.* (2007) and extended in Jiampojamarn and Kondrak (2010). This algorithm is capable of learning complex G↔P relationships like `PH`→/`F`/, and represents an improvement over earlier one-to-one stochastic alignment algorithms such as that introduced in Yianilos and Ristad (1998). This approach is also similar to that proposed in Deligne *et al.* (1995), but appears to have been independently discovered.

The alignment algorithm utilized in this work includes three minor modifications to the work of Jiampojamarn *et al.* (2007): (1) A constraint is imposed such that, in addition to `one-to-one` relationships, only `multiple-to-one` and `one-to-multiple` relationships 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 Error Rate (WER) on G2P tasks. Finally, the training procedure is cast in the WFST framework, which admits a very concise representation and implementation of the algorithm.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

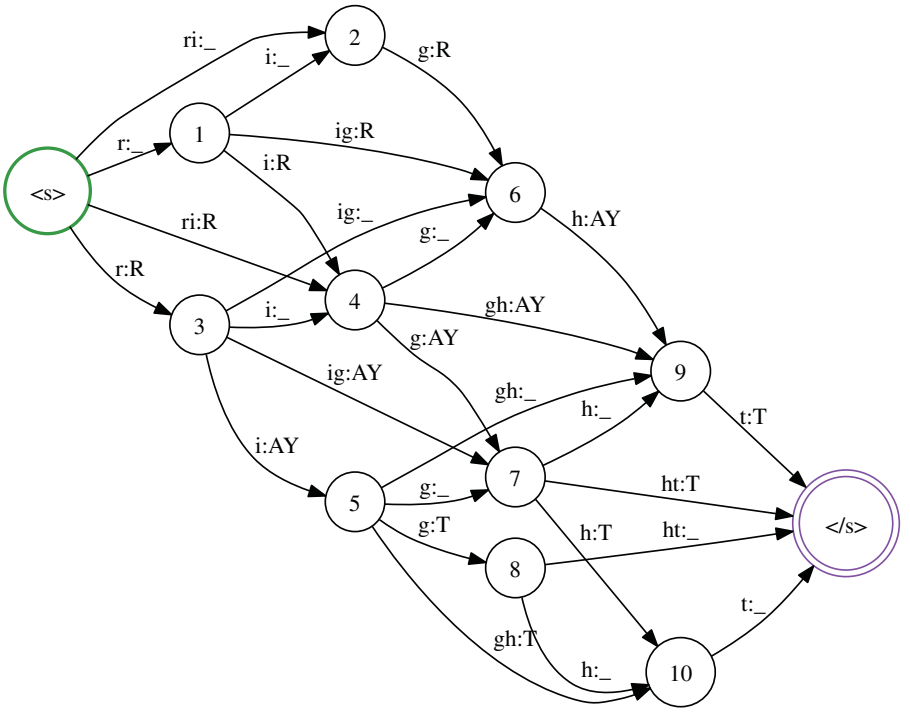


Fig. 3. (Colour online) Example alignment lattice for the word↔pronunciation pair 'RIGHT', 'R AY T' illustrating all possible alignments between the graphemes and phonemes in the given entry. In this example, the maximum length for a grapheme subsequence has been set to two, the maximum phoneme subsequence length to one, and deletions were permitted only on the phoneme side.

The first step in the WFST-based version is to generate an alignment lattice for each word-pronunciation pair, based on the user-supplied input parameters.

Figure 3 illustrates an example FST that encodes all permissible alignments for the word-pronunciation pair 'RIGHT', 'R AY T', given the constraints that (a) the maximum length for a grapheme subsequence is two, (b) the maximum phoneme subsequence length is and deletions are permitted only on the phoneme side. Once an alignment lattice has been generated for each entry in the training dictionary, the EM training procedure is initialized by setting all possible grapheme–phoneme alignment pairs to uniform probability. Next, the set of alignment lattices are passed to the expectation function. The WFST-based version of the algorithm is summarized in Algorithm 1. The procedure is initialized with a lexicon and a set of user-supplied constraints. Lines 1–2 process each word/pronunciation pair and generate a WFST lattice like the example depicted in Figure 3. The EM steps described in lines 3–5 are then repeated until the algorithm either converges, or until a prespecified maximum number of iterations is reached. The expectation step is described in Algorithm 2; this is fundamentally the same as the approach described in Jiampojamarn *et al.* (2007), with the exception that it has been reformulated to take advantage of the WFST framework.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Algorithm 1: EM-driven M2One/One2M

Input: *sequence pairs*, *seq*1_*max*, *seq*2_*max*, *seq*1_*del*, *seq*2_*del*
Output: γ, AlignedLattices
1 **foreach** *sequence pair* (*seq*1, *seq*2) **do**
2 | *lattice* ←Seq2FST(*seq*1, *seq*2, *seq*1_*max*, *seq*2_*max*, *seq*1_*del*, *seq*2_*del*)
3 **foreach** *lattice* **do**
4 | Expectation(*lattice*, γ)
5 Maximization(γ, *total*);

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* $\oplus v$;

Here, the traditional *forward* and *backward* steps are implemented in lines 2–3 using the *shortest-distance* algorithm. This is computed in the *log* semiring because we wish to compute the sum of *all* paths leading into each state. Next, the *alpha* and *beta* values are used to compute the arc posteriors according to the standard formula, $\gamma_{ij} = \frac{\alpha_i \cdot w_{ij} \cdot \beta_j}{\beta_0}$. Here, α_i represents the shortest distance in the *log* semiring from the start state to state i, and is represented by $\alpha[q]$ on line 6 in Algorithm 2. Similarly, β_j represents the shortest distance in the *log* semiring from the final state to state j and is represented by $\beta[n[e]]$ on line 6. Finally, w_{ij} represents the original weight for the arc connecting states i and j, and corresponds to $w[e]$ on line 6. Line 7 sums the arc posteriors for each possible grapheme–phoneme correspondence over the set of alignment lattices, and Line 8 keeps a running total, which is utilized to normalize the arc weights during the maximization step.

The maximization step in this case corresponds to normalizing the partial counts that were accumulated during the expectation step. This is outlined in 3. It is possible to perform a conditional maximization using the FST formalism, in which case the partial counts are normalized on a per-grapheme or per-phoneme basis,

Algorithm 3: Maximization step

Input: γ, *total*
Output: γ_{new}
1 **foreach** $i[e] \in \gamma$ **do**
2 | $\gamma[i[e]]_{new} \leftarrow \gamma[i[e]] \oslash$ *total*

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

or jointly in which case the partial counts are normalised over the complete set of grapheme–phoneme correspondences (Shu and Hetherington 2002).

In the present work, we focus on joint maximization, which reduces to normalizing the partial counts for each correspondence using the final value of *total* returned at the end of the expectation step. Finally, the lattice arc weights are reset to the new values and the expectation step is called again. This process terminates either when it reaches the maximum number of iterations, or when the change between the current iteration and previous iteration is less than some prespecified threshold.

Once the EM training process has successfully terminated, the most likely alignment can be extracted by mapping each alignment lattice to the *tropical* semiring and running the shortest-path algorithm (Mohri 2002). This is necessary as the Shortest-path algorithms require that the associated semiring have the path property, and be right distributive (Mohri 2002). The alignments may be printed out to create a corpus such as that depicted in Table 4. The n-best alignments may also be extracted, however in this case the resulting fractional counts will complicate downstream model training. It is similarly possible to utilize the full alignment lattices however, in practice this tends to be extremely resource intensive, even for small corpora, and results in little gain in terms of word or phoneme accuracy.

4 Joint sequence *n*-gram models for G2P conversion

4.1 Introduction

Once the input pronunciation dictionary has been successfully aligned, the next step is to train a model that can be used to produce pronunciation hypotheses for previously unseen words. In this work, we focus exclusively on joint n-gram models, which continue to enjoy considerable success in the area of G2P conversion. The training approach in the proposed system is identical to that used for modeling word sequences except for the fact that the 'words' are joint G↔P chunks learned during the alignment process. This means that any standard statistical language modeling toolkit may be used to train a joint n-gram model for the proposed system.

In the current work, we focus on two popular smoothing approaches: Witten–Bell smoothing (Bell, Cleary and Witten 1990; Witten and Bell 1991), and Kneser–Ney smoothing (Kneser and Ney 1995). Kneser–Ney and its variations consistently outperform Witten–Bell smoothing, however Witten–Bell smoothing generalizes naturally to fractional counts. This means it can be used with lattices and n-best results such as those produced during the G2P alignment process.

