LOW-RESOURCE GRAPHEME-TO-PHONEME CONVERSION USING RECURRENT NEURAL NETWORKS

Preethi Jyothi[†] *and Mark Hasegawa-Johnson*[§]

[†] Indian Institute of Technology Bombay, India
[§]University of Illinois at Urbana-Champaign, USA

ABSTRACT

Grapheme-to-phoneme (G2P) conversion is an important problem for many speech and language processing applications. G2P models are particularly useful for low-resource languages that do not have well-developed pronunciation lexicons. Prominent G2P paradigms are based on initial alignments between grapheme and phoneme sequences. In this work, we devise new alignment strategies that work effectively with recurrent neural network based models when only a small number of pronunciations are available to train the models. In a small data setting, we build G2P models for Pashto, Tagalog and Lithuanian that significantly outperform a joint sequence model and a baseline recurrent neural network based model, giving up to 14% and 9% relative reductions in phone and word error rates when trained on a dataset of 250 words.

Index Terms— grapheme-to-phoneme conversion, low-resource languages, recurrent neural network models

1. INTRODUCTION

The grapheme-to-phoneme (G2P) conversion task can be defined as follows: Given a grapheme sequence representing a word, what is its corresponding most likely pronunciation (i.e. phoneme sequence)? While some languages like Spanish and Haitian Creole have systematic grapheme to phoneme mappings, many languages like English, French and Arabic have irregular G2P correspondences, thus making pronunciation harder to model.

G2P models are routinely employed in speech synthesis systems and automatic speech recognition (ASR) systems. G2P models are essential for ASR systems in languages that do not have pre-built dictionaries to train pronunciation models. Even when pronunciation lexicons are available, G2P models can be used to derive pronunciations for out-of-vocabulary (OOV) words that do not appear in the dictionary.

Related Work: In the literature on G2P conversion, joint sequence models are a popular paradigm [1, 2, 3, 4]. In this class of models, grapheme and phoneme sequences are first aligned to form a sequence of joint G-P tokens (called graphones). A joint N-gram model trained on graphone sequences can then be used to predict pronunciations for new words. In recent work, deep neural network-based approaches have emerged as the new state-of-the-art on standard English G2P tasks [5, 6, 7]. Prior work on neural network-based models for G2P relied on sufficiently large dictionaries to train the G2P models, which are hard to obtain for low-resource languages. In prior work, pronunciation models for low-resource languages were built using semi-automatic bootstrapping algorithms [8, 9, 10] or acoustic data-driven methods [11, 12]. In this work, we explore how to effectively train neural models for G2P in low-resource settings.

Our contribution: In low-resource settings, representations are crucial for neural network models much more so than when large amounts of data are available. In this work, our contribution is to develop such representations that help us significantly outperform state-of-the-art G2P systems when limited training data is available. Specifically, we designed novel representations for aligned grapheme-phoneme sequences which are more amenable to learning using recurrent neural networks (RNN). We evaluate our techniques on three different languages and show consistent improvements in G2P performance with small amounts of training data.

2. USING ALIGNMENTS IN G2P FOR SMALL DATA

Given the grapheme sequence of a word *and* the phoneme sequence corresponding to its pronunciation, an alignment strategy associates short segments (often of length 1) of the first sequence with short segments of the second sequence. A phonologically meaningful alignment strategy can typically be learned from data using an expectation-maximization (EM) based algorithm [2]. Some of the best G2P solutions depend on such alignment strategies [3, 4, 6].

A challenge in exploiting alignments for G2P, especially in RNN-based models, is that the alignment information is available only for the training data, and is not available during test time. Note that multiple graphemes may align with a single phoneme and vice-versa, and during decoding the given grapheme sequence does not indicate such groupings. This can be remedied by requiring the alignment to be a "1-to-2" alignment, where every grapheme in the input sequence is aligned to a single phoneme, a phone bigram or an empty symbol (as in [6]). We will refer to this alignment as BIGRAM-ALIGN. Here, the grapheme sequence remains unchanged, using only single graphemes, and without any empty symbol or grapheme multi-grams.

