GRAPHEME-TO-PHONEME CONVERSION USING LONG SHORT-TERM MEMORY RECURRENT NEURAL NETWORKS

Kanishka Rao, Fuchun Peng, Haşim Sak, Françoise Beaufays

Google Inc. U.S.A.

{kanishkarao,fuchunpeng,hasim,fsb}@google.com

ABSTRACT

Grapheme-to-phoneme (G2P) models are key components in speech recognition and text-to-speech systems as they describe how words are pronounced. We propose a G2P model based on a *Long Short-Term Memory* (LSTM) recurrent neural network (RNN). In contrast to traditional joint-sequence based G2P approaches, LSTMs have the flexibility of taking into consideration the full context of graphemes and transform the problem from a series of grapheme-to-phoneme conversions to a word-to-pronunciation conversion. Training joint-sequence based G2P require explicit grapheme-to-phoneme alignments which are not straightforward since graphemes and phonemes don't correspond one-to-one. The LSTM based approach forgoes the need for such explicit alignments.

We experiment with unidirectional LSTM (ULSTM) with different kinds of output delays and deep bidirectional LSTM (DBLSTM) with a connectionist temporal classification (CTC) layer. The DBLSTM-CTC model achieves a word error rate (WER) of 25.8% on the public CMU dataset for US English. Combining the DBLSTM-CTC model with a joint n-gram model results in a WER of 21.3%, which is a 9% relative improvement compared to the previous best WER of 23.4% from a hybrid system.

Index Terms— speech recognition, pronunciation, RNN, LSTM, G2P, CTC

1. INTRODUCTION

Knowing how words are pronounced is an essential ingredient in any automatic speech recognition (ASR) or in text-to-speech (TTS) systems. In ASR, pronunciations are the middle-layer between the acoustic model and the language model and the performance of the overall system relies on the coverage and quality of the pronunciation component. A pronunciation system typically comprises a static word-pronunciation dictionary which is typically written by experts or may even be generated using a data-driven approach [1]. However, such a static list can never cover all the possible words in a language and is usually complemented with a G2P engine to generate pronunciations. A G2P converts a word, as a series of characters or graphemes, to a prounciation, as a series of phones. For example, given a word 'google' a G2P would predict;

$$google \rightarrow g\ u\ g\ @\ l$$

A robust G2P is an essential piece in the ASR system as it is invoked anytime a word is not in the static dictionary which may be very frequent depending on the size of the dictionary.

For some languages with consistent pronunciations, like Spanish, this task is relatively easy, but for languages like US English pronunciations are much harder to predict. In typical G2P approaches, such as joint-sequence models [2], the problem is sub-divided into three parts:

- Aligning: aligning $G \rightarrow P$
- Training: learning the $G \rightarrow P$ conversions
- Decoding: finding the most likely pronunciation given the model

Joint-sequence models create an initial $G \rightarrow P$ alignment and then model the sequence of such joint tokens. However, such alignments may not always be straightforward. For example, the word $able$ may be pronounced as "$eI\ b\ @\ l$": here the graphemes a, b and l align with the phonemes eI, b and l respectivelym however, the grapheme e is omitted from the phoneme sequence and instead the phoneme @ is inserted. A joint-sequence model approach may overcome this with the use of an empty symbol, ϵ and align:

$$a : eI \quad b : b \quad \epsilon : @ \quad l : l \quad e : \epsilon$$

However, establishing such an alignment is not endemic to the G2P task which only requires the final sequence of phonemes for a given word without the need of specific grapheme-to-phoneme alignments.

In this paper we present a novel approach to the problem using *Long Short-Term Memory* (LSTM) neural networks [3] which are a class of recurrent neural network especially suited for sequence modeling. LSTMs avoids the need for explicit alignment before training; instead, with a dynamic contextual window, the LSTM may see several graphemes before outputting any phoneme, which allows it to make contextually-aware decisions. For example, in the word $able \rightarrow eI\ b\ @\ l$ the grapheme e is not rendered in the phoneme sequence, while in $get \rightarrow gEt$ the grapheme e corresponds to phoneme E. The contextual window over the previous graphemes, eg abl versus g, help the LSTM make the correct prediction for the grapheme e. In some cases the left context may not be enough, e.g. in $car \rightarrow k\ A\ r$ vs. $care \rightarrow k\ E\ r$: here the model needs to see the *future* (right) context to output the correct phoneme. To exploit the future context, we experiment with output delays where the output is delayed by a certain amount and with bidirectional LSTM which can see the entire input word before outputting phonemes.

We show that LSTM models outperform previous state of the art techniques. A hybrid approach that combines LSTMs with joint ngram models further improves accuracy.

2. RELATED WORK

G2P conversion can be considered as a machine translation problem where we need to translate source graphemes into target phonemes. In such a formulation, an alignment model needs to be first constructed and then a translation model – such as a joint ngram model – is built from the alignments [2, 4]. Such ngram based translation models are usally implemented as a weighted finite state transducer (WFST) [5, 6]. G2P can also be seen as a classification problem and implemented with a maximum entropy classifier [7], or as a sequence labeling problem where statistical sequence labeling techniques such as conditional random fields (CRF) [8, 9] and perceptron HMM [10] can be used. Neural network approaches have also been proposed for G2P problems. For example, Bilcu [11] investigated different types of neural network structures and found that multilayer perceptrons performed best. Hybrid models were found effective. For example, Wu et al. [12] combines a joint ngram model with a CRF model, and Hahn et al. [13] combines a basic joint ngram model with a decision tree model. In this paper, we explore various LSTM archictures, and we show that combining LSTM with a basic joint ngram model achieves the best G2P performance.

3. LSTM

Recurrent neural networks (RNN), unlike feedforward neural networks (FFNN), can utilize the context of previous inputs while processing the current input using cyclic connections. This makes RNNs well suited for sequence modeling tasks where context within the sequence is useful, such as with phoneme recognition and handwriting recognition tasks. RNNs store the activations from previous steps in their internal state and can build a dynamic temporal context window instead of a fixed context window which may be used with FFNNs.

However, conventional RNNs suffer from the vanishing gradient and exploding gradient problems [14, 15] which limit their ability to model long range dependencies. LSTM [3] RNNs have been proposed to overcome these limitations. LSTMs contain special units called *memory units* in the recurrent hidden layer that have self connections which allow them to store their temporal state. Special multiplicative gates in these units control the temporal flow of inputs and outputs. They may also *forget* or reset their states. These gates dynamically maintain the temporal context window in the LSTM.

Having such a contextual memory makes LSTMs ideal for sequential tasks where the current task output may depend on previous task inputs. LSTMs have successfully been applied to, e.g. phonetic labeling of acoustic frames [14], handwriting recognition [16], and language modeling [17]. They have been shown to outperform standard RNNs and deep neural networks (DNNs) in acoustic frame labeling tasks. LSTMs are very well-suited for the G2P task which can be modeled as a sequence transcription task requiring temporal context. In this paper, we develop an LSTM based G2P which to our knowledge is the first such application of LSTMs.

4. LSTM-BASED G2P IMPLEMENTATION

We configure LSTMs with an input layer of size equal to the number of graphemes and an output layer of size equal to the number of phonemes. In US English, this means 27 graphemes for the lowercase alphabet symbols plus the apostrophe, and 40 phonemes following the XSampa phoneset [1]. The inputs (outputs) are constructed as 27 (40) dimension ”one-hot” vector representations with a value of one for the index representing the grapheme (phoneme), and values of zero for all other indices. The input layer is connected to a hidden LSTM layer which is connected to the output layer. In some experiments we consider deep LSTM models where multiple hidden LSTM layers are connected in a series.

4.1. Unidirectional models

A unidirectional LSTM is setup with 1024 memory units with an output layer with softmax activations and a cross-entropy loss function. LSTMs with fewer units (512, 128 and 64) were also evaluated but did not perform as well. The LSTM is intialized with random weights, trained with a learning rate 0.002, and terminated according to performance on a development data set. Since ULSTMs only exploit left/past context, we introduce a concept of output delays, and experiment with various configurations.

[1] http://en.wikipedia.org/wiki/X-SAMPA

4.1.1. *Zero-delay*

In the simplest approach, without any output delay, the input sequence is the series of graphemes and the output sequence as the series of phonemes. In the (common) case of unequal number of graphemes and phonemes we pad the sequence with an empty marker, ϕ. For example, we have:
Input: $\{$g, o, o, g, l, e$\}$
Output: $\{$g, u, g, @, l, $\phi\}$

4.1.2. *Fixed-delay*

In this mode, we pad the output phoneme sequence with a fixed delay, this allows the LSTM to see several graphemes before outputting any phoneme, and builds a contextual window to help predict the correct phoneme. As before, in the case of unequal input and output size, we pad the sequence with ϕ. For example, with a fixed delay of 2, we have:
Input: $\{$g, o, o, g, l, e, $\phi\}$
Output: $\{\phi, \phi$ g, u, g, @, l$\}$

4.1.3. *Full-delay*

In this approach, we allow the model to see the entire input sequence before outputting any phoneme. The input sequence is the series of graphemes followd by an end marker, Δ, and the output sequence contains a delay equal to size of the input followed by the series of phonemes. Again we pad unequal input and output sequences with ϕ. For example;
Input: $\{$g, o, o, g, l, e, Δ, $\phi, \phi, \phi, \phi\}$
Output: $\{\phi, \phi, \phi, \phi, \phi, \phi, \phi$, g, u, g, @, l$\}$

With the full delay setup we use an additional end marker to indicate that all the input graphemes have been seen and that the LSTM can start outputting phonemes. We discuss the impact of these various configurations of output delay on the G2P performance in Section 6.1.

