INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Deep Bidirectional Long Short-Term Memory Recurrent Neural Networks for Grapheme-to-Phoneme Conversion utilizing Complex Many-to-Many Alignments

Amr El-Desoky Mousa[1], *Björn Schuller*[1,2]

[1]Chair of Complex & Intelligent Systems, University of Passau, Passau, Germany
[2]Department of Computing, Imperial College London, London, UK

amr.mousa@uni-passau.de, schuller@ieee.org

Abstract

Efficient grapheme-to-phoneme (G2P) conversion models are considered indispensable components to achieve the state-of-the-art performance in modern automatic speech recognition (ASR) and text-to-speech (TTS) systems. The role of these models is to provide such systems with a means to generate accurate pronunciations for unseen words. Recent work in this domain is based on recurrent neural networks (RNN) that are capable of translating grapheme sequences into phoneme sequences taking into account the full context of graphemes. To achieve high performance with these models, utilizing explicit alignment information is found essential. The quality of the G2P model heavily depends on the imposed alignment constraints.

In this paper, a novel approach is proposed using complex many-to-many G2P alignments to improve the performance of G2P models based on deep bidirectional long short-term memory (BLSTM) RNNs. Extensive experiments cover models with different numbers of hidden layers, projection layer, input splicing windows, and varying alignment schemes. One observes that complex alignments significantly improve the performance on the publicly available CMUDict US English dataset. We compare our results with previously published results.

Index Terms: grapheme-to-phoneme conversion, long short-term memory, many-to-many alignments

1. Introduction

G2P conversion is the task of generating a sequence of pronunciation symbols (phonemes) for a given sequence of letters (graphemes). It is an essential component in ASR and TTS systems in order to provide highly accurate pronunciations for words outside the base lexicon. The quality of the G2P model significantly affects the performance of the underlying systems.

Successful approaches to G2P conversion use joint sequence models [1, 2]. Therein, an initial grapheme-phoneme sequence alignment is created. Based on this alignment, an inventory of paired units called *graphones* is constructed, where every unit consists of two joint components, a graphemic component and a phonemic component. Then, a joint probability distribution of words and their pronunciations is modeled as a standard n-gram language model (LM) over graphone sequences. Usually, this n-gram model is represented as a weighted finite state transducer (WFST) [3, 4]. An important parameter of this model is the maximum possible number of letters or phonemes per graphone. This parameter represents the alignment constraints. The quality of the G2P model is heavily influenced by the alignment constraints imposed on the model.

In [3], the joint sequence modeling is improved by enhancing the alignment algorithm and using a RNN LM for N-best rescoring along with the standard n-gram LM. In [5], conditional and joint maximum entropy (MaxEnt) models are used for G2P conversion. In [6, 7, 8], conditional random fields (CRF) are used for G2P conversion. In [9], a discriminative structure-prediction model based on averaged perceptron and a margin infused relaxed algorithm (MIRA) is utilized to perform G2P conversion for English, French, Dutch and German. In [10], a joint n-gram model is combined with a CRF model. In [11], many-to-1 alignment constraints are used with CRF models. However, in [12, 7], many-to-many alignments are used.

Neural network models have also been used for G2P conversion. In [13], multilayer perceptron (MLP) neural networks are found to outperform RNNs. Most recently, in [14], unidirectional LSTMs are used with different forms of output delays that enable the model to see several graphemes before outputting any phoneme. Another approach experimented in [14], is the use of BLSTMs with a connectionist temporal classification (CTC) layer [15]. The objective in [14] is to avoid the need of explicit alignment before training. The best results are achieved via BLSTMs with a CTC layer of 512 units. A further improvement gain is obtained by combining this model with a traditional 5-gram WFST model. Another recent work in [16] uses an encoder-decoder neural network architecture with a side-conditioned LM to perform sequence-to-sequence G2P conversion without explicit alignment. This is inspired by the successful translation model of [17]. Also in [16], a BLSTM model that utilizes explicit alignment is found to significantly outperform the encoder-decoder approach without alignment.