In addition to standard n-gram language modeling techniques, maximum-entropy language models have been shown to perform competitively in G2P tasks (Chen 2003). The maximum-entropy formulation admits a direct conversion to ARPA-format without loss of information (Wu 2002), and an open-source implementation of this algorithm, which focuses on n-gram features, can be found in the SRILM toolkit (Stolcke 2002; Alumäe and Kurimo 2010). Evaluations with maximum-entropy joint n-gram models are also provided in Section 7.2.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

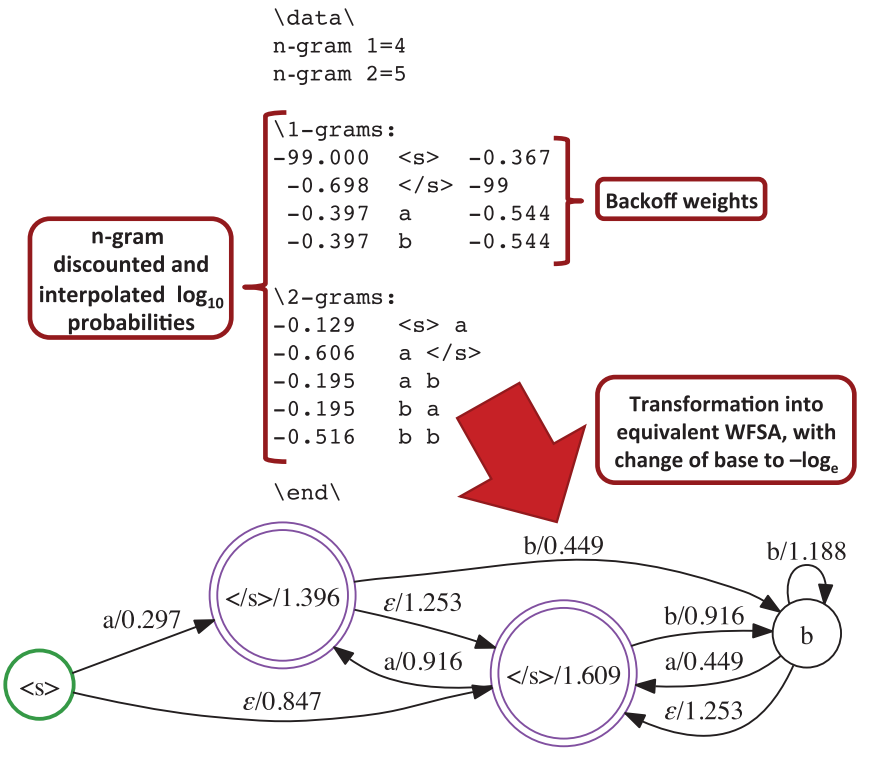


Fig. 4. (Colour online) Example transforming an ARPA format statistical language model into equivalent WFSA format. This is a bi-gram model trained on a small toy corpus, using interpolated Kneser–Ney smoothing. Note that the conventional base for the ARPA format and the *tropical* semiring differ, and that the sentence-begin (`<s>`) and sentence-end (`</s>`) tags are given special treatment during model estimation.

4.2 WFSA and WFST representations

In order to utilize a joint n-gram model in the proposed WFST-based G2P system, it is first necessary to convert the model from the standard ARPA format representation to an equivalent transducer. Several algorithms and implementations suitable for representing a standard n-gram language model as an equivalent WFSA are proposed in Allauzen, Mohri and Roark (2003), Roark *et al.* (2012).

An example of the result of converting a standard ARPA format n-gram model to WFSA format is depicted in Figure 4. In this representation, both the sentence-begin (`<s>`) and sentence-end (`</s>`) tokens are implicitly represented by the start and final states in the WFSA, but are not explicitly represented in the graph. As noted in Roark *et al.* (2012), this representation is more concise and also allows the sentence-begin and sentence-end tokens to be specified at run-time. It is also worth noting here that the ARPA format traditionally represents n-gram probabilities in log_{10} format, while the WFST framework conventionally employs $-log_e$.

Note also that the `<s>` and `</s>` tokens are given special treatment during the model estimation phase. Each sentence is implicitly understood to begin with `<s>` and this token is not counted as a unigram event, although it is included in the

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

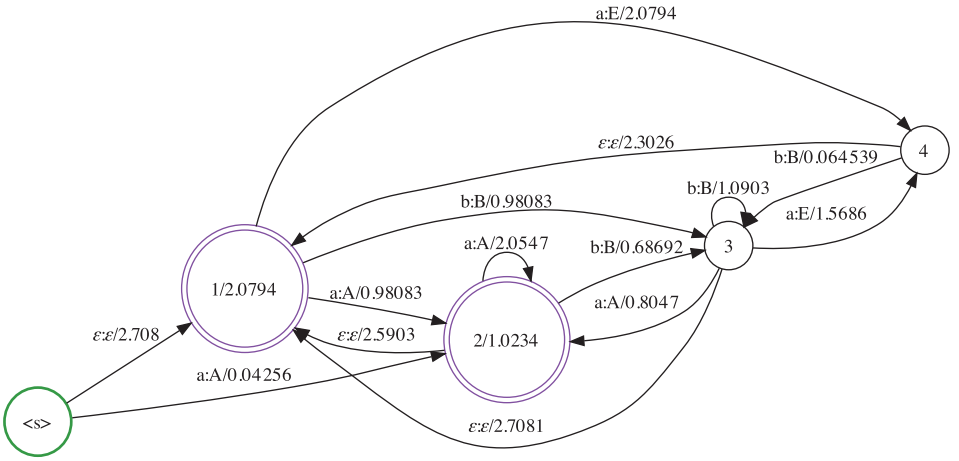


Fig. 5. (Colour online) G2P n-gram model representation using ϵ back-off transitions.

model vocabulary. Similarly, the `</s>` token is implicitly defined as marking the end of a sentence, and thus has no back-off weight associated with it.

In the case of a joint n-gram model, the model must be represented not as an acceptor but as a transducer. Here, the joint tokens learned during the alignment process are reseparated. Input labels represent graphemes, while the output labels represent phonemes.

This final step will allow downstream composition with novel input words, which are represented as unweighted acceptors. The default approach in Phonetisaurus utilizes standard epsilon transitions to represent back-off weights. Strictly speaking, this is not correct as it means that the back-off arc will be traversed regardless of whether or not a higher-order n-gram exists. In Allauzen *et al.* (2003), an alternative solution is proposed using special failure arcs, which are only traversed when no other valid match is found. Two algorithms for adapting this approach to joint n-gram models are proposed in Novak, Minematsu and Hirose (2013). These are summarized in Section 4.3 and implementations are provided in the proposed toolkit.

4.3 Failure transitions and joint n-gram models

The starting point for model transformation is to utilize ϵ-based back-off transitions. This is an approximate solution which works well in practice, but is both inexact, and tends to generate redundant paths which differ only in the placement of back-off transitions. An example of this starting point is depicted in Figure 5.

Utilizing the ϕ-based method from Allauzen *et al.* (2003) will largely produce incorrect results.

A ϕ-based equivalent of Figure 5, is depicted in Figure 6. The green arcs depict the result of composing the word '`aab`' with the model. This produces the hypothesis '`AAB`', but the hypotheses, '`EAB`', '`EEB`', '`AEB`' are ignored because the failure arc is not traversed. Two solutions to this problem follow.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

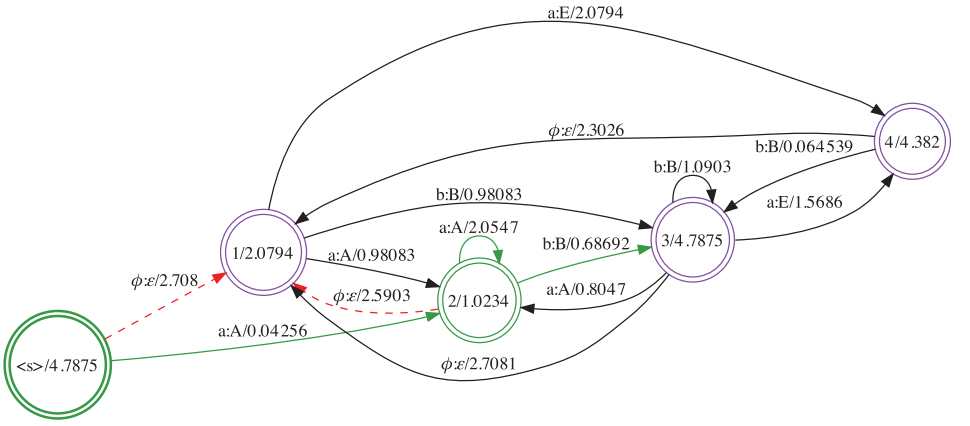


Fig. 6. (Colour online) Illustration of attempting to compose linear FSA '`aab`' with a WFST representation of a joint n-gram model while interpreting back-off transitions as ϕ arcs. Green arrows indicate arcs that are traversed, while red, dashed arcs indicate arcs that are incorrectly ignored.

4.3.1 *Encode-based solution*

The first solution is to encode the input and output labels in the joint n-gram model, creating an acceptor. The same is done with the input test word, taking care to generate all possible grapheme–phoneme alignments that were learned during the training phase. Once this has been done, the standard ϕ-based approach from Allauzen *et al.* (2003) can be used. An example illustrating this solution is depicted in Figure 7.

4.3.2 *Transition modification solution*

The second solution augments the original algorithm (Allauzen *et al.* 2003), by inspecting the input label of each outgoing arc, and adding new transitions wherever necessary in order to guarantee that all valid grapheme–phoneme correspondences are supported in the final result.

```
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()` iterates across the back-off arcs until an arc with the missing il/ol pair is found. A new arc is created connecting the original state to the destination state, and the weight is set to the accumulated back-off cost.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

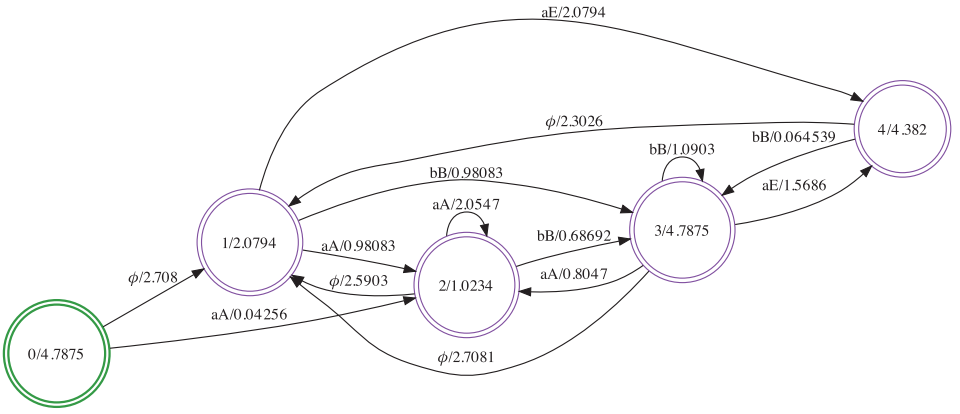


Fig. 7. (Colour online) Example of ϕ-enabled, WFSA version of the G2P joint n-gram model where the input–output labels have been encoded.

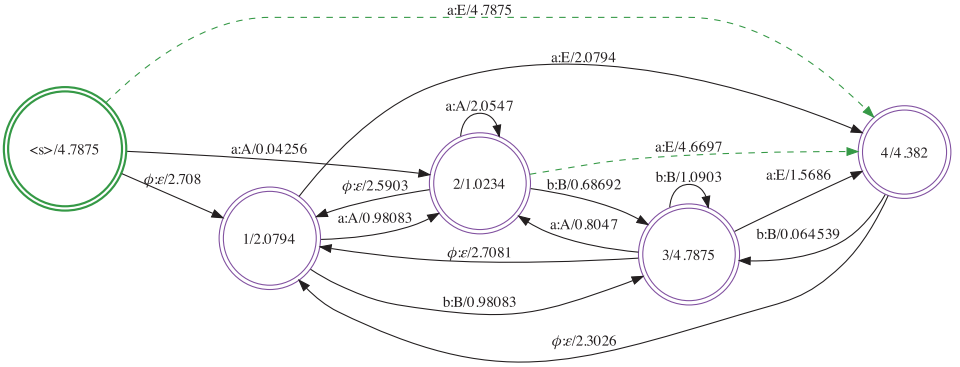


Fig. 8. (Colour online) Example of modified ϕ-enabled WFST version of the G2P joint n-gram model. Explicit transitions have been added for each missing grapheme–phoneme correspondence at each state. Transitions added by the algorithm are dashed and green.

An example of the second solution is shown in Figure 8. New transitions have been added to guarantee that all valid pronunciation hypotheses will be considered during composition.

4.3.3 Comparison of methods

In practice, the three methods summarized above produce equivalent PER/WER results. The two ϕ-transition based methods produce the exact same results, while the standard ϵ based method produces slight variation but no consistent improvement or degradation in accuracy.

In terms of run times, the ϵ based solution is the fastest for one-best, but slows down dramatically when generating n-best. This is due to the fact that there are often multiple variations of each pronunciation hypothesis, which differ only in the placement of back-off transitions. In the case of n-best, it is more efficient to employ the `fsa_phi` or `fst_phi` solution instead. Further details regarding the performance

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

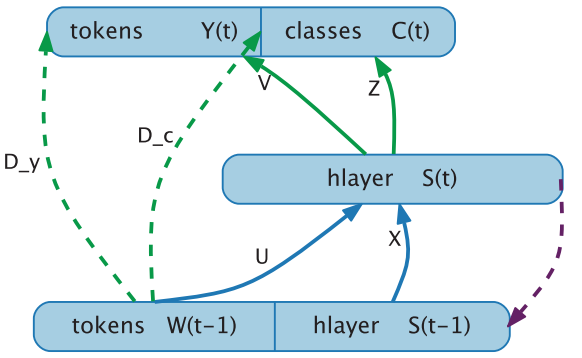


Fig. 9. (Colour online) Full RnnLM architecture utilized for G2P conversion.

and accuracy characteristics for the three methods mentioned above can be found in Novak, Minematsu and Hirose (2013).

5 Joint sequence RnnLMs for G2P conversion

5.1 Introduction

Recurrent Neural Network Language Models have recently enjoyed a resurgence in popularity in the context of Automatic Speech Recognition applications (Mikolov *et al.* 2010). In another recent publication (Novak *et al.* 2012), we investigated the applicability of this approach to G2P conversion with joint sequence models, by providing rescoring support for the `rnnlm toolkit` (Mikolov *et al.* 2011).

Here, we provide a brief description of the RnnLM architecture, and introduce a series of recommendations that serve to optimize the approach for G2P conversion. Finally, we provide a mechanism and implementation for performing efficient direct decoding using a joint token G2P RnnLM. The full architecture of the RnnLM is depicted in Figure 9.

The network is represented using specially partitioned input, hidden and output layers. These have corresponding weight matrices **U** and **X**, which map between the input and hidden layers; **V** and **Z**, which map between the hidden and output layers; and $\mathbf{D}_y$ and $\mathbf{D}_c$, which map directly between the input and output layers (Mikolov 2012). The network is trained using the backpropagation through time (BPTT) algorithm, with the goal of maximizing the likelihood of the training data.

The input layer consists of an indexed vector representing the vocabulary, which corresponds to the set of joint multigrams in the G2P case. This is augmented with a copy of the hidden layer activations from the previous time step. This two-component input layer feeds into the hidden layer. The hidden layer then feeds separately into an output layer, which is again split into two components. The first component partitions the vocabulary into disjoint classes based on unigram frequency. The second represents the class conditional probabilities for the vocabulary. The goal of this partitioning is primarily to speed up computation. Finally, the direct connections between the input and output layers simulate maximum-entropy features based on n-gram histories. In practice, the n-gram histories are hashed into a fixed size array.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

This serves to keep the number of direct-connections tractable, while at the same time effect a pruning process where more frequently occurring features are favored when collisions occur. Output values for the various layers are then computed as described in Mikolov (2012).

5.2 RnnLM training caveats for G2P conversion

In practice, training efficient, effective and accurate RnnLMs requires the use of considerable tuning both at the implementation and training levels. Recommendations for efficient training, as well as an implementation are provided in Mikolov *et al.* (2011), and Mikolov (2012), however these focus on applying the work to textual data, specifically in the area of Large Vocabulary Speech Recognition. In the case of G2P conversion, there are a couple additional concerns.

First, it is important to always treat each dictionary entry independently during training. By default, the RnnLM toolkit treats the input text as a continuous stream of tokens. In this situation, BPTT training is conducted across sentence boundaries. This makes sense for Large Vocabulary Speech Recognition tasks, but it is ill suited to pronunciation dictionaries.

Second, it is essential that the training corpus be provided in randomized order. Pronunciation dictionaries are typically provided in lexicographically sorted order. If the sorted dictionary is utilized directly during training then the online or mini-batch variants of gradient descent have a strong tendency to find a poor local minimum.

Phonetisaurus also provides a further set of RnnLM-based examples suitable for training high-quality joint sequence RnnLMs for G2P conversion.

6 G2P decoding in the WFST framework

6.1 Introduction

The default decoder used in the proposed WFST-based approach, and implemented in the proposed toolkit, is similar to that described in Caseiro and Trancoso (2002). This version of the decoding process is summarized in Equation (1),

$$p_{best} = shortestpath(project_o(w \circ M)). \tag{1}$$

Here, 'p_{best}' refers to the best pronunciation hypothesis given the model, 'w' is a linear FSA representing the target word, and 'M' is a WFST-based representation of the joint n-gram model. The '$\circ$' operator refers to weighted composition, and $project_o(\cdot)$ refers to projecting just the output labels (phonemes) of the composition result, which produces a phoneme lattice of potential pronunciations. Finally, $shortestpath(\cdot)$ refers to the shortest-path algorithm as described in Section 2. The n-best pronunciation hypotheses may be extracted in a similar fashion.

In the proposed approach, it is necessary to make several modifications to the input FSA. The multiple-to-multiple EM alignment process may produce grapheme and phoneme tokens that consist of a cluster of two or more symbols from the original grapheme or phoneme alphabets. If these clustered tokens are not accounted for when constructing the input FSA, any examples of these clusters in the joint n-gram

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

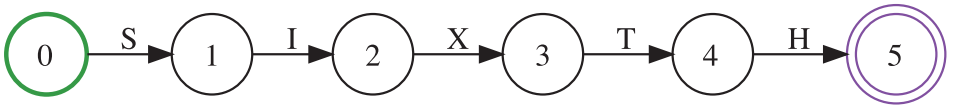


Fig. 10. (Colour online) Example result of converting the word 'SIXTH' to an equivalent linear FSA w. This example does not utilize insertion self-loops.

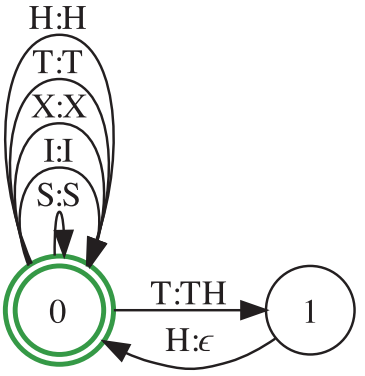


Fig. 11. (Colour online) Flower transducer C, suitable for expanding clusters.

model will be ignored during composition. There are two possible solutions to this problem. One solution is to create a single-state FST that maps any longer grapheme subsequences to a single cluster label, while passing through single graphemes, then compose it with the linear FSA w, and apply $project_o(\cdot)$ to the result. The alternative is to build the combined FSA explicitly. The WFST-based solution is arguably more flexible, however it is also somewhat more computationally intensive. Implementations for both approaches are provided in the proposed toolkit.

One final potential pitfall needs to be mentioned. If the data utilized to train the alignments is limited and certain subsequences occur only in clustered contexts, this might result in an aligned corpus that does not contain examples of every individual grapheme in isolation. In order to ensure that creation of the linear FSA representation succeeds, it is necessary to add all individual graphemes to the input symbols table, even if some do not appear in the aligned corpus.

An example of the component machines and conversion process for the input word 'SIXTH' is illustrated in Figures 10–14. Figure 10 depicts a linear FSA w, representing the word 'SIXTH'. Figure 11 depicts a flower transducer C (Kempe 2001), that may be used to augment w with cluster arcs. Figure 13 depicts the result of computing $RmEps(Project_o(w \circ C))$. Figure 12 depicts a single state flower transducer that may be used to map G2P correspondences. Here, the '_' indicates that the grapheme 'H' may optionally map to a null phoneme output. Figure 14 depicts the result of computing $RmEps(Min(Det(w' \circ F)))$. This final result is an FST w'', that encodes all possible paths through the joint n-gram model, based on the set of G↔P correspondences learned during the alignment process. Note that the optimization routines are not strictly necessary at this stage, and may require label-encoding to succeed in the general case.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

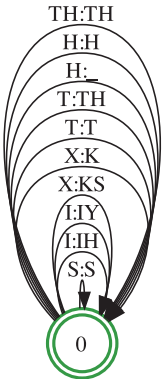


Fig. 12. (Colour online) Flower transducer F, suitable for mapping G↔P correspondences.

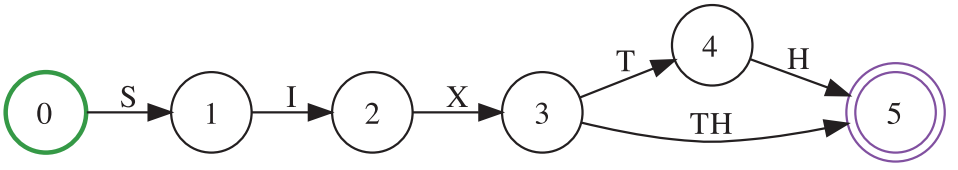


Fig. 13. (Colour online) FSA w', result of computing $RmEps(Project_o(w \circ C))$. This is utilized to expand grapheme clusters learned during the alignment process. It is necessary in order to guarantee that all valid pronunciation hypotheses can be generated.

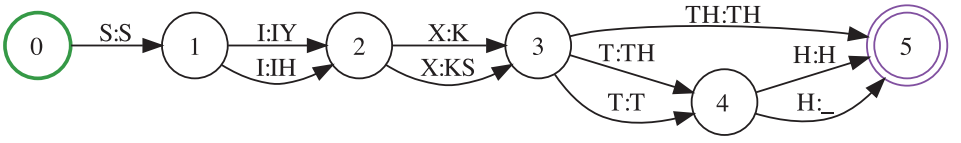


Fig. 14. (Colour online) FST w'', result of computing $RmEps(Min(Det(w' \circ F)))$.

If only one-to-one correspondences are permitted during the alignment process, then it is sufficient to simply use the linear FSA w. If clusters are permitted, then it will be necessary to utilize w'. If one prefers to interpret the back-off transitions in M as ϕ-transitions then it will be necessary to utilize w''. The latter case also requires modifying the input model according to one of the approaches described in 4.3.

If phoneme insertions are permitted, then it is also necessary to augment w with an insertion self-loop at each state, or to use one of the more sophisticated techniques described earlier.

6.2 Direct decoding with joint sequence RnnLMs

The structure of the joint sequence RnnLMs requires a specialized decoding approach (Auli *et al.* 2013). We have implemented an efficient decoder suitable for direct decoding of joint sequence RnnLMs, and released it as part of the Phonetisaurus toolkit. The decoder uses a strategy similar to the phrase-based decoder described in Auli *et al.* (2013). First, the input word is converted into an

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

equivalent FSA, in the same manner as the n-gram based decoder. Next, a vector of priority queues is initialized, one for each state in the linear input FSA. The input network is a directed, linear, acyclic acceptor, thus is it sufficient to cycle once through the list of states. The first priority queue in the vector is initialized with an empty token. Next, a token is created for each grapheme–phoneme combination leaving from the start state. Each token tracks the history, hidden-layer activations, total score, current label score and a pointer to the parent token. The score for the current arc is computed using the feed-forward formula described in Mikolov (2012). The tokens are inserted into the priority queue associated with the destination state of the current input arc, subject to a possible beam constraint. The decoding process then proceeds to the next state in the input FSA and begins exploring the set of tokens associated with the appropriate priority queue, in order of least total cost. The algorithm terminates when a final state is encountered, and returns the best hypothesis. In practice, it is only necessary to explore the first 15–20 tokens in the priority queues associated with each state, thus the overall decoding process, while slower, is still competitive with the joint n-gram solution, and does not suffer from an exponential explosion in the state space. The algorithm extends naturally to N-best results, and the implementation is released as a companion module to Phonetisaurus.

7 Experiments

In this section, we present experimental results covering alignment, standard n-gram models, direct decoding with joint sequence RnnLMs, model combination and finally summarize the results of independent third-party experiments.

7.1 Alignment experiments

Here, we present a short series of alignment evaluations using the proposed toolkit, exploring several basic parameters. These include whether or not deletions are allowed in the grapheme or phoneme sequences. The maximum subsequence length is fixed to two, and a joint maximization function is used. These settings were chosen after replicating the recommendations from Jiampojamarn *et al.* (2007).

Here, two popular English language pronunciation dictionaries are considered, the NETtalk dictionary (Sejnowski and Rosenberg 1993) and the CMUdict dictionary (Weide 1998). These two dictionaries are utilized throughout this work, primarily because they are open-source and available for download via the web. Furthermore, widely used training/testing splits exist for both of these dictionaries, making it straightforward to replicate and compare experiments from previous work. Details on the NETtalk 15k/5k split and the CMUdict 113k/13k split are reproduced from Bisani and Ney (2008) in Table 6. These are faithful replications of the splits from Chen (2003).

The results of applying different alignment parameters to these two test sets are displayed in Table 7. The purpose of these experiments was to determine what, if any impact the choice of alignment parameters might have on downstream accuracy

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Table 6. *Overview of the NETtalk (Sejnowski and Rosenberg 1993), and CMUdict (Weide 1998) dictionary splits from Bisani and Ney (2008)*

	# Symbols		Word length			Number of words	
Dictionary	G	P	G	P	Prons/word	Train	Test
NETtalk 15k/5k	26	50	7.3	6.2	1.010	14851	4951
CMUdict 113k/13k	27	39	7.5	6.3	1.062	106837	12000

Table 7. *Impact of allowing deletions (***del G**,**del P***), in the alignment lattice. Here,* **WER** *refers to Word Error Rate and* **PER** *to Phoneme Error Rate for the NETtalk 15k/5k split and the CMUdic 113k/13k split*

	Alignment evaluations			
Test set	del G	del P	WER(%)	PER(%)
NETtalk	Y	N	34.6	8.6
NETtalk	N	Y	34.6	**8.2**
NETtalk	Y	Y	**33.6**	**8.2**
CMUdic	Y	N	**25.6**	6.2
CMUdic	N	Y	25.7	**5.9**
CMUdic	Y	Y	25.8	7.0

of the G2P system. In order to focus on the impact of the alignment process, the parameters for the joint n-gram model were fixed. In both cases, the Google OpenGrm (Roark *et al.* 2012) toolkit was used to train the joint n-gram model, using standard Kneser–Ney smoothing. For the comparatively small NETtalk dataset, the order of the n-gram model was set to seven, and for the CMUdict to thirteen. Ultimately, the results indicate that the choice of deletion parameters has little impact on the downstream accuracy, however allowing deletions on both sides increases the computational complexity of the alignment algorithm. As a result, the choice was made to permit deletions on the grapheme side but not the phoneme side as the default configuration for further experiments. In Section 4, further discussion of language model tuning for G2P will illustrate several simple ways to further improve these results.

The **del G** and **del P** parameters indicate whether deletions were permitted in the grapheme and phoneme sequences respectively. A joint maximization was performed over the full set of joint G↔P correspondences. **WER** refers to Word Error Rate, which is $C(P_c)/C(W)$, the percentage of exactly correct pronunciations, $C(P_c)$, in full set of unique test words, $C(W)$. In the event that a test word has multiple correct pronunciations, the word is counted exactly once towards the total, and a pronunciation hypothesis is counted correct if it matches one of the acceptable variants. **PER** corresponds to Phoneme Error Rate. This is equivalent to WER for speech recognition, and is computed based on the Levenshtein distance, $PER = 1.0 - \frac{N-(S+D+I)}{N}$

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

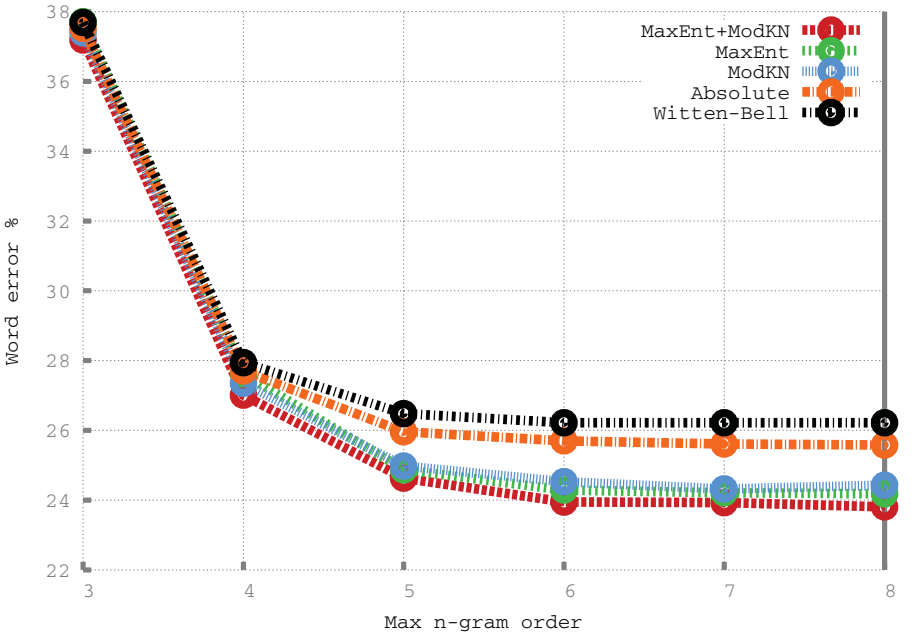


Fig. 15. (Colour online) Plot of WER vs. n-gram order for several different smoothing algorithms on the CMU dataset.

where N refers to the number of phonemes in the reference pronunciation, and S, D and I refer to substitutions, deletions, and insertions respectively.