4.2. Bidirectional models

While unidirectional models require artificial delays to build a contextual window, bidirectional LSTMs (BLSTM) achieve this naturally as they see the entire input before outputting any phoneme. The BLSTM setup is nearly identical to the unidirectional model, but has ”backward” LSTM layers (as described in [14]) which process the input in the reverse direction.

4.2.1. *Deep Bidirectional LSTM*

We found that deep-BLSTM (DBLSTM) with mutiple hidden layers perform slightly better than a BLSTM with a single hidden layer. The optimal performance was achieved with a architecture, shown in Figure 1, where a single input layer was fully connected to two parallel layers of 512 units each;

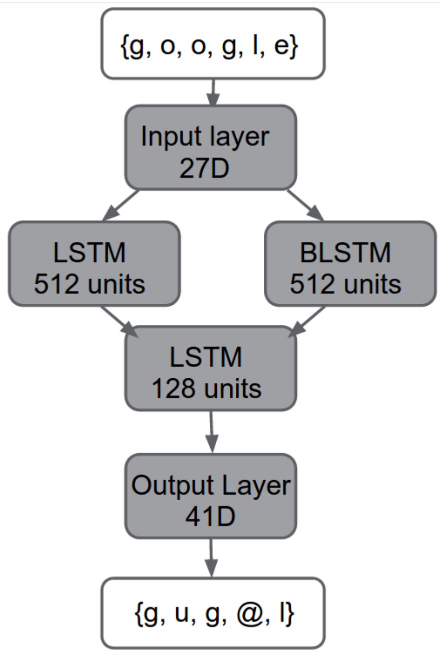

Fig. 1. The best performing G2P neural network architecture using a DBLSTM-CTC.

one unidirectional and one bidirectional. This first hidden layer was fully connected to a single unidirectional layer of 128 units. The second hidden layer was connected to an output layer. The model was initialized with random weights and trained with a learning rate of 0.01.

4.2.2. *Connectionist Temporal Classification*

Along with the DBLSTM we use a connectionist temporal classification [18] (CTC) output layer which interprets the network outputs as a probability distribution over all possible output label sequences, conditioned on the input data. The CTC objective function directly maximizes the probabilities of the correct labelings.

The CTC output layer has a softmax output layer with 41 units, one each for the 40 output phoneme labels and an additional ”blank” unit. The probability of the CTC ”blank” unit is interpretted as observing no label at the given time step. This is similar to the use of ϵ described earlier in the joint-sequence models, however, the key difference here is that this is handled implicitly by the DBSLTM-CTC model instead of having explicit alignments with join-sequence models.

4.3. Combination G2P Implementation

LSTMs and joint n-gram models are two very different approaches to G2P modeling since LSTMs model the G2P task at the full sequence (word) level instead of the n-gram (grapheme) level. These two models may generalize in different ways and a combination of both approaches may result in a better overall model. We combine both models by representing the output of the LSTM G2P as a finite state transducer (FST) and then intersect it with the output of the n-gram model which is also represented as a FST. We select the single best path in the resulting FST which corresponds to a single best pronunciation. (We did not find any significant gains by using a scaling factor between the two models.)

5. EXPERIMENTS

In this paper, we report G2P performance on the publicly available CMU pronunciation dictionary. We evaluate performance using phoneme error rate (PER) and word error rate (WER) metrics. PER is defined as the number of insertions, deletions and substitutions divided by the number of true phonemes, while WER is the number of words errors divided by the total number of words. The CMU dataset contains 106,837 words and of these we construct a development set using 2,670 words to determine stopping criteria while training, and a test set using 12,000 words. We use the same training and testing split as found in [12, 7, 4] and thus the results are directly comparable.

6. RESULTS AND DISCUSSION

6.1. Impact of Output Delay

Table 1 compares the performance of unidirectional models with varying output delays. As expected, we find that when using fixed delays increasing the size of the delays helps, and that full delay outperforms any fixed delay. This confirms the importance of exploiting future context for the G2P task.

Output Delay	Phoneme Error Rate (%)
0	32.0
3	10.2
4	9.8
5	9.5
7	9.5
Full-delay	9.1

Table 1. Accuracy of ULSTM G2P with output delays.