In all the recent work with RNN G2P models, not much attention is paid to the alignment problem. Either the alignment is greatly simplified to the 1-to-1 or 1-to-2 cases, like in [16], or it is completely avoided, like in [14], which always comes at the price of significant performance degradation. In this paper, we explore the utilization of many-to-many alignment constraints between graphemes and phonemes in order to improve the performance of G2P models based on deep BLSTM RNNs. We experiment with networks with different numbers of hidden layers, optional linear projection layer, optional splicing window at input side, and various alignment schemes. We test our models on the publicly available CMUDict[1] US English dataset. For the purpose of comparison, we use the same dataset partitioning defined in [1, 2, 4, 5, 14].

[1]http://www.speech.cs.cmu.edu/cgi-bin/cmudict

http://dx.doi.org/10.21437/Interspeech.2016-1229

2. Alignment constraints

The pronunciation generation of the US English words is considered a difficult task compared to other languages like German or Spanish. The pronunciations of US English words appear inconsistent and involve complex alignment relationships between graphemes and phonemes. These alignments are not usually restricted to only 1-to-1, 1-to-many or many-to-1. Rather, the alignment relationships are better described by a mixture of all these cases. This is known as many-to-many alignment which is the most general and natural way of alignment. For example, the word "EXCUSING" with pronunciation "IH K S K Y UW Z IH NG", can be aligned as:

graphemes	E	X	C	U	S	I	N:G
phonemes	IH	K:S	K	Y:UW	Z	IH	NG

Here, we can see the the existence of 1-to-1, 1-to-many and many-to-1 alignments. In addition, it is quite common in the alignment process to find some silent letters that are not naturally aligned to any phonemes, like in the word "LATE" with pronunciation "L EY T" with the alignment:

graphemes	L	A	T	E
phonemes	L	EY	T	_

Therefore, the "null" phoneme (indicated by "_" in the previous example) should be considered as a valid phonetic symbol. In other words, the word "many" in the phonetic side of the alignment should include the case of zero (means no) symbols. A comprehensive discussion of the alignment constraints and methods can be found in [18]. In this work, we use the alignment algorithm provided by the aligner tool from the *Phonetisaurus*toolkit[2]. Using this alignment algorithm, we can impose different types of constraints on the alignment process. The following constraints are explored in our experiments:

- c1) **1-to-{0,1,2}** : one grapheme is aligned to null, one, or two phonemes.
- c2) **{1,2}-to-{0,1,2}**: one or two graphemes are aligned to null, one, or two phonemes.
- c3) **{1,2,3}-to-{0,1,2}**: one, two, or three graphemes are aligned to null, 1, or 2 phonemes.

Here, it is worth noting that we do not allow "null" graphemic symbols. Thus, we do not consider cases where no grapheme is aligned to one or more phonemes. The reason is that the existence of such a symbol will add a great complexity to the process of finding all possible segmentations of a given graphemic string given a finite set of all possible single and compound graphemes. We will see later that this process is required by our decoding approach.

3. Recurrent neural networks

3.1. Standard RNN

A RNN can map from a sequence of input observations to a sequence of output labels. The mapping is defined by activation weights and a non-linear activation function as in a standard MLP. However, recurrent connections allow to access activations from past time. For an input sequence x_1^T, a RNN computes the hidden sequence h_1^T and the output sequence y_1^T by performing the following operations for $t = 1$ to T [19]:

$$h_t = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

$$y_t = W_{hy}h_t + b_y \quad (2)$$

[2]https://github.com/AdolfVonKleist/Phonetisaurus

where $\mathcal{H}$ is the hidden layer activation function, W_{xh} is the weight matrix between input and hidden layer, W_{hh} is the recurrent weight matrix between hidden layer and itself, W_{hy} is the weight matrix between the hidden and output layer, b_h and b_y are the hidden and output layer bias vectors, respectively. In a standard RNN, $\mathcal{H}$ is usually an element-wise application of sigmoid function. Such a network is usually trained using the back-propagation through time (BPTT) training [20].

3.2. LSTM

In [21], an alternative RNN called Long Short-Term Memory (LSTM) is introduced where the conventional neuron is replaced with a so-called *memory cell* controlled by input, output and forget gates in order to overcome the vanishing gradient problem of traditional RNNs [22, 23]. In this case, $\mathcal{H}$ can be described by the following composite function [19]:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (3)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (4)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (5)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (6)$$

$$h_t = o_t \tanh(c_t), \quad (7)$$

where σ is the sigmoid function. i,f,o and c are, respectively, the input, forget, output gates and cell activation vectors.

3.3. Bidirectional LSTM

A BLSTM processes the input sequence in both directions with two sub-layers in order to account for the full input context. These two sub-layers compute forward and backward hidden sequences $\overrightarrow{h}$, $\overleftarrow{h}$ respectively, which are then combined to compute the output sequence y as follows [19] (see Fig.1):

$$\overrightarrow{h}_t = \mathcal{H}(W_{x\overrightarrow{h}}x_t + W_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \quad (8)$$

$$\overleftarrow{h}_t = \mathcal{H}(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}) \quad (9)$$

$$y_t = W_{\overrightarrow{h}y}\overrightarrow{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y \quad (10)$$

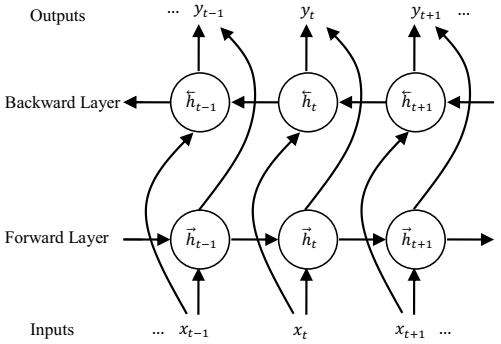

Figure 1: Architecture of BLSTM.

4. G2P conversion model

The training data for the G2P conversion model comes in a form of a dictionary with pairs of words and their pronunciations. It is commonly possible that one word has multiple pronunciation variants in the dictionary. These variants are treated as separate training examples for the G2P model. We start our model creation by aligning all the word-pronunciation pairs of the training dictionary using the Phonetisaurus aligner tool following each of the three different alignment constraints in Sec.2. We

create a different G2P conversion model for each imposed constraint separately. We use the aligned grapheme-phoneme sequences to train a BLSTM network described in Sec.3.3. For a given grapheme sequence $\mathbf{g} = g_1, g_2, ...g_T$ and a corresponding aligned phoneme sequence $\mathbf{p} = p_1, p_2, ..., p_T$, the BLSTM network provides the posterior probability $p(\mathbf{p}|\mathbf{g})$ as follows:

$$p(\mathbf{p}|\mathbf{g}) = p(p_1^T|g_1^T) \approx \prod_{t=1}^{T} p(p_t|g_1^T). \quad (11)$$

This probability assumes that every output phonemic token[3] p_t at each time step t is conditionally dependent on the complete grapheme sequence g_1^T. Thus, the full context is exploited to predict the individual component tokens of the phoneme sequence. Here, we should note that the product of Eq. 11 runs over the highest probable phonemic tokens at all the time steps from 1 to T as provided by the neural network.

In case we use the simplest alignment constraint c1 defined in Sec.2. Then, for a given word of our test set, we have only a single grapheme sequence $\mathbf{g}$ defined by segmenting the word into single letters, and thus we can find the optimum corresponding phoneme sequence $\mathbf{p}^*$ by maximizing over the posterior probability of Eq.11 as follows:

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmax}}\, p(\mathbf{p}|\mathbf{g}). \quad (12)$$

However, if we use the other more complex alignment constraints c2 or c3, then it becomes more difficult to find the optimum corresponding phoneme sequence to a given word. The problem arises from the fact that there is no single segmentation of this word that can be used to deterministically define the graphemic tokens at all time step. There are indeed many valid grapheme sequences for a given word. Every grapheme sequence corresponds to a different valid segmentation.

To solve this problem, we propose to get a list of all valid segmentations that are defined over the grapheme vocabulary. This vocabulary contains all the valid single and compound graphemes collected from the aligned training data. The segmentation algorithm uses a recursive tree building procedure. Then, to find the optimum phoneme sequence for a given test word, we actually need to find the most probable grapheme sequence and its corresponding phoneme sequence. Therefore, we choose to maximize over the joint probability as follows:

$$\mathbf{p}^*, \mathbf{g}^* = \underset{\mathbf{p},\mathbf{g}}{\operatorname{argmax}}\, p(\mathbf{p}, \mathbf{g}) = \underset{\mathbf{p},\mathbf{g}}{\operatorname{argmax}}\, p(\mathbf{p}|\mathbf{g})p(\mathbf{g}). \quad (13)$$

The posterior probability of $p(\mathbf{p}|\mathbf{g})$ is computed from the BLSTM neural network as in Eq.11. However, the prior probability of the grapheme sequence $p(\mathbf{g})$ is modeled by a grapheme LM trained on the segmented training words as defined by the underlying alignment. This LM is given as:

$$p(\mathbf{g}) = p(g_1^T) = \prod_{t=1}^{T} p(g_t|g_1^{t-1}). \quad (14)$$

We use a long-span LSTM LM [24, 25] linearly interpolated with a traditional n-gram backoff LM smoothed with modified Kneser-Ney (MKN) smoothing [26], and another n-gram MaxEnt LM [27, 28]. Another useful score that can be used to refine the optimization process of Eq.13 is the probability of the phoneme sequence $p(\mathbf{p}) = p(p_1^T)$ as estimated by another phoneme LM trained analogously to the grapheme LM.

[3] A phonemic token could be a phoneme or multi-phoneme, and a graphemic token could be a grapheme or multi-grapheme.

5. Dataset

We perform our experiments on the CMUDict US English dataset (release 0.6, 1998). We use the dataset partitioning that comes with the Phonetisaurus distribution [4]. Therein, both the training and test partitions are provided. The training set contains 106,837 words (113,438 pronunciations), whereas the test set contains 12,000 words (12,753 pronunciations). Form the training set, we randomly select 2,670 words (2,835 pronunciations) as a development set. This is the same evaluation setup as in [1, 2, 4, 5, 14]. Thus, the results are directly comparable.

We report the word error rate (WER) and the phoneme error rate (PER) for our best hypothesized pronunciation. In case we have multiple reference pronunciations, a word error is counted only if the hypothesized pronunciation does not match any reference pronunciation. For PER computation, the pronunciation variant with the smallest edit distance is used. This strategy is similar to the one adopted in most previous work [16].

In addition to WER/PER for our best hypothesized pronunciation, we report the best WER/PER for all pronunciations that correspond to segmentation variants of the same test words. This gives an idea about the potential improvement.

Table 1 shows the number of unique grapheme and phoneme tokens produced after applying the three alignment constraints c1, c2 or c3 listed in Sec.2. In addition, we report the average number of tokens per aligned sequence as well as the average number of valid segmentations per test word. Here, we should note that we use special start and end tokens for grapheme and phoneme sequences, namely `<is>` and `</is>` for graphemes, and `<os>` and `</os>` for phonemes.

Table 1: Statistics on the aligned CMUDict.

	alignment		
	c1	**c2**	**c3**
number of unique grapheme tokens	29	170	679
number of unique phoneme tokens	176	188	193
average tokens per sequence	9	8	8
average segmentations per word	1	7	20

6. Experiments

In our initial set of experiments, we use BLSTM neural networks to perform G2P mappings without using any score from grapheme or phoneme LMs. We use different numbers of hidden layers from 1 up to 3 layers. Each hidden layer consists of 500 or 1000 memory cells. The hidden layers are followed by an output softmax layer with a cross-entropy error function and momentum of 0.9. To speed up the training, we use sequence level parallelization with 50 sequences without truncation. The learning rate is set initially to 10^{-3} and then decreased gradually to 10^{-6}. The training process is controlled by monitoring the cross-entropy error on the development set. Table 2 shows the WER results for the three alignment constraints.

Table 2: WERs [%] on CMUDict test set using BLSTMs with different alignment constraints.

	alignment		
#hidden layers × layer size	**c1**	**c2**	**c3**
1 × 500	35.84	33.02	33.18
1 × 1000	34.80	33.08	32.73
2 × 500	29.40	28.22	29.18
2 × 1000	29.33	28.90	28.88
3 × 500	**28.16**	**27.64**	28.00
3 × 1000	28.19	29.18	**27.93**

The results of Table 2 show significant performance improvements in WERs when using the alignment constraints c2 and c3 compared to c1 (at the 95% statistical significance level following the test in [29]). In addition, we see consistent WER improvements when adding more hidden layers to the neural networks. The best performances with alignment constraints c1 and c2 are achieved with 500 cells per hidden layer, whereas, for alignment constraint c3, the best performance is achieved with 1000 cells per hidden layer. This is likely due to the remarkably larger input layer size of 679 units in case of c3 compared to 29 units for c1 and 170 units for c2 (see Table 1).

Based on the results of Table 2, we decided to go further with alignment constraints c2 and c3 using a larger number of hidden layers and fixing the number of hidden cells to 500 for alignment constraint c2, and 1000 for alignment constraint c3. In addition, to select the best pronunciation sequence, we add probability scores from a grapheme LM and a phoneme LM. As previously discussed in Sec.4, both LMs are obtained by linearly interpolating probabilities from 3 types of models, namely a long-span unidirectional LSTM LM, a 7-gram backoff MKN smoothed LM and a 7-gram MaxEnt LM. The latter two models are estimated using the SRILM toolkit [30]. The interpolation weights are optimized on the segmented development data. Each of the two LSTM LMs uses a single hidden layer with 300 memory cells. All the uni- and bidirectional LSTM neural networks in our work are trained and optimized using our own actively developed CURRENNT[4] toolkit [31]. Table 3 shows the WER and PER results for this set of experiments.

Table 3: Results on CMUDict test set using BLSTMs with 3 to 6 hidden layers, grapheme/phoneme LMs and many-to-many alignments (c2, c3). The dimension of each hidden layer is 500 for c2 alignment, and 1000 for c3 alignment.

# hidden layers	alignment			
	c2		c3	
	WER [%]	PER [%]	WER [%]	PER [%]
3	26.45	5.97	26.80	6.09
4	**26.08**	**5.88**	**26.74**	**6.05**
5	26.46	5.97	27.34	6.23
6	26.10	5.89	29.07	6.55

We can see in Table 3 that the best results are achieved with the alignment constraint c2. The use of grapheme and phoneme LMs helps to improve the performance with a significant margin. This can be observed by comparing the WERs at the first row of Table 3 with the WERs at the last two rows of Table 2. Moreover, the optimum number of hidden layers for this setup is found to be 4 hidden layers for both c2 and c3 alignments.

For our best performing experiment in Table 3, we go a step further and experiment with an additional linear projection layer of dimension 100 units following the input layer that uses the 1-hot encoding. In addition, we examine the influence of using an input splicing window of size 3 graphemic tokens (1 token before and 1 token after) on the performance of the G2P BLSTM network as well as the use of a decoding beam of size 10. This delivers our best results as shown in Table 4 in comparison to the previously reported results. In addition to the results for our best hypothesized pronunciation, we report the oracle WER and PER for all pronunciations that correspond to the valid segmentations of test words.

[4]http://sourceforge.net/p/currennt; note that some improved features for easier estimation of LMs are not yet available in this public version.