7.2 *Smoothing experiments*

Here, we present a series of evaluations looking at the effect of different n-gram orders and smoothing techniques. In these experiments, the language models were all trained utilizing either the Google OpenGrm tools (Roark *et al.* 2012) or SRILM (Stolcke 2002). In particular, the system was evaluated using the Modified Kneser–Ney, Absolute and Witten–Bell smoothing techniques for n-gram orders 3–8 using the CMU dataset. In addition, the SRILM toolkit with the maximum-entropy extension was utilized to train a parallel maximum-entropy n-gram model, which was then interpolated with the Modified Kneser–Ney model at each n-gram order. Empirically, this last combination provides the best results. The WER versus n-gram order results for these experiments are plotted in Figure 15. These results indicate that accuracy gains plateau around $n = 6$. Independent results for several larger datasets, which are reproduced in Section 7.6 indicate that this plateau point is fairly consistent, at least for European languages.

In order to illustrate the scalability of the proposed toolkit and approach, we also conducted experiments using a much larger, proprietary Russian language lexicon, which is currently in use at Yandex as a component of their Automatic Speech Recognition system. The Russian language has a comparatively regular G2P relationship (Stüker and Schultz 2004), however the lexicon also contains a wide variety of proper names, abbreviations and acronyms. Details for this lexicon are

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Table 8. *Overview of the Yandex Russian training and testing lexica. Note: this is the first report on this particular dataset*

Dictionary	# Symbols		Word length		Prons/word	Number of words	
	G	P	G	P		Train	Test
Yandex-ru	34	49	11.0	10.6	1.01	1701679	188930

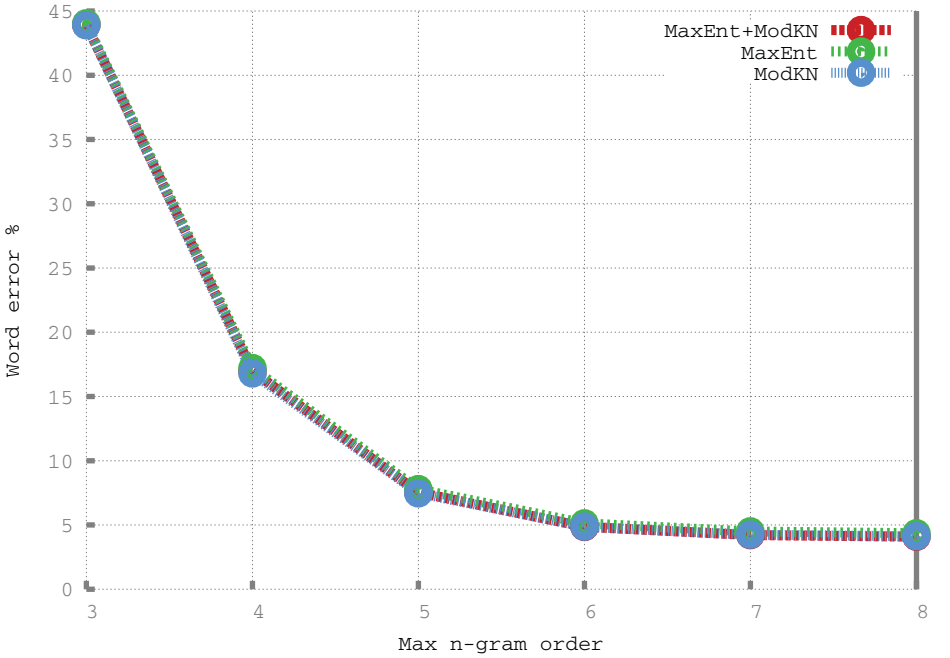


Fig. 16. (Colour online) Plot of WER vs. n-gram order for several different smoothing algorithms on the Yandex-ru dataset.

presented in Table 8. The training set for this lexicon contains roughly 1.7 million entries, while the test set contains approximately 188 thousand words. To the best of our knowledge, this is the largest test set for which any G2P results have previously been reported. This is also the first report on this particular dataset. Unfortunately, no other available open-source tools were efficient enough to scale to this task, so we do not have a comparison for this benchmark. The alignment and n-gram model training processes for this experiment utilized exactly the same parameters as were applied for the CMU evaluations. Based on the results from the CMU experiments, we decided to only train Modified Kneser–Ney, maximum-entropy, and an interpolated variant of the two for this experiment. The results for this experiment are depicted in Figure 16. In this case, the accuracy gains again plateau around $n = 6$. The minimum WER and PER achieved were 4.09 per cent and 0.64 per cent for the interpolated $MaxEnt + ModKN$ system, however there was not a significant WER difference in this case between the three evaluated systems. In terms of training time, it required approximately three hours to perform the alignment

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Table 9. *WER% for the CMUdic 113k/13k split using N-best alignments with* $1 \leq N \leq 8$*, Witten–Bell smoothing, and n-gram order seven. Here, dY means deletions were allowed on the output side only*

	CMUdic 113k/13k WER(%) versus N-best							
Aligner	1	2	3	4	5	6	7	8
dYdX	26.2	26.2	26.1	26.3	25.9	26.0	25.9	26.0
dY	26.2	26.2	26.1	26.0	25.9	26.0	25.9	26.0

step, five minutes to train the eight-gram Kneser–Ney model, and fifteen minutes to train the eight-gram maximum-entropy model. Decoding the test set required approximately eight minutes.

7.3 N-best alignments and fractional counts

The preceding experiments all focused on training joint n-gram models based on the one-best alignment for each word in the training corpus, however as mentioned in Section 3, `phonetisaurus-align` is also capable of outputting n-best alignments or full alignment lattices. In practice, most standard language modeling techniques focus on integral counts, and do not naturally generalize to the sort of fractional counts that are encoded in a lattice. The Witten–Bell smoothing technique does however generalize naturally to fractional counts and we present results looking at oracle n-best results using this technique and various values of n in Table 9. The purpose of these experiments was to ascertain what, if any potential there might be in further investigating system combination and reranking techniques.

The results from these experiments indicate that there is minor room for further improvement using the n-best alignments, assuming a effective reranking methodology, but that this tapers off around $N = 5$. Using the full lattices is also an option, however computing counts in this case quickly becomes very expensive, and the return in terms of accuracy improvements is likely to minimal or non-existent. Although the n-best results potentially provides minor accuracy improvements within Witten–Bell smoothing, even at $N = 5$ Modified Kneser–Ney using one-best alignments is still superior. The standard formulation for Kneser–Ney smoothing is based on integral counts, and thus cannot be directly applied to lattices, however in recent years, two generalizations to fractional counts have been proposed (Bisani and Ney 2008; Tam 2009).

The approach proposed in Tam (2009) is compared with Witten–Bell using bigram models in the context of latent semantic analysis, where they achieve small accuracy gains in the speaker adaptation domain. In our results, we note that for $n \leq 3$, Kneser–Ney smoothing only marginally outperforms Witten–Bell, whereas for higher order models the difference becomes quite significant. This may suggest that further gains could be expected from implementing fractional Kneser–Ney or extending the approach to multiple discount parameters. The approach proposed in Bisani and Ney (2008) is very similar, however in this case, the authors do not compare the

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Table 10. *G2P training values determined during brute force search*

Param	bptt	bptt-block	direct (M)	direct-order	hidden	class
Value	6	10	15	5	150	80

fractional method they propose to a standard Kneser–Ney or Modified Kneser–Ney joint n-gram model trained on one-best alignments. This may suggest that much of the gains from this approach derive from the implicit use of n-best alignments via lattice pruning. In any case, this implies that there is yet room for improvement using the proposed loosely-coupled approach via more sophisticated language modeling techniques.

7.4 Direct decoding with joint RnnLMs

In this section, we present G2P results using joint sequence RnnLMs directly to decode pronunciations for novel words. This is the first instance of this approach in the G2P literature. We describe the training and testing procedures below, and report results for both the CMUdict and the much larger Yandex-ru lexicon.

7.4.1 Brute force parameter search

In order to find reasonable parameters, the full CMUdict was randomly partitioned into a 10per cent test set and 90per cent training set. A further 10per cent of the training data was held out as validation data. Next, a series of brute force experiments were used to search the parameter space for the `-bptt`, `-bptt-block`, `-direct`, `-direct-order`, `-hidden` and `-class` parameters for the `rnnlm` tool. These determine how many time steps to propagate error backwards during BPTT training, at what step size to perform the propagation, the maximum number of direct connections, the maximum n-gram order for direct connections, the number of nodes in the hidden layer, and the number of classes respectively.

The best set of parameters, as determined by WER performance on the held out test set, were then used to train five networks using the standard test/train split described in Table 6. The training data was randomly shuffled for each training procedure, but all networks utilized the set of parameters described in Table 10. Finally, these parameters were also utilized to train five reversed models, as these have also been shown to perform well in practice (Schuster and Paliwal 1997). The same procedure was also used to train five forward and five backward models for the Yandex-ru lexicon.

The N-best oracle results (with $N = 1 - 5$) for the RnnLM experiments for the CMUdict are described in Table 11, and for the Yandex-ru lexicon in Table 12. The tables also illustrate best results from the preceding n-gram smoothing experiments for comparison. Finally, we present model combination and reranking results, as these have proven quite fruitful in this area in recent years (Hahn, Vozila and Bisani

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Table 11. *Oracle N-best CMUdict WER results for direct decoding with joint RnnLMs.*† *Averaged over five randomized trials*

	N-best				
Model	1	2	3	4	5
rnnlm†	25.0	15.4	12.4	10.8	09.9
n-gram	23.8	13.0	09.6	07.4	06.3
backwards-rnnlm†	24.9	14.6	11.0	09.2	08.1
backwards-n-gram	24.0	13.4	09.7	07.5	06.2
rnnlm-combined	23.6	12.9	09.3	07.2	06.1
n-gram-combined	23.8	12.9	09.5	07.2	06.0
all-combined†	**23.1**	**12.6**	**09.1**	**07.0**	**05.9**

Table 12. *Oracle N-best Yandex-ru WER results for direct decoding with joint RnnLMs.*† *Averaged over five randomized trials*

	N-best				
Model	1	2	3	4	5
rnnlm†	06.0	02.8	02.3	02.2	02.1
n-gram	04.3	01.7	00.8	**00.5**	**00.3**
backwards-rnnlm†	06.1	02.8	02.2	02.1	02.1
backwards-n-gram	04.2	01.7	00.8	**00.5**	00.4
rnnlm-combined	05.4	01.7	00.9	00.7	00.6
n-gram-combined	04.2	01.7	00.8	**00.5**	**00.3**
all-combined†	**04.0**	**01.4**	**00.7**	**00.5**	**00.3**

2012; Schlippe, Quaschningk and Schultz 2014; Cortesa, Kuznetsov and Mohri 2014).

For the reranking experiments, a simple formula was used to combine and re-rank N-best results for the various models. First, the the log posteriors for each N-best list, for each system, were normalized to sum to one. Next, a new score was computed for each unique hypothesis by summing the normalized scores across all N models according to the formula $h_{new} = \sum_n^N Score(h_n)$, where $Score(h_n)$ equals the normalized score for hypothesis h and model n, if the hypothesis was predicted by the model, and zero otherwise.

From the results in both tables, it is clear that, while the direct RnnLM results are competitive, they still fall consistently short of the best n-gram only models. The relative superiority of the n-gram models also increases as the size of the N-best increases. There is no statistically significant improvement with regard to the choice of backward and forward models, nevertheless combination of backward and forward models have been shown to be effective in previous works (Schuster and Paliwal 1997), and ensemble methods tend to provide strong and consistent improvements, especially when modeling techniques differ significantly (Schlippe

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Table 13. *N-best Yandex-ru WER results for forward n-gram models using various subsets of the training corpus. Results are averaged over five randomized trials*

Model	Percentage of training data				
	10%	25%	50%	75%	100%
n-gram WER	13.3	08.6	06.3	05.0	4.3

et al. 2014; Cortesa *et al.* 2014). Indeed, all three ensemble reranking results show consistent gains across all N-best orders and both test sets. The all-combined one-best results for the CMUdict also represent a new state-of-the-art on this data set.

The above model combination approach is surprisingly effective, however much more promising techniques are suggested in Cortesa *et al.* (2014), and we expect that these would further improve the above results. In addition, incorporating additional complementary models such as linear-chain conditional random fields (CRFs) (Wu *et al.* 2014) should again further improve the ensemble system. We plan to incorporate these ideas into future work.

The results on the Yandex lexicon indicate the utility of the proposed toolkit to extremely large pronunciation lexica. Nevertheless, another interesting question is whether or not all this training data is in fact necessary. For certain highly regular languages like Spanish or Italian, it is likely that WER/PER improvements plateau with much less data. In order to determine the case for the Yandex dataset, we also investigated the accuracy on the stated test set when utilizing an array of considerably smaller training subsets. In this case, we look only at the performance on the basic forward n-gram models. The results of these experiments are presented in Table 13.

From the table it is clear that, at least in the current case there is ample cause to utilize the full training dataset. While the basic rules are quite simple, the lexicon contains a large number of idiosyncratic entries, acronyms, names and other special words. This may account for the majority of the continued improvement.

7.5 Training and decoding efficiency

The proposed toolkit is both flexible and highly competitive in terms of accuracy, however perhaps its largest advantage is speed. Table 14 from Novak, Minematsu and Hirose (2012) compares training times for the proposed toolkit with previously reported results. The *m2m-fst-P* for system for the large 113k entry CMUdict training set requires just a tiny fraction of the training time. This turn-around time may be very important for rapid system development. The RNNLM rescoring approach requires more time, however it is still significantly faster than other available options.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Table 14. *Training times for the smallest (15k entries) and largest (113k entries) training sets (Novak et al. 2012)*

System	NETtalk-15k	CMUdict
Sequitur (Bisani and Ney 2008)	Hours	Days
direcTL+ (Jiampojamarn and Kondrak 2010)	Hours	Days
m2m-P	2m56s	21m58s
m2m-fst-P	1m43s	13m06s
rnnlm-P	20 m	2 h

7.6 Independent experiments

The proposed toolkit has recently been independently evaluated by several third-party groups in industry and academia. In particular, a series of experiments using large scale pronunciation dictionaries for several different European languages, and comparing a selection of different industry and open-source tools was recently reported in Hahn *et al.* (2012). These experiments were conducted without any input from the authors of this work, however we mention the results here as they represent a completely independent evaluation, which illustrates the flexibility, competitive accuracy and real-world utility of the proposed toolkit and the approach that it embodies.

The experiments in Hahn *et al.* (2012) investigated five different European languages including English, German, French, Italian and Dutch, using large-scale industry pronunciation dictionaries, which all contained over 200k unique words. Six different G2P systems from industry and academia were then evaluated on these dictionaries.

The results of these independent experiments clearly showed that the proposed system is highly competitive with other available industry standard tools. In particular, it consistently outperforms all but `seq` on each of the large scale datasets, even using the default decoder setup and a relatively simple language model. The performance is also competitive with `seq`.

Experimental results with the CMU dictionary were also recently successfully replicated in Wu *et al.* (2014), and the toolkit has been compared and utilized successfully in the ensemble methods described in Schlippe *et al.* (2014).

8 Conclusions

In this work, we have presented the open-source G2P conversion toolkit, Phonetisaurus.

The toolkit provides a variety of standalone, loosely-coupled applications, which can be used to rapidly train sophisticated, high quality G2P conversion models suitable for use in a variety of speech applications.

This work provided detailed discussion on theoretical and practical issues in this area, as well as several novel contributions. A variety of standard language modeling techniques and tools were compared on an equal footing in the context of joint n-gram G2P, using well-known, open-source pronunciation dictionaries.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Detailed discussion of the decoding framework for WFST-based G2P with joint n-gram models was also provided. This also included a decoder suitable for efficient direct decoding of joint sequence RnnLMs.

A suite of experimental results were provided, including the first instance in the literature of using joint sequence RnnLMs for G2P conversion. Ensemble results achieved a new state-of-the-art WER on the well-known CMUdict dataset. In future work in this area, we plan to explore the incorporation of linguistically motivated maximum entropy features as well as the possibility of bi-directional models such as those described in Schuster and Paliwal (1997).

Independent experimental results as well as preliminary exploratory experimental results for two decoder extensions were also provided. These results illustrated that the proposed toolkit is highly competitive with or superior to the leading solutions in both industry and academia. Experiments illustrating the speed and efficiency of the proposed toolkit were also provided, showing that it dramatically outperforms other solutions in from this perspective.

In the past, many publications have been made describing different joint n-gram models in the context of G2P conversion, however it was difficult to compare these on an even footing due to the lack of a common framework. The loosely-coupled nature of the proposed system, and the ability to directly leverage other mature tools from the Statistical Language Modeling community have made it possible to conduct such evaluations. Furthermore, standard approaches to language model combination, interpolation, class-based models and rescoring are all applicable to the G2P domain, and in many cases may be explored without any direct modification to the proposed toolkit.

The WFST framework also provides a unified ecosystem suitable for incorporating rule-based or language-specific constraints, such as those discussed in Caseiro and Trancoso (2002). This is another area ripe for further exploration. Furthermore, the incorporation of additional model types such as linear-chain CRFs (Wu *et al.* 2014), and more advanced ensemble techniques (Schlippe *et al.* 2014; Cortesa *et al.* 2014) stand to significantly improve the overall quality of the system.

Finally, in recent years other evaluation metrics for G2P have been proposed (Hixon, Schneider and Epstein 2011), which may correlate better with downstream model usage. Further investigation, implementation and combination of these methods may stand to further improve the validity and utility of results in this area.

The proposed toolkit is released under the BSD license, and is freely available for download (Novak 2011). It is our hope that it will continue to enjoy use among speech researchers in industry and academia, and help to further promote innovation in G2P conversion and related areas. Bug reports, fixes and contributions are also welcome.

Acknowledgments

We would like to thank the speech team at Yandex for their help and support in this work.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

References

Allauzen, C., Mohri, M., and Roark, B. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting of the Assocication for Computational Linguistics*, Stroudsburg, PA, USA, pp. 40–7.

Allauzen, C., Riley, M., Schalkwyk, J., Wojciech, S., and Mohri, M. 2007. OpenFst: a general and efficient weighted finite-state transducer library. In *Proceedings of CIAA 2007*, pp. 11–23, Lecture Notes in Computer Science, vol. 4783. Berlin Heidelberg: Springer.

Alumäe, T., and Kurimo, M. 2010. Efficient estimation of maximum entropy language models with N-gram features: an SRILM extension. In *Proceedings of Interspeech 2010*, Chiba, Japan.

Auli, M., Galley, M., Quirk, C., and Zweig, G. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of EMNLP 2013*, Melbourne, Australia, pp. 1044–54.

Bell, T., Cleary, J., and Witten, I. 1990. *Text Compression*. Upper Saddle River, NJ, USA: Prentice Hall.

Bisani, M., and Ney, H. 2008. Joint-sequence models for grapheme-to-phoneme conversion. In *Speech Communication*, pp. 434–51. Amsterdam: Elsevier Science Publishers B. V.

Caseiro D., and Trancoso, I. 2002. Grapheme-to-Phoneme using finite state transducers. In *Proceedings of the 2002 IEEE Workshop on Speech Synthesis*, Piscataway NJ, USA.

Chen, S. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Proceedings of EUROSPEECH*.

Chen, S., and Goodman, J. 1998. An empirical study of smoothing techniques for language modeling. Technical Report, Computer Science Group, Harvard Univerisity.

Cortes, C., Kuznetsov, V., and Mohri, M. 2014. Ensemble methods for structured prediction. In *Proceedings of ICML 2014*, Bonn, Germany, pp. 896–903.

Damper, R., Marchand, Y., Adsett, C., Soonklang, T., and Marsters, S. 2005. Multilingual data-driven pronunciation. In *Proceedings of the 10th International Conference on Speech and Computer (SPECOM 2005)*, Patras, Greece, pp. 167–70.

Deligne, S., Yvon, F., and Bimbot, F. 1995. Variable-length sequence matching for phonetic transcription using joint multigrams. In *Proceedings of EUROSPEECH 1995*, Madrid, Spain, pp. 2243–46.

Galescu, L., and Allen, J. F. 2001. Bi-directional conversion between graphemes and phonemes using a joint n-gram model. In *Proceedings of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Perthshire, Scotland.

Hahn, S., Vozila, P., and Bisani, M. 2012. Comparison of grapheme-to-phoneme methods on large pronunciation dictionaries and LVCSR tasks. In *Proceedings of INTERSPEECH 2012*, Portland, Oregon.

Hixon, B., Schneider, E., and Epstein, S. 2011. Phonemic similarity metrics to compare pronunciation methods. In *Proceedings of INTERSPEECH 2011*, Florence, Italy, pp. 825–8.

Jiampojamarn, S., and Kondrak, G. 2010. Letter-to-phoneme alignment: an exploration. In *Proceedings of the ACL 2010*, Uppsala, Sweden, pp. 780–8.

Jiampojamarn, S., Kondrak, G., and Sherif, T. 2007. Applying many-to-many alignments and hidden Markov models to letter-to-phoneme conversion. In *Proceedings of NAACL HLT 2007*, Rochester, New York, pp. 372–9.

Kempe, A. 2001. Factorization of ambiguous finite-state transducers. In *Proceedings of CIAA 2001*, Pretoria, South Africa, pp. 170–81.

Kneser, R., and Ney, H. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 1995*, Detroit, Michigan, pp. 1:181–4.

Mikolov, T. 2012. *Statistical Language Models Based on Neural Networks*. PhD Thesis, Brno University of Technology, Czech republic.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Mikolov, T., Karafiat, M., Burget, L., Černocký, J., and Khundanpur, S. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH 2010*, Chiba, Japan, pp. 1045–8.

Mikolov, T., Kombrink, S., Anoop, D., Burget, L., and Černocký, J. 2011. RNNLM - recurrent neural network language modeling toolkit. In *ASRU 2011, demo session*, Waikoloa, Hawaii.

Mohri, M. 2002. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics* **7**(3): 321–50. Magdeburg: Otto-von-Guericke-Universitat.

Mohri, M., Pereira, F., and Riley, M. 2002. Weighted finite-state transducers in speech recognition. *ComputerSpeech and Language* **16**(1): 69–88. Elsevier.

Novak, J. (2011) Available at: http://code.google.com/p/phonetisaurus.

Novak, J., Dixon, P., Minematsu, M., Hirose, K., Horie, C., and Kashioka, H. 2012. Improving WFST-based G2P Conversion with alignment constraints and RNNLM N-best rescoring. In *Proceedings of INTERSPEECH 2012*, Portland, Oregon, pp. 2526–9.

Novak, J., Minematsu, M., and Hirose, K. 2012. WFST-based grapheme-to-phoneme conversion: open source tools for alignment, model-building and decoding. In *Proceedings of FSMNLP 2012*, San Sebastian, Spain, pp. 45–9.

Novak, J., Minematsu, M., and Hirose, K. 2013. Failure transitions for Joint n-gram models and G2P conversion. In *Proceedings of INTERSPEECH 2013*, Lyon, France, pp. 1821–5.

Roark, B., Sproat, R., Allauzen, C., Riley, M., Sorensen, J., and Tai, T. 2012. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 - System Demonstrations*, Jeju, South Korea, pp. 61–6.

Ristad, E., and Yianilos, P. 1998. Learning string edit distance. *IEEE Transactions PRMI* **20**(5): 522–32.

Schlippe, T., Quaschningk, W., and Schultz, T. 2014. Combining grapheme-to-phoneme converter outputs for enhanced pronunciation generation in low-resource scenarios. In *Proceedings of the 4th Workshop on Spoken Language Technologies for Under-resourced Languages*, St. Petersburg, Russia, pp. 14–16.

Schuster, M., and Paliwal, K. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* **45**(11): 2673–81.

Sejnowski, T. J., and Rosenberg, C. R. 1993. NETtalk corpus. Available at: ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionar- ies/beep.tar.gz.

Shu, H., and Hetherington, I. 2002. EM training of finite-state transducers and its application to pronunciation modeling. In *Proceedings of ICSLP 2002*, Denver, Colorado, pp. 1293–6.

Stolcke, A. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of ICSLP 2002*, Denver, Colorado, pp. 901–4.

Stüker, S., and Schultz, T. 2004. A grapheme based speech recognition system for Russian. In *Proceedings of SPECOM*, St. Petersburg, Russia, pp. 297–303.

Tam, Y. 2009. *Rapid Unsupervised Topic Adaptation - A Latent Semantic Approach*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

Weide, R. L. 1998. The Carnegie Mellon pronouncing dictionary. Available at: http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Witten, I., and Bell, T. 1991. The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory* **37**(4): 1085–94. IEEE Transactions on Information Theory.

Wu, J. 2002. *Maximum Entropy Language Modeling with Non-Local Dependencies*. PhD thesis, Baltimore, Maryland, USA.

Wu, K., Allauzen, C., Hall, K., Riley, M., and Roark, B. 2014. Encoding linear models as weighted finite-state transducers. In *Proceedings of INTERSPEECH 2014*, Singapore, pp. 1258–62.

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Appendix A: Tool usage

This section provides several basic usage examples for the main tools provided by the toolkit. Most of the tools contain a wide variety of additional, tunable parameters. The interested reader is referred to the Phonetisaurus toolkit homepage (Novak 2011) for more detailed usage instructions and tutorials. The downloads section also includes self-contained experiments suitable for reproducing results from previous publications.

Align a dictionary:

```
$ phonetisaurus-align --input=test.dic --ofile=test.corpus \
    --seq1_del=false
```

Train a 7-gram model with mitlm:

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

Train a 7-gram model with OpenGrm:

```
$ ngramsymbols < test.corpus > test.syms
$ farcompilestrings --symbols=test.syms --keep_symbols=1 \
    test.corpus > test.far
$ ngramcount --order=7 test.far > test.cnts
$ ngrammake --v=2 --bins=3 --method=kneser_ney \
    test.cnts > test.mod
$ ngramprint --ARPA test.mod > test.arpa
```

Train a 7-gram MaxEnt model with SRILM:

```
$ ngram-count -text test.corpus -maxent-lm test-me.arpa \
    -order 7 -maxent-convert-to-arpa
```

Build an interpolated ME+KN model with SRILM:

```
$ ngram -order 7 -mix-lm test.arpa -lm test.arpa \
    -write-lm test-mix.arpa -lambda .5
```

Convert the model to a failure-transition compatible WFST

```
$ phonetisaurus-arpa2fst-omega --lm=test.arpa --ofile=test-f.fst
```

Apply the failure-transition decoder

```
$ phonetisaurus-g2p-omega --model=test-f.fst--decoder_type=fst_phi \
   --input=abbreviate --nbest=3
    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
```

Train a jont sequence RnnLM

```
$ rnnlm -train test.train -valid test.valid -rnnlm test.rnnlm  \
   -independent -binary -bptt 6 -bptt-block 10 -direct 15 \
   -direct-order 5 -hidden 150 -class 80
```

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press

Apply the joint sequence RnnLM decoder

```
$ phonetisaurus-g2prnn --rnnlm=test.rnnlm --test=<(echo "TESTING") \
   --nbest=3 | ./prettify.pl
    TESTING     T EH S T IH NG  11.7486
    TESTING     T EH S T AH NG  18.5673
    TESTING     T EH S T IY NG  20.3235
```

https://doi.org/10.1017/S1351324915000315 Published online by Cambridge University Press