Limitations of BIGRAM-ALIGN: We observed two limitations of BIGRAM-ALIGN when dealing with small data. Firstly, using phone bigrams as part of the output alphabet leads to insufficient coverage of the output alphabet in the training data. Secondly, alignments that use phone bigrams are more error-prone when the aligner is trained on a small amount of data. For example, a Pashto word containing three graphemes $\mathtt{g}_1\ \mathtt{g}_2\ \mathtt{g}_3$, pronounced as "m u z e ɾ", should be aligned with the graphemes as $(\mathtt{m}+\mathtt{u}), (\mathtt{z}+\mathtt{e}), (\mathtt{ɾ})$. However, in a small data setting, we observed that the word gets incorrectly aligned as $(\mathtt{m}), (\mathtt{u}+\mathtt{z}), (\mathtt{e}+\mathtt{ɾ})$. Both these issues can adversely affect the accuracy of the RNN models.

Alignment INTER-ALIGN: To address the above issues, we propose a new alignment representation INTER-ALIGN that avoids the use of phoneme bigrams. A key requirement of an alignment representation is that it should be possible to deduce the grapheme part of the alignment from the original grapheme sequence alone. In the case of BIGRAM-ALIGN, the grapheme part of the alignment was the original grapheme sequence itself. In our case, instead, it will be the original grapheme sequence *interleaved* with a new symbol #. Intuitively, this extra spacing allows up to two phonemes to be aligned with each grapheme, without the need for combining them into a bigram symbol.

Below, we describe in more detail how an INTER-ALIGN alignment is obtained. First, we find a one-to-one alignment between the grapheme and phoneme sequences in the training data. For example, suppose a grapheme sequence "$\mathtt{g}_1\ \mathtt{g}_2\ \mathtt{g}_3$" and a phoneme sequence "$\mathtt{p}_1\ \mathtt{p}_2\ \mathtt{p}_3$" are aligned as follows:

$$\begin{array}{cccc} \mathtt{g}_1 & \mathtt{g}_2 & - & \mathtt{g}_3 \\ - & \mathtt{p}_1 & \mathtt{p}_2 & \mathtt{p}_3 \end{array} \tag{1}$$

where $-$ indicates an empty symbol.

We further modify the aligned grapheme sequence by inserting a new symbol (call it "#") between every pair of consecutive grapheme symbols other than the empty symbol, and also at the beginning if the aligned grapheme sequence does not begin with the empty symbol. (We use end-of-sequence markers to pad the grapheme/phoneme sequences to a fixed length; this precludes the need to insert # at the end of the grapheme sequence.) Also, we modify the phoneme sequence by inserting "#" for every "#" seen in the grapheme sequence, as shown in the example below:

$$\begin{array}{cccccc} \# & \mathtt{g}_1 & \# & \mathtt{g}_2 & - & \mathtt{g}_3 \\ \# & - & \# & \mathtt{p}_1 & \mathtt{p}_2 & \mathtt{p}_3 \end{array} \tag{2}$$

This yields a phoneme sequence which will be aligned against a fixed representation of grapheme sequences, obtained by interleaving the grapheme sequence with #, as shown below.

$$\begin{array}{cccccc} \# & \mathtt{g}_1 & \# & \mathtt{g}_2 & \# & \mathtt{g}_3 \\ \# & - & \# & \mathtt{p}_1 & \mathtt{p}_2 & \mathtt{p}_3 \end{array} \tag{3}$$

We point out that grapheme sequence in (3) is the same as that in (2) if the $-$ in the latter is replaced by the # symbol. This will be the case as long as there are no two consecutive empty symbols or a trailing empty symbol in the original aligned grapheme sequence (as in (1)).[1]