Table 4: Results on CMUDict test set using a 4 hidden layer BLSTM of dimension 500, grapheme/phoneme LMs, many-to-many alignments (c2), a linear projection layer of dimension 100, input splicing window of size 3, and a decoding beam of size 10. Previous results are listed at the end of the table.

experiment	1^{st} best		oracle	
	WER [%]	PER [%]	WER [%]	PER [%]
BLSTM: 4 × 500	26.08	5.88	23.46	4.66
+ projection layer	25.47	5.73	22.91	4.52
+ splicing window	25.10	5.66	22.38	4.45
+ decoding beam	**23.23**	**5.37**	**3.37**	**0.62**
Novak et al. [4]	33.55	8.24		
Galescu & Allen [2]	28.50	7.00		
Rao et al. [14]	25.80	-		
Chen [5]	24.70	5.90		
Bisani & Ney [1]	24.53	5.88		

In Table 4, we can see improvement gains due to the use of a linear projection layer. Also, a further improvement gain can be obtained by using an input splicing window. Our best results **[WER: 23.23%, PER: 5.37%]** are achieved by using a decoding beam of size 10 at the BLSTM output layer. No further gains are observed with larger beams. Our results are considerably better than the best previously published results in terms of both WER and PER. Moreover, our oracle WERs and PERs over segmentation variants show a large room of potential improvement. Here, it is important to note that we do not consider comparisons with experiments that involve model combinations, like in [14] (BLSTM-CTC + 5-gram WFST, WER = 21.3%), include the development set in training, like in [10] (WER = 23.4%, PER = 5.5%), or use a different version of CMUDict, like in [16][5] (WER = 23.55%, PER = 5.45%).

7. Conclusions

In this paper, we have introduced a novel approach to G2P conversion that uses complex many-to-many alignments with deep BLSTM RNNs. We have examined models with different numbers of hidden layers, linear projection layer, input splicing windows, and varying alignment constraints. The models are enhanced by using grapheme level LMs based on unidirectional LSTMs, backoff models and MaxEnt models. It has been found that the many-to-many alignments allow for improved results on the publicly available CMUDict US English dataset compared to the 1-to-many alignments. Further improvements have been achieved by decoding the output of the BLSTM by a phoneme level LM using a beam search mechanism. Results of the proposed models are compared to the previously published results with similar setup. We have achieved a new state-of-the-art in terms of both WER and PER. As a future work, we plan to compare our approach with an alignment free approach using encoder-decoder neural networks with attention mechanism.

8. Acknowledgements

The research leading to these results has received funding from the European Unions Horizon 2020 Programme through the Innovation Actions #644632 (MixedEmotions) and #645378 (ARIA-VALUSPA), and the German Federal Ministry of Education, Science, Research and Technology (BMBF) under grant agreement #16SV7213 (EmotAsS). We thank NVIDIA Corporation for supporting this research by Tesla K40 GPU donation.

[5]In the work of [16], CMUDict release 0.7 is used.

9. References

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

[2] L. Galescu and J. F. Allen, "Pronunciation of proper names with a joint n-gram model for bi-directional grapheme-to-phoneme conversion," in *Proc. Interspeech Conference of the International Speech Communication Association*, Denver, Colorado, USA, September 2002, pp. 109 – 112.

[3] 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 *Proc. Interspeech Conference of the International Speech Communication Association*, Portland, Oregon, USA, September 2012, pp. 2526–2529.

[4] J. R. Novak, N. Minematsu, and K. Hirose, "Failure transitions for joint n-gram models and g2p conversion," in *Proc. Interspeech Conference of the International Speech Communication Association*, Lyon, France, August 2013, pp. 1821–1825.

[5] S. F. Chen, "Conditional and joint models for grapheme-to-phoneme conversion," in *Proc. European Conference on Speech Communication and Technology*, Geneva, Switzerland, September 2003.

[6] H. Chen and A. F. Murray, "Continuous restricted Boltzmann machine with an implementable training algorithm," *Vision, Image and Signal Processing, IEE Proc.*, vol. 150, no. 3, pp. 153 – 158, 2003.

