Section A-Research paper
ISSN 2063-5346

Enhancing Speech Recognition with Bidirectional Recurrent Neural Networks and LSTM Models

Author 1: Ms. Sheetal Pandya
Assistant professor, CSE department,
Computer Science and engineering department
Indus University, Ahmedabad, Gujarat, India
Email: sheetalpandya.cse@indusuni.ac.in
Author 2: Ms. Krupa Chotai
Assistant Professor,
Computer Engineering Department,
Shah & Anchor Kutchhi Engineering College, Mumbai, India
Email: krupa.chotai@sakec.ac.in

DOI: 10.48047/ecb/2023.12.si4.1626

Abstract

One possible explanation for RNN language models' outsized effectiveness in voice recognition is its superiority over word n-gram models at simulating long-distance context. In RNNs, the hidden-layer recurrent connections are used to simulate the context of prior inputs. RNNs with units that can hold values over infinitely long periods of time are called LSTM neural networks. These RNNs have the potential to mimic a wide range of complicated behaviors. Bidirectional networks may be designed to further condition on the inputs they will acquire in the future, allowing them to make more accurate predictions about future outputs than typical unidirectional networks can by themselves. In this research, we propose using both LSTM and bidirectional RNNs to language modeling for the purpose of speech recognition. Before comparing the efficiency of unidirectional and bidirectional models on a transcription task using English Broadcast News, we discuss some of the potential issues that may arise when employing bi-directional models for speech. We find that bidirectional RNNs perform much better than their unidirectional counterparts, but bidirectional LSTMs show no performance gains over their unidirectional counterparts.

Keywords: Modeling of language, LSTM, RNN, and bidirectional neural networks

1. Introduction

It has been shown that recurrent neural networks can do better than other approaches when it comes to representing language in a variety of settings. In contrast to feed forward neural networks, which are limited number of data points from the past that may be used as inputs, RNNs allow for an unlimited number of such points. Instead, the existence of recurrent connections in a recurrent neural network's hidden layer makes it possible for this kind of network to imitate interactions that last forever. To forecast the likelihood of the next word, which is what the term “n-gram model” of words refers to, RNN language models need to employ a lot more than the customary two to four words that are used in a “n-gram model” of words. This is what is meant by the phrase “n-gram model” of words. However, the wellknown "vanishing gradient" issue has proven that in practice RNNs are unable to effectively employ information from more than around 5–10-time steps back. This is true even if the data is useful. This is what occurs when the RNNs are put through their paces and tested. As time goes on, the exponential decay of the gradient of the error function causes the effect of inputs that occurred in the past to have less and less of an impact on the outcome of the prediction.

LSTM neural networks were established by researchers so that they could circumvent this constraint. These networks were created by exchanging the RNN's non-linear units in the hidden layer with memory blocks that had indefinite storage space. These blocks may be read from, written to, and cleared using a technique called multiplicative gates. One application area in which LSTM neural networks have been shown to perform very well is automatic voice detection. LSTM models have shown to be superior to even the most cutting-edge deep neural networks when it comes to the task of modeling auditory data. In addition, the performance of LSTM language models is superior to that of the more prevalent n-gram and RNN models.

The traditional RNN and LSTM models are unidirectional, which means that they assess the input data in the order that they were received and then calculate output probabilities based only on the context of the data that has already been processed. Unidirectional models do not show the influence that words in the future may have on the likelihood of the current word, despite the fact that the context of the future is obviously very important. Instead, the only way to account for this reliance is to take into account the way in which the present word affects the probability of the words that come after it. If reaping the advantages of model combination calls for explicitly modeling dependencies in both directions, it makes intuitive sense to build models that can go forward and backward in time. This is because modeling dependencies in both directions is necessary for constructing models that can ahead and backward in time.

Bidirectional RNN offer a complicated framework that may combine input from several sources, including the past and the future. This can be useful in a number of contexts. The "backward hidden layer" is a hidden layer learned by bidirectional RNN when the input data is reversed in sequence. Both the forward and the backward hidden layers are linked, not to each other, but rather to the same output layer. This eliminates the need for a direct connection between the two hidden layers. Because it is the initial hidden layer, the forward hidden layer has the name “forward.” The creation of a bidirectional LSTM begins with the construction of a bidirectional RNN and continues with the replacement of the RNN's hidden units with memory blocks. Thanks to the adoption of bidirectional RNN and LSTM network, a number of obstacles, including audio modeling and handwriting recognition, have been successfully overcome. This is just one example of the numerous challenges that have been successfully overcome.