Alignment PAIR-ALIGN: We also consider a modification of INTER-ALIGN, called PAIR-ALIGN, in which the grapheme sequence is not interleaved with # symbols, but instead we align each grapheme against a pair of phoneme symbols. For example, instead of the alignment in (3), $\mathtt{g}_1\ \mathtt{g}_2\ \mathtt{g}_3$ will be aligned with $(\#, -)\ (\#, \mathtt{p}_1)\ (\mathtt{p}_2, \mathtt{p}_3)$. Note that this is different from BIGRAM-ALIGN in that instead of using unigram and bigram phoneme symbols, each grapheme is always aligned with a pair of unigram phonemes. In the RNN models discussed in the next section, this difference is significant.

3. RNN-BASED G2P MODELS FOR SMALL DATA

Figure 1(A) shows the architecture of a baseline RNN model adapted from [6]. Each input $\mathbf{x}_t$ at time step t is fed as input to a recurrent hidden node whose state $\mathbf{h}_t$ is a function of $\mathbf{x}_t$ and the state from the previous time step $\mathbf{h}_{t-1}$. We use bidirectional hidden layers [13] in BIGRAM-ALIGN; Figure 1(A) shows a stack of two bidirectional hidden layers unrolled over three time-steps. Each bidirectional layer contains two hidden layers where one layer has recurrent connections from past time-steps (going left to right) and the other has connections in the backward direction (going right to left). Activations from both layers are concatenated and passed to subsequent layers as input. Outputs from the topmost hidden layer in the stack are projected to the target space (using a layer of nodes shown as triangles).[2] Following [6], each node in the forward layer of the topmost bidirectional layer also receives the output prediction from the previous time-step as input.

This baseline model uses output aligned according to the BIGRAM-ALIGN representation. Thus the output alphabet consists of the empty symbol, individual phonemes and phoneme bigrams. Figure 1(B) shows the RNN model adapted to use INTER-ALIGN where the output alphabet consists of the empty symbol and individual phonemes. And, Figure 1(C) shows the RNN model used with PAIR-ALIGN where outputs from the topmost hidden layer in the stack are duplicated and projected to a pair of unigram phonemes.

[1]In our experiments, about 1% of the training utterances did not conform to this condition. In these cases, for the training grapheme sequences the fixed representation with alternating # symbols was used, resulting in a slight misalignment with the phoneme sequences.

[2]The output projection layer is preceded by a recurrent layer in [6]. In our experiments, we did not include this recurrent layer as this had no significant effect in the final error rates.