[7] P. Lehnen, S. Hahn, A. Guta, and H. Ney, "Hidden conditional random fields with m-to-n alignments for grapheme-to-phoneme conversion," in *Proc. Interspeech Conference of the International Speech Communication Association*, Portland, OR, USA, September 2012, pp. 2554–2557.

[8] P. Lehnen, A. Allauzen, T. Lavergne, F. Yvon, S. Hahn, and H. Ney, "Structure learning in hidden conditional random fields for grapheme-to-phoneme conversion," in *Interspeech*, Lyon, France, August 2013, pp. 2326–2330.

[9] S. Jiampojamarn, C. Cherry, and G. Kondrak, "Joint processing and discriminative training for letter-to-phoneme conversion," in *Proc. Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, USA, June 2008, pp. 905–913.

[10] K. Wu, C. Allauzen, K. B. Hall, M. Riley, and B. Roark, "Encoding linear models as weighted finite-state transducers," in *Proc. Interspeech Conference of the International Speech Communication Association*, Singapore, Singapore, September 2014.

[11] P. Lehnen, S. H. Andreas, Guta, and H. Ney, "Incorporating alignments into conditional random fields for grapheme to phoneme conversion," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Prague, Czech Republic, May 2011, pp. 4916–4919.

[12] S. Jiampojamarn, G. Kondrak, and T. Sherif, "Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion," in *Proc. Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, New York, USA, April 2007, pp. 372–379.

[13] E. B. Bilcu, "Text-to-phoneme mapping using neural networks," Ph.D. dissertation, Tampere University of Technology, Tampere, Finland, October 2008.

[14] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Brisbane, Australia, April 2015, pp. 4225–4229.

[15] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. International Conference on Machine Learning*, Pittsburgh, Pennsylvania, USA, June 2006, pp. 369–376.

[16] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," in *Proc. Interspeech Conference of the International Speech Communication Association*, Dresden, Germany, May 2015.

[17] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Neural Information Processing Systems*, Montreal, Canada, December 2014, pp. 3104–3112.

[18] S. Jiampojamarn and G. Kondrak, "Letter-phoneme alignment: An exploration," in *Proc. Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden, July 2010, pp. 780–788.

[19] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vancouver, BC, Canada, May 2013, pp. 6645 – 6649.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, no. 323, pp. 533 – 536, 1986.

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

[22] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451 – 2471, 1999.

[23] F. A. Gers, "Long short-term memory in recurrent neural networks," Ph.D. dissertation, Department of Computer Science, Swiss Federal Institute of Technology, Lausanne, EPFL, Switzerland, 2001.

[24] T. Mikolov, M. Karafiát, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Interspeech*, Makuhari, Chiba, Japan, Sep. 2010, pp. 1045 – 1048.

[25] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Interspeech*, Portland, OR, USA, Sep. 2012.

[26] R. Kneser and H. Ney, "Improved backing-off for M-gram language modeling," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, Detroit, Michigan, USA, May 1995, pp. 181 – 184.

[27] J. Wu, "Maximum entropy language modeling with non-local dependencies," Ph.D. dissertation, John Hopkins University, Baltimore, Maryland, USA, 2002.

[28] T. Alumäe and M. Kurimo, "Efficient estimation of maximum entropy language models with N-gram features: an SRILM extension," in *Proc. Interspeech Conference of the International Speech Communication Association*, Makuhari, Chiba, Japan, September 2010, pp. 1820–1823.

[29] M. Bisani and H. Ney, "Bootstrap estimates for confidence intervals in ASR performance evaluation," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, Montreal, Canada, May 2004, pp. 409 – 412.

[30] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *Proc. International Conference on Spoken Language Processing*, vol. 2, Denver, Colorado, USA, Sep. 2002, pp. 901 – 904.

[31] F. Weninger, J. Bergmann, and B. Schuller, "Introducing CURRENNT – the Munich open-source CUDA RecurREnt Neural Network Toolkit," *Journal of Machine Learning Research*, vol. 15, no. 99, Oct. 2014.