BIDIRECTIONAL RECURRENT NEURAL NETWORK LANGUAGE MODELS FOR AUTOMATIC SPEECH RECOGNITION

Ebru Arisoy[1], *Abhinav Sethy*[2], *Bhuvana Ramabhadran*[2], *Stanley Chen*[2]

[1] IBM Turkey, Istanbul, Turkey
[2] IBM T. J. Watson Research Center, Yorktown Heights, NY, USA

ebruas@tr.ibm.com, {asethy, bhuvana, stanchen}@us.ibm.com

ABSTRACT

Recurrent neural network language models have enjoyed great success in speech recognition, partially due to their ability to model longer-distance context than word n-gram models. In recurrent neural networks (RNNs), contextual information from past inputs is modeled with the help of recurrent connections at the hidden layer, while Long Short-Term Memory (LSTM) neural networks are RNNs that contain units that can store values for arbitrary amounts of time. While conventional unidirectional networks predict outputs from only *past* inputs, one can build bidirectional networks that also condition on *future* inputs. In this paper, we propose applying bidirectional RNNs and LSTM neural networks to language modeling for speech recognition. We discuss issues that arise when utilizing bidirectional models for speech, and compare unidirectional and bidirectional models on an English Broadcast News transcription task. We find that bidirectional RNNs significantly outperform unidirectional RNNs, but bidirectional LSTMs do not provide any further gain over their unidirectional counterparts.

Index Terms— Language modeling, recurrent neural networks, long short term memory, bidirectional neural networks

1. INTRODUCTION

Recurrent neural networks have been shown to yield superior performance in language modeling over a wide range of domains [1, 2, 3]. In contrast to feedforward neural networks, RNNs are not limited to a fixed number of past inputs. Instead, an RNN can potentially model long-range dependencies of arbitrary length by utilizing recurrent connections in its hidden layer. For RNN language models, this translates to predicting the probability of the next word using many more previous words than the typical 2–4 used in a word n-gram model. However, in practice it has been found that RNNs cannot effectively use information beyond about 5–10 time steps back, due to the well-known "vanishing gradient" problem [4]. The gradient of the error function decays exponentially over time, reducing the influence of inputs far back in time.

Long Short-Term Memory (LSTM) neural networks address this limitation by replacing the nonlinear units in the hidden layer of an RNN with memory blocks that can store values for arbitrary amounts of time [5]. Multiplicative gates are used to read, write, and clear the values of these blocks. LSTM neural networks have been shown to give excellent performance for automatic speech recognition. In acoustic modeling, LSTM models outperform state-of-the-art deep neural networks [6, 7] while LSTM language models yield better performance than conventional n-gram and RNN language models [8, 9].

Conventional RNN and LSTM models are *unidirectional*; *i.e.*, input data is processed in temporal order and output probabilities are estimated conditioning only on past context. While future context is clearly also important, unidirectional models do not explicitly model the effect of future words on the probability of the current word. Instead, this dependence is captured only via the effect of the *current* word on the probabilities of *future* words. Intuitively, there may be benefit from explicitly modeling dependencies in *both* directions; *i.e.*, to build models going both forward and backward in time to reap the benefits of model combination.

Bidirectional RNNs provide an elegant framework for fusing information from past and future contexts together [10]. In bidirectional RNNs, an extra hidden layer, the *backward hidden layer*, is trained by processing the input data in reverse order. The original hidden layer (or *forward hidden layer*) and the backward hidden layer are not directly connected; instead, they both connect to the same output layer. Bidirectional LSTMs are obtained by replacing the hidden units of a bidirectional RNN with memory blocks [11]. Bidirectional RNNs and LSTMs have been successfully applied to tasks such as handwriting recognition and acoustic modeling [12, 6].

In this paper, we propose applying bidirectional RNNs and LSTMs to language modeling for speech recognition. Unlike for unidirectional language models, conditional word probabilities from bidirectional models cannot be correctly combined using the chain rule to compute the probability of a complete utterance. In addition, conditioning on future inputs complicates speech recognition decoding. We discuss how we address these issues, and then compare both unidirectional and bidirectional RNNs and LSTMs on an English Broadcast News transcription task. We achieve an improvement of 0.2% absolute for a bidirectional RNN as compared to its unidirectional counterpart, but achieve no gains with bidirectional LSTMs as compared to unidirectional models.