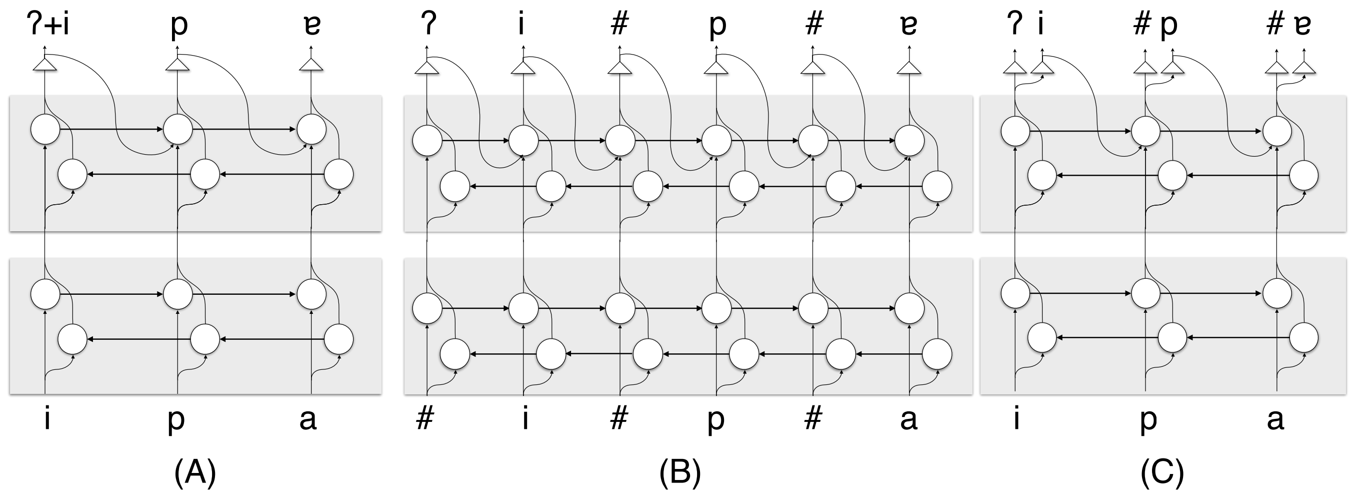

Fig. 1. (A) Baseline RNN architecture used with BIGRAM-ALIGN. "ʔ+i" denotes a phone bigram aligned with the grapheme "i". (B) RNN architecture used with INTER-ALIGN. (C) RNN architecture used with PAIR-ALIGN.

4. EXPERIMENTAL DETAILS

4.1. Data description

We used pronunciation lexicons for three different languages from language collections that were released as part of the IARPA Babel program [14]: Pashto, Tagalog and Lithuanian. We preprocessed the lexicons to remove all syllable break and word boundary markers from the pronunciations.

We simulated low-resource conditions for each language by randomly sampling a small subset of words (of size N) in order to train our G2P models. We assume that the grapheme and phoneme vocabularies for each language are known a priori. Using this information, we randomly sample the pronunciation lexicon and construct a minimal set of words that fully cover the grapheme and phoneme vocabularies i.e. each grapheme and phoneme appears in at least one word. Assuming the minimal set contains $\bar{N}$ words, we randomly sample the remaining $N - \bar{N}$ words from the pronunciation lexicon to create our training data. In this manner, we construct training sets containing 250, 500 and 1000 words for each of the three languages; we will refer to these training conditions as T-250, T-500 and T-1000. We also construct development and evaluation sets for each language; word and phone counts corresponding to these data splits are detailed in Table 1.

PASHTO	DEV	1400 words	8709 phones
(50 graphemes, 44 phonemes)	EVAL	1454 words	8908 phones
TAGALOG	DEV	1678 words	11802 phones
(53 graphemes, 38 phonemes)	EVAL	1598 words	11419 phones
LITHUANIAN	DEV	1697 words	12797 phones
(58 graphemes, 89 phonemes)	EVAL	1997 words	15317 phones

Table 1. Statistics of development/evaluation data.

4.2. RNN implementation

We used 3-layer bi-directional Long Short-term Memory (LSTM) cells [15] for all three RNN models: BIGRAM-ALIGN, PAIR-ALIGN and INTER-ALIGN. Alignments to train the BIGRAM-ALIGN model were generated using the *m2m-aligner* software package [2]. Alignments for both the interleaved RNN models PAIR-ALIGN and INTER-ALIGN were constructed using the alignment module in the Phonetisaurus toolkit [4]. All the RNN hyperparameters were determined by tuning on the development set of Lithuanian; this setting was then used for the other two languages.

We used 3-layer bidirectional LSTM models, with a 32-dimensional projection layer to encode the input sequences and 256 nodes in each hidden layer. We used the Adam optimizer [16] to learn the weights of the network with a default learning rate of 0.001. All the RNN models were implemented using TensorFlow [17].

5. RESULTS AND DISCUSSION

Table 2 shows phone error rates (PER) and word error rates (WER) for all three training conditions using the three RNN models described in Section 3. From Table 2, we observe that RNN models using the interleaved alignment strategies (PAIR-ALIGN and INTER-ALIGN) perform consistently better than BIGRAM-ALIGN.

In the training condition T-250, INTER-ALIGN significantly outperforms (at $p < 0.01$) the other two alignment strategies in all three languages. (Statistical significance was computed using the MAPSSWE test, which is implemented in the NIST Scoring Toolkit.) For instance, from Table 2, for T-250 there is a relative PER drop of 14% for Lithuanian and a relative WER drop of 9% for Tagalog on the evaluation sets. We also observe that the advantage of using INTER-

Lang	System	T-250				T-500				T-1000			
		Dev		Eval		Dev		Eval		Dev		Eval	
		PER	WER	PER	WER	PER	WER	PER	WER	PER	WER	PER	WER
Pashto	bigram-align	25.87	80.50	26.48	80.47	18.34	66.79	19.40	69.67	14.90	56.86	16.23	61.07
	pair-align	19.66	69.07	20.73	70.98	15.57	59.00	16.74	63.27	14.16	56.21	14.98	57.08
	inter-align	**17.92***	**64.86***	**18.21***	**66.64***	**14.81***	**57.21***	**14.99***	**58.94***	**13.63***	**54.57**	**13.81***	**51.65***
Tagalog	bigram-align	15.15	61.03	15.46	61.20	11.69	51.01	11.90	52.50	9.34	43.33	**9.34**	**44.62**
	pair-align	13.23	55.78	13.67	57.26	11.12	50.72	**11.16**	**49.50**	**9.16**	**43.27**	9.44	46.31
	inter-align	**12.86***	**55.24***	**13.01***	**56.26***	**10.82**	**50.06**	11.32	51.44	9.17	44.28	9.36	45.81
Lithuanian	bigram-align	13.54	71.01	13.50	69.70	10.40	60.70	10.32	59.94	8.77	52.03	9.07	53.33
	pair-align	12.60	68.71	12.59	69.00	10.10	58.81	10.45	59.89	8.87	51.50	9.15	52.83
	inter-align	**11.71***	**65.70***	**11.68***	**65.25***	**9.68**	**57.63**	**9.92**	**59.49**	**8.60**	**50.50**	**8.90**	**51.93**

Table 2. PER/WER results for three languages in three training conditions. The best numbers for each training condition in each language are in bold. Systems that significantly outperform the other two (at $p < 0.01$) are marked with an asterisk.

Lang	System	T-250				T-500				T-1000			
		Dev		Eval		Dev		Eval		Dev		Eval	
		PER	WER	PER	WER	PER	WER	PER	WER	PER	WER	PER	WER
Pashto	Psaurus	19.32	70.50	19.31	69.81	15.86	61.86	16.61	61.83	14.56	55.93	14.67	56.05
	inter-align	17.92*	64.86*	18.21*	66.64*	14.81*	57.21*	14.99*	58.94*	13.63*	54.57*	13.81*	51.65*
	hybrid	**17.28***	**64.64***	**17.33***	**64.99***	**13.95***	**56.07***	**14.52***	**57.29***	**13.01***	**52.93***	**13.10***	**49.93***
Tagalog	Psaurus	13.43	58.58	13.59	58.95	11.49	52.62	11.43	53.00	9.91	46.01	9.68	47.12
	inter-align	12.86	55.24	13.01	56.26	10.82	50.06	11.32	51.44	9.17	44.28	9.36	45.81
	hybrid	**12.24***	**53.99***	**12.23***	**53.82***	**10.51***	**48.57***	**10.94***	**50.38***	**8.63***	**41.42***	**8.91***	**43.87***
Lithuanian	Psaurus	16.62	73.90	17.19	76.11	13.22	64.88	13.40	66.45	10.82	57.57	11.20	59.69
	inter-align	11.71*	65.70*	11.68*	65.25*	9.68*	57.63*	9.92*	59.49*	8.60*	50.50*	8.90*	51.93*
	hybrid	**11.40***	**64.70***	**11.66***	**64.90***	**9.54***	**57.22***	**9.77***	**59.19***	**8.39***	**50.38***	**8.80***	**52.13***

Table 3. Comparison of psaurus with inter-align and a hybrid system. Lowest error rates for each language in each training condition are shown in bold. Asterisks indicate statistically significant improvement over psaurus.

align over bigram-align diminishes as we increase the number of training instances from T-250 to T-1000. We expect bigram-align to perform better with more training data since the alignments from the m2m-aligner get more reliable as the number of training instances increase.

inter-align significantly outperforms bigram-align in all three training conditions for Pashto. One potential reason for the poor performance of bigram-align could be due to the absence of vowel markers in Pashto. The Arabic script used for Pashto indicates long vowels but does not explicitly indicate short vowels in its orthography. This leads to erroneous alignments when there is less data to learn from.

Table 3 shows a joint sequence 5-gram model trained using the Phonetisaurus toolkit [4] (psaurus) in comparison with inter-align.[3] inter-align significantly outperforms psaurus in all training conditions (at $p < 0.001$) for Pashto and Lithuanian. We also show a hybrid system that combines the N-gram model in psaurus with inter-align by representing both models as finite state machines and composing them. We scale the N-gram model before composition; the scaling factor was tuned on the development set for each language. The hybrid system further lowers error rates across all the training conditions for all three languages.

[3]Higher order (>5) N-gram models for psaurus did not improve the error rates any further.

6. CONCLUSIONS

In this work, we demonstrate that in low-resource settings, seemingly simple alterations to the representation of alignments in RNN G2P models yield significant improvements in error rates over state-of-the-art baselines. This points to the need for further research into designing representations that improve the performance of neural network models when low amounts of training data are available.

7. ACKNOWLEDGEMENTS

A significant part of this research was carried out while the first author was at the Beckman Institute, UIUC, and was funded in part by DARPA LORELEI grant HR0011-15-2-0043. Pronunciation lexicons from the IARPA Babel Program IARPA-babel106-v0.2g (Tagalog), IARPA-babel104b-v0.4bY (Pashto) and IARPA-babel304b-v1.0b (Lithuanian) were used.

8. REFERENCES

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

[2] S. Jiampojamarn, G. Kondrak, and T. Sherif, "Applying many-to-many alignments and hidden Markov models to letter-to-phoneme conversion.," in *Proceedings of NAACL HLT*, 2007.

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

[4] J. R. Novak, N. Minematsu, K. Hirose, C. Hori, H. Kashioka, and P. R. Dixon, "Improving WFST-based G2P Conversion with Alignment Constraints and RNNLM N-best Rescoring.," in *Proceedings of Interspeech*, 2012.

[5] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *Proceedings of ICASSP*, 2015.

[6] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," in *Proceedings of Interspeech*, 2015.

[7] S. Toshniwal and K. Livescu, "Jointly learning to align and convert graphemes to phonemes with neural attention models," in *Proceedings of IEEE Workshop on Spoken Language Technology (SLT)*, 2016.

[8] S. Maskey and A. W. Black, "Boostrapping phonetic lexicons for new languages," in *Proceedings of Interspeech*, 2004.

[9] M. Davel and O. Martirosian, "Pronunciation dictionary development in resource-scarce environments," in *Proceedings of Interspeech*, 2009.

[10] S. S. Juan, L. Besacier, and S. Rossato, "Semi-supervised G2P bootstrapping and its application to ASR for a very under-resourced language: Iban," in *Proceedings of International Workshop on Spoken Language Technologies for Under-resourced Languages*, 2014.

[11] R. Rasipuram and Magimai-Doss M., "Combining acoustic data driven G2P and letter-to-sound rules for under resource lexicon generation," in *Proceedings of Interspeech*, 2012.

[12] G. Chen, D. Povey, and S. Khudanpur, "Acoustic data-driven pronunciation lexicon generation for logographic languages," in *Proceedings of ICASSP*, 2016.

[13] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[14] "IARPA Babel Program," `https://www.iarpa.gov/index.php/research-programs/babel`, 2011.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980v8*, 2014.

[17] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.