6.2. Impact of CTC and Bi-directional Modeling

Table 2 compares LSTM models to various approaches proposed in the literature. The numbers reported for the LSTM are *raw* outputs, i.e. we do not decode the output with any language model. In our experiments, we found that while unidirectional models benefitted from decoding with a phoneme language model (which we implemented as another LSTM trained on the same training data), the BLSTM with CTC outputs did not see any improvement with the additional phoneme language model, likely because it already memorizes and enforces contextual dependencies similar to those imposed by an external langauge model.

Model	Word Error Rate (%)
Galescu and Allen [4]	28.5
Chen [7]	24.7
Bisani and Ney [2]	24.5
Novak et al. [6]	24.4
Wu et al. [12]	23.4
5-gram FST	27.2
8-gram FST	26.5
Unidirectional LSTM with Full-delay	30.1
DBLSTM-CTC 128 Units	27.9
DBLSTM-CTC 512 Units	25.8
DBLSTM-CTC 512 + 5-gram FST	21.3

Table 2. Comparison of various G2P technologies.

The table shows that BLSTM architectures outperform unidirectional LSTMs, and also that they compare favorably to WFST based ngram models (25.8% WER vs 26.5%). Furthermore, a combination of the two technologies as described in 4.3 outperforms both models, and other approaches proposed in the literature.

Table 3 compares the sizes of some of the models we trained and also their execution time in terms of average number of milliseconds per word. It shows that BLSTM architectures are quite competitive with ngram models: the 128-unit BLSTM which performs at about the same level of accuracy as the 5-gram model is 10 times smaller and twice as fast, and the 512-unit model remains extremely compact if arguably a little slow (no special attempt was made so far at optimizing our LSTM code for speed, so this is less of a concern). This makes LSTM G2Ps quite appealing for on-device implementations.

Model	Model Size	Model Speed
5-gram FST	30 MB	35 ms/word
8-gram FST	130 MB	30 ms/word
DBLSTM-CTC 128 Units	3 MB	12 ms/word
DBLSTM-CTC 512 Units	11 MB	64 ms/word

Table 3. Model size and speed for n-gram and LSTM G2P.

7. CONCLUSION

We suggested LSTM-based architectures to perform G2P conversions. We approached the problem as a word-to-pronunciation sequence transcription problem in contrast to the traditional joint grapheme-to-phoneme modeling approach and thus do not require explicit grapheme-to-phoneme alignment for training. We trained unidirectional models with various output delays to capture some amount of future context, and found that models with greater contextual information perform better. We also trained deep BLSTM models that can leverage the context of the entire input sequence along with a CTC output layer which directly maximizes the probabilities of the correct output labelings. The DBLSTM-CTC based G2P outperforms n-gram based approach in terms of accuracy and a combination of the DBLSTM-CTC and the n-gram models results in a word error rate of 21.3% on the public CMU dataset, this is, to our knowledge, the best performance reported so far on the CMU dataset.

8. REFERENCES

[1] A. Rutherford, F. Peng, and F. Beaufays, "Pronunciation learning for named-entities through crowd-sourcing," in *Proceedings of InterSpeech*, 2014.

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

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9(8), pp. 17351780, 1997.

[4] L. Galescu and J. F. Allen, "Pronunciation of proper names with a joint n-gram model for bi-directional grapheme-to-phoneme conversion," in *Proceedings of InterSpeech*, 2002.

[5] J. R. Novak et al., "Improving wfst-based g2p conversion with alignment constraints and rnnlm n-best rescoring," in *Proceedings of InterSpeech*, 2012.

[6] J. R. Novak, N. Minematu, and K. Hirose, "Failure transitions for joint n-gram models and g2p conversion," in *Proceedings of InterSpeech*, 2013.

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

[8] D. Wang and S. King, "Letter-to-sound pronunciation prediction using conditional random fields," *IEEE Signal Processing Letters*, vol. 18 (2), pp. 122 – 125, 2011.

[9] 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 *Proceedings of InterSpeech*, 2013.

[10] S. Jiampojamarn, C. Cherry, and G. Kondrak, "Joint processing and discriminative training for letter-to-phoneme conversion," in *Proceedings of ACL*, 2008, pp. 905 – 913.

[11] E. B. Bilcu, *Text-to-Phoneme Mapping Using Neural Networks*, Ph.D. thesis, Tampere University of Technology, 2008.

[12] K. Wu et al., "Encoding linear models as weighted finite-state transducers," in *Proceedings of InterSpeech*, 2014.

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

[14] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of ICASSP*, 2013, pp. 6645 – 6649.

[15] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," in *Proceedings of InterSpeech*, 2014.

[16] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Proceedings of NIPS*, 2008, pp. 545 – 552.

[17] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Proceedings of InterSpeech*, 2012, pp. 194 – 197.

[18] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of ICML*, 2006.