Several other methods have been proposed for explicitly incorporating future information when computing the probability of the current output. In acoustic modeling, one can condition on future acoustic inputs while still having a valid left-to-right model; this is equivalent to delaying outputs for a few time steps [7]. Another method, applied to language modeling for machine translation, is to simply train two independent models, one on the original input and one on the reversed input [13]. Both models can be used to compute the probability of an utterance; these probabilities are combined during evaluation.

The rest of the paper is organized as follows: Section 2 briefly explains the RNN models used in the paper. Experiments and results are described in Section 3, and Section 4 presents conclusions.

Ebru Arisoy (ebruarisoy.saraclar@mef.edu.tr) is now with the Electrical and Electronics Engineering Department, MEF University, Istanbul, Turkey.

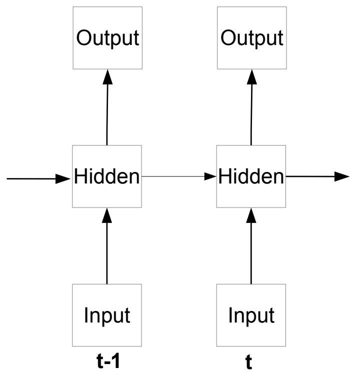

Fig. 1. RNN architecture unfolded in time.

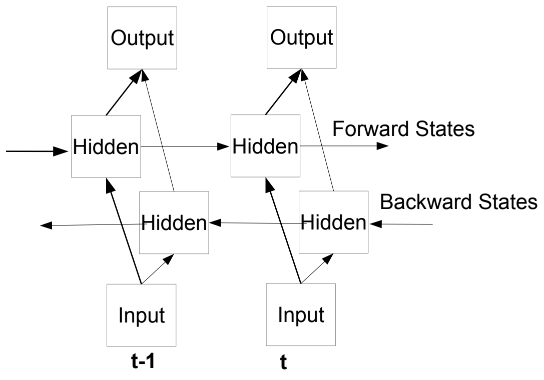

Fig. 2. Bidirectional RNN architecture unfolded in time.

2. NEURAL NETWORKS

2.1. Recurrent Neural Networks

Recurrent neural networks are generally used to model sequential data. The power of an RNN comes from keeping a representation of all previous inputs in its hidden state. Fig. 1 shows a basic RNN architecture unfolded in time for two time steps. As shown in the figure, the hidden state at time t depends on the input at time t as well as the hidden state at time $t-1$, which depends on the input at time $t-1$ as well as the hidden state at time $t-2$, etc.

Formally, given an input vector sequence $X = \{x_1, \cdots, x_T\}$ and an output vector sequence $Y = \{y_1, \cdots, y_T\}$, RNN activations are calculated as follows:

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$
$$y_t = W_{hy}h_t + b_y \quad (2)$$

where h_t denotes the hidden layer vector, W_{xh} denotes the input-to-hidden-layer weight matrix, W_{hh} denotes the hidden-to-hidden-layer weight matrix and W_{hy} denotes the output-to-hidden-layer weight matrix. The values b_h and b_y denote the hidden and output layer biases, respectively.

In RNN language modeling, the conditional word probabilities $P(w_t|w_{t-1}, h_{t-2})$ are calculated as follows:

$$p(w_t = i|w_{t-1}, h_{t-2}) = \frac{\exp(y_t^i)}{\sum_{j=1}^{N}\exp(y_t^j)} \quad (3)$$

where y_t^i represents the ith element of the output vector y_t. Here, each output target corresponds to a word in the vocabulary. The probability of a word sequence $W = w_1, w_2, \cdots, w_T$ is calculated by multiplying conditional word probabilities, given as

$$P(W) = \prod_{t=1}^{T} p(w_t|w_{t-1}, h_{t-2}) \quad (4)$$

The advantage of RNN language models as compared to word or class n-gram models and feedforward neural networks is that they do not restrict the history to the preceding $n-1$ words.

2.2. Long Short-Term Memory Neural Networks

Even though RNNs potentially utilize arbitrarily long histories, in practice the effective context length of an RNN is quite limited. Long Short-Term Memory neural networks were proposed to remedy this limitation. An LSTM neural network replaces the nonlinear units in the hidden layer of an RNN with memory blocks containing memory cells for storing values; and multiplicative gates for reading (*output*), writing (*input*), and resetting (*forget*) these values. A memory cell can be used to store information for long periods, and gates collect activations from both inside and outside a memory block to update a memory cell's value.

The LSTM equations are given as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (5)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (6)$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (7)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (8)$$
$$h_t = o_t \tanh(c_t) \quad (9)$$

where i_t, f_t, o_t and c_t represent the input gate, forget gate, output gate and cell activation vectors at time t, respectively. The matrices W_{**} denote the weight matrices between various layers, gates, and cells; *e.g.*, W_{xi} represents the weight matrix between the input layer and the input gates. The gate and cell bias terms are denoted as b_i, b_f, b_o and b_c; and $\sigma(\cdot)$ is the logistic sigmoid function. For language modeling, after computing h_t, conditional word probabilities $P(w_t|w_{t-1}, h_{t-2})$ are calculated using Eqs. (2) and (3).

2.3. Bidirectional Recurrent Neural Networks

Bidirectional RNNs exploit both the past and future context by processing the input data in both directions. Figure 2 shows a bidirectional RNN architecture unfolded in time for two time steps. As shown in the figure, bidirectional RNNs compute a forward hidden layer h_t^F by iterating through the input sequence from $t = 1, \ldots, T$, and a backward hidden layer h_t^B by iterating through the input sequence from $t = T, \ldots, 1$. These two hidden layers are combined into a single output layer using the following equations:

$$h_t^F = \tanh(W_{xh}^F x_t + W_{hh}^F h_{t-1}^F + b_h^F) \quad (10)$$
$$h_t^B = \tanh(W_{xh}^B x_t + W_{hh}^B h_{t+1}^B + b_h^B) \quad (11)$$
$$y_t = W_{hy}^F h_t^F + W_{hy}^B h_t^B + b_y \quad (12)$$

Bidirectional LSTMs are obtained by replacing the hidden layer activations with memory blocks.

Note that in language modeling, the output from the last time step is the input for the next time step.[1] With bidirectional models, this causes circular dependencies to arise when combining probabilities from multiple time steps. Unlike with unidirectional models, multiplying individual conditional probabilities $P(w_t|w_t - 1, h_t - 2, w_t + 1, h_t + 2)$ from a bidirectional language model does not compute a true likelihood, but rather a *pseudolikelihood*. Still, it is straightforward to optimize the pseudolikelihood of training data rather than its likelihood during model training, and this is in fact what we do.

3. EXPERIMENTS

3.1. Experimental Set-up

We performed experiments on an English Broadcast News task. The baseline system is based on the 2007 IBM GALE speech transcription system [14]. The discriminatively-trained speaker-adaptive acoustic model is trained on 430h of Broadcast News audio. The baseline language model is a conventional word 4-gram model trained on a total of 350M words from multiple sources, and the vocabulary is about 84K words. The held-out set consists of the reference transcriptions of the *rt03* and *dev04* test sets and contains about 48K words. The test set is the *rt04* test set, containing approximately 4 hours of data. The baseline language model yields a word-error rate (WER) of 13.0% on this test set.

3.2. Training RNN and LSTM Language Models

RNN and LSTM language models are trained on a 12M-word subset of the original 350M-word corpus. We implemented training algorithms for the unidirectional and bidirectional RNN and LSTM language models using the Theano library [15]. The RNN/LSTM models are trained on a GPU using back propagation through time with mini-batch updates. In mini-batch training, a bunch of sentences are propagated through the network independently and the network parameters are updated after processing all of the words in the sentences in the mini-batch. To take maximal advantage of the fast matrix operations available on a GPU, each sentence in a mini-batch should be of the same length. However, there can be huge variability in sentence length, especially in the Broadcast News domain. While shorter sentences can be padded with special symbols to equalize their lengths, doing so limits the potential speed-up obtained from using a GPU.

In order to efficiently train RNN/LSTM models on GPUs, we first concatenate all of the sentences in the training data and then split this long word sequence into equal lengths. The sequence length is set to 18 words, the average sentence length in the training data. We use 8 fixed-length sequences in each mini-batch in our experiments. At the beginning of training, the initial hidden state vector at $t = 0$ is initialized with all zeros. After processing the word sequences in one mini-batch, the initial hidden state vector is updated as well as the other network parameters; these are utilized to initialize the hidden state vectors for the next batch. When evaluating a model, each sentence in the test data is processed independently by initializing the hidden state vector to its latest updated value. Note that our approach may not yield optimal performance due to the arbitrary locations of word sequence boundaries. However, our results indicate that this algorithm makes a reasonable trade-off between training speed and performance. Another approach proposed for efficient RNNLM training on GPUs is to splice sentences together to form approximately equal-length chunks, to minimize the amount of padding required to equalize batch sizes [16].

[1]In contrast, in acoustic modeling, the predicted outputs are context-dependent phonetic states while the inputs are features derived from the speech signal.

	Perplexity	WER (%)
Baseline	133.2	13.0
Baseline + uni-RNN	123.2	12.8
Baseline + uni-LSTM	114.1	12.6

Table 1. Perplexities and WERs for unidirectional RNN and LSTM models linearly interpolated with the baseline model.

The computation required to train or evaluate a neural network language model is generally dominated by the multiplications needed to compute the output layer. In order to reduce this cost, we restrict the output vocabulary to the 20K most frequent words in the vocabulary. All words outside of this output vocabulary are mapped to the *out-of-shortlist* class; the probabilities of words in this class are assumed to be equally likely. Partitioning the output layer into classes can also be used to reduce the computational complexity of neural network language models [3, 17]. In order to reduce the overall number of parameters, we use a projection layer at the input. Each word in the vocabulary is mapped to a d-dimensional continuous feature vector via a weight matrix shared across word positions. In Eq. (1), x_t represents the continuous feature representation of w_t, the word at time t. Continuous feature vector representations of words as well as the corresponding weight matrices and biases are learned using back propagation through time.

Unidirectional and bidirectional RNN and LSTM language models are trained using a 180-dimensional linear projection layer, a 300-dimensional hidden layer and a 20K dimensional output layer. Note that the number of parameters for each network is different even though they all have the same projection layer, hidden layer and output layer dimensions. LSTMs utilize approximately 4 times more parameters than RNNs due to the replacement of hidden units with memory blocks. Bidirectional models utilize approximately 2 times more parameters than their unidirectional counterparts due to the computation of a backward hidden state in addition to the forward hidden state.

3.3. Results

While there have been recent efforts to apply RNNLMs directly in speech recognition decoding [18, 19], most past work has evaluated RNNLMs via N-best list rescoring due to their computational expense. As bidirectional models condition word probabilities on words arbitrarily far in the future, it is especially challenging to evaluate these models using first-pass decoding or lattice rescoring. On the other hand, it is straightforward to use bidirectional models to rescore N-best lists.

We evaluate unidirectional and bidirectional RNN and LSTM language models by rescoring 50-best lists obtained by decoding the *rt04* test set with the baseline acoustic and language models. The unidirectional models are linearly interpolated with the baseline language model before evaluation and the interpolation weights are chosen to minimize held-out set perplexity; a weight of 0.3 was found

	WER (%)
Baseline	13.0
Baseline + uni-RNN	12.7
Baseline + bi-RNN	12.5
Baseline + uni-LSTM	12.4
Baseline + bi-LSTM	12.4

Table 2. WERs for unidirectional and bidirectional RNN and LSTM models log-linearly interpolated with the baseline model.

for the RNN and a weight of 0.4 was found for the LSTM. The perplexity and word-error rate results for the unidirectional models are given in Table 1. The LSTM language model yields around 15% relative improvement in perplexity on top of the baseline language model and around 7% relative improvement on top of the RNN language model. The LSTM language model gives a 0.4% absolute WER improvement over the baseline (12.6% vs. 13.0%); this difference is statistically significant at $p < 0.001$.[2] It also outperforms the RNN language model by 0.2% absolute in WER (significant at $p = 0.008$).

Conditional word probabilities obtained from a bidirectional model are conditioned on future inputs in addition to past inputs, while the baseline model is a undirectional model conditioning word probabilities only on past inputs. It is unclear that it makes sense to interpolate bidirectional and unidirectional probabilities at the word level, especially since interpolated word probabilities will be multiplied together to compute utterance probabilities using an equation analogous to Eq. (4). As noted earlier, the product of conditional word probabilities from a bidirectional model is a pseudolikelihood rather than a likelihood.

Instead, we interpolate bidirectional models with the baseline model at the sentence level rather than the word level. We use the bidirectional model to compute the log pseudolikelihood of each hypothesis and treat this as another score to be log-linearly interpolated with the baseline language model log-likelihood score and the acoustic model score. The log-linear interpolation weights are chosen to minimize the WER on the *dev04* set using the simplex algorithm implementation from the SRILM toolkit [20].

The results are given in Table 2. Perplexity results are omitted since it is not straightforward to compute a valid perplexity with a bidirectional language model. We also report log-linear interpolation results with the unidirectional models to provide a fair comparison of unidirectional and bidirectional models. Unidirectional and bidirectional models are indicated with the "uni-" and "bi-" prefixes, respectively. If we compare the WERs in Tables 1 and 2 for the unidirectional models, we see that log-linear interpolation yields lower WERs than linear interpolation. One possible explanation is that choosing weights to optimize perplexity (as was done with linear interpolation) is less than optimal when evaluating word-error rates.

The combination of the baseline model with the unidirectional RNN yields 0.3% absolute improvement in WER over the baseline (significant at $p < 0.001$). The combination of the baseline and bidirectional RNN models reduces the WER from 13.0% to 12.5%, yielding a 0.5% absolute improvement (significant at $p < 0.001$). The combination of the baseline with the unidirectional LSTM model yields a 0.6% improvement over the baseline (significant at $p < 0.001$), our best result. However, no further gains are obtained with a bidirectional LSTM as compared to its unidirectional counterpart.

[2]Statistical significance is measured by the NIST MAPSSWE test.

4. CONCLUSION

In this paper, we compare unidirectional and bidirectional RNNs and LSTMs for language modeling in speech recognition. We discuss the issues that arise in training and evaluating bidirectional language models, propose a method for efficiently training RNNs on a GPU, and show how pseudolikelihoods can be effectively combined with other scores in a speech recognition system. Our experiments on the Broadcast News transcription task indicate:

- LSTM language models are significantly better than RNN language models by 0.3% absolute in WER (significant at $p < 0.001$) and 0.6% absolute over the baseline (significant at $p < 0.001$).
- Bidirectional RNNs outperform unidirectional RNNs by 0.2% absolute (significant at $p < 0.05$). This is an encouraging result demonstrating the potential of bidirectional networks in language modeling.
- Bidirectional LSTMs do not yield any additional gain over unidirectional LSTM models. Further investigation is needed to understand this behavior with larger data sets. One possible factor is the limited number of N-best hypotheses (50) used in our experiments.

5. REFERENCES

[1] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Proceedings of Interspeech*, 2010, pp. 1045–1048.

[2] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, "Extensions of recurrent neural network language model," in *Proceedings of ICASSP*, 2011, pp. 5528–5531.

[3] Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocky, "Strategies for training large scale neural network language models," in *Proceedings of ASRU*, 2011, pp. 196–201.

[4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 2, no. 5, pp. 157–166, 1994.

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

[6] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proceedings of ASRU*, 2013, pp. 273–278.

[7] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," in *ArXiv e-prints*, 2014.

[8] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "LSTM neural networks for language modeling," in *Proceedings of Interspeech*, 2012.

[9] D. Soutner and L. Müller, "Application of LSTM neural networks in language modelling," *Lecture Notes in Computer Science*, pp. 105–112, 2013.

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

[11] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM and other neural network architectures,” *Neural Networks*, vol. 5-6, no. 18, pp. 602–610, 2005.

[12] Alex Graves, Marcus Liwicki, Santiago Fernandez, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 31, pp. 855–868, 2009.

[13] Deyi Xiong, Min Zhang, and Haizhou Li, “Enhancing language models in statistical machine translation with backward n-grams and mutual information triggers,” in *Proceedings of ACL-HLT*, Portland, Oregon, USA, June 2011, pp. 1288–1297.

[14] S. F. Chen, B. Kingsbury, L. Mangu, D. Povey, G. Saon, H. Soltau, and G. Zweig, “Advances in speech transcription at IBM under the DARPA EARS program,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1596–1608, 2006.

[15] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, “Theano: a CPU and GPU math expression compiler,” in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[16] X. Chen, Y. Wang, X. Liu, M.J.F. Gales, and P. C. Woodland, “Efficient GPU-based training of recurrent neural network language models using spliced sentence bunch,” in *Proceedings of Interspeech*, 2014.

[17] Hong-Kwang Jeff Kuo, Ebru Arisoy, Ahmad Emami, and Paul Vozila, “Large scale hierarchical neural network language models,” in *Proceedings of Interspeech*, Portland, Oregon, USA, 2012.

[18] Gwénolé Lecorvé and Petr Motlicek, “Conversion of recurrent neural network language models to weighted finite state transducers for automatic speech recognition,” in *Proceedings of Interspeech*, Portland, Oregon, USA, 2012.

[19] Zhiheng Huang, Geoffrey Zweig, and Benoit Dumoulin, “Cache based recurrent neural network language model inference for first pass speech recognition,” in *Proceedings of ICASSP*, 2014, pp. 6354–6358.

[20] Andreas Stolcke, “SRILM–An extensible language modeling toolkit,” in *Proceedings of ICSLP*, Denver, 2002, vol. 2, pp. 901–904.