We suggest using bidirectional RNNs and LSTMs in this study to achieve the goal of language modeling within the framework of voice recognition. This is because LSTMs are better at storing information for longer periods of time. When compared to unidirectional language models, bidirectional language models do not provide conditional word probabilities that are capable of being efficiently integrated using the chain rule in order to infer the prospect of a whole speech. This is in contrast to unidirectional language models. The use of conditioning on upcoming inputs adds another layer of complexity to the decoding process for voice recognition. The evaluation of the performance of unidirectional and bidirectional RNNs and LSTMs on a task requiring the transcription of English broadcast news follows a description of the tactics that we utilize to overcome these challenges. When compared to their unidirectional counterparts, bidirectional LSTMs do not show any improvement; however, when compared to their bidirectional RNN counterparts, they do show an improvement of 0.2% absolute.

There are a few different approaches that have been put up as potential ways to explicitly factor in future knowledge while calculating the likelihood of the current outcome. In the field of acoustic modeling, it is possible to have a valid left-to-right model even if one delays the outputs of the model by a few time steps by conditioning on future auditory inputs. The training of two separate models, one based on the original input and the other based on the inverse of the input, is necessary for one of the alternative strategies for language modeling that may be used in the context of machine translation. Estimating the likelihood of a particular speech may be done using either model; the probabilities derived from both sets of data are then merged together to get the final evaluation.

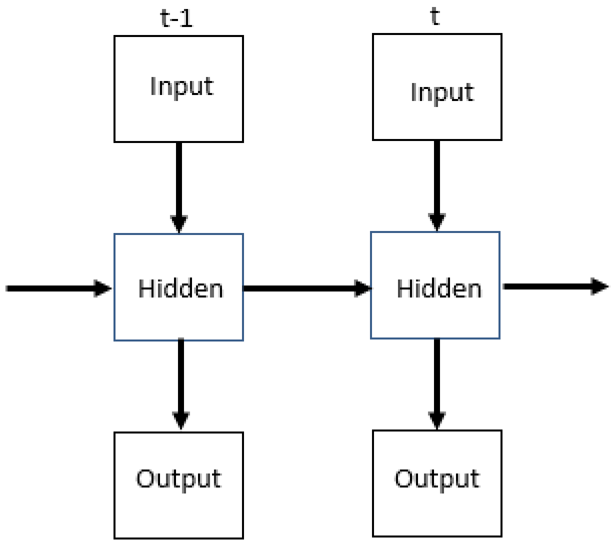


Fig. 1. The RNN architecture is depicted as a temporal unfolding.

2. NEURAL NETWORKS

2.1.RNN

When modeling sequential data, it is usual practice to employ RNN as the modeling tool. The fact that a recurrent neural network's hidden state is able to persistently store a representation of the whole data history is one of the network's advantages. The evolution of a fundamental RNN architecture is shown throughout the course of two time periods in Figure 1. In addition to being reliant on the input time is t, the hidden state time is t −1, which in turn is dependent on the hidden input time is t − 1 and the hidden state time is t − 2, and so on. As may be seen in the accompanying image. RNN activations may be determined explicitly using the following formula when given an input vector series of the kind $X = x_1,...,x_T$ and an output vector sequence of the form $Y = y_1,......,y_T$

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

where h_t represent the unseen layer of vector, W_{xh} represent the input to hidden layer of weight matrix, W_{hh} represent the hidden-to-hidden layer of weight matrix, and W_{hy} represent the output to hidden layer of weight matrix. The value b_h represent the bias in the hidden layer, while the value b_y represent the bias in the output layer. Both of these values are referred to as the bias.
Consider the following example to better understand how RNN language modeling may be used to determine the odds of using conditional words: $P\ (w_t|w_{t-1}|\ h_{t-2})$.

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

where y_t^i represent the ith is the component of the output vector y_t. In this scenario, every output target is a dictionary word. The probability of a word sequence, $W = w_1, w_2, w_T,$ denoted by is computed by multiplying the probabilities of the individual words based on their context.

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

RNN language models do not limit the past to the n - 1 words preceding the current word. This model, when compared to the other two that are widespread, is the one that is most realistic.

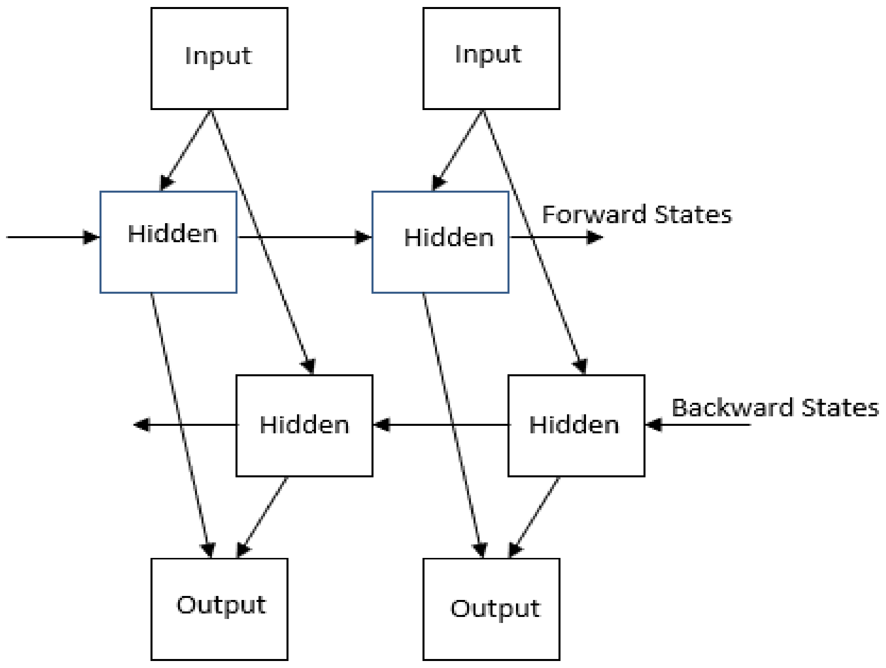


Fig. 2. The temporal unfolding of the Bidirectional RNN architecture is illustrated.

2.2.LSTM Neural Networks

In spite of the fact that RNNs have the potential to make use of infinitely lengthy histories in theory. Neural networks that combine long-term and short-term memories have been suggested as a potential workaround for this problem. Comparable to the memory of a computer, an LSTM neural network is made up of blocks of memory cells that have the capacity to store values, as well as multiplicative gates that have the ability to read (*output*), write (*input*), and reset (*forget*) these values. In the buried layer, these data storage nodes stand in for the RNN's nonlinear units. Memory cells are able to keep information stored for very long periods of time, and the value of a memory cell may be updated by collecting the activations from gates both within and outside of the memory block. Memory cells can also retain information indefinitely.

The LSTM equations may be expressed as:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (5)$$

$$ft = \sigma\left(W_{xf}xt + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \quad (6)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (7)$$

$$Ot = \sigma(W_{xo}xt + W_{ho}h_{t-1} + W_{co}ct + b_o) \quad (8)$$

$$ht = Ot \tanh(ct) \quad (9)$$

The gates at time t are represented by input gate of vector (i_t), forget gate of vector(ft), output gate of vector (Ot), and cell activation of vector (c_t). The letter W stands for the weight matrices that are present between layers, gates, and cells. For example, the letter W_{xi} stands for the weight matrices of the vector that are present between input layer and the input gates. *bi*, *bf*, *bo*, and *bc* are abbreviations that stand for the gate and cell bias components, while the symbol (σ) stands for the logistic sigmoid function. Equations (2) and (3) are used in the process of language modeling once the value of *ht* has been computed in order to derive the conditional word probabilities.

$P(w_t|w_{t-1}, h_{t-2})$.

2.3.Bidirectional RNN

As a result of its capacity to perform analysis of input data in both directions, bidirectional RNNs is able to make the use of not only previous, but also future circumstances. The evolution of the architecture of a bidirectional RNN is shown in Figure 2, which covers a span of time that encompasses two separate time periods. It is possible to discover both the forward hidden layer h_t^F and the backward hidden layer h_t^B in bidirectional RNNs by repeatedly processing. At the point when the network's output is being produced, none of these levels can be seen.

$$h_t^F = \tanh(W_{xh}^F xt + W_{hh}^F h_{t-1}^F + b_h^F) \quad (10)$$

$$h_t^B = \tanh(W_{xh}^B xt + W_{hh}^B h_{t+1}^B + b_h^B) \quad (11)$$

$$yt = W_{hy}^F h_t^F + W_{hy}^B h_t^B + b_y \quad (12)$$

It is possible to generate bidirectional LSTMs by swapping memory blocks for the activations that are included in the hidden layer.

When language modeling is in effect, it is important to keep in mind that the result of one-time step becomes the input for the following time step in the process. When the probabilities from several different time steps are merged, bidirectional models run the risk of producing circular dependencies. We acquire a “pseudolikelihood” rather than the actual likelihood if we take the conditional probabilities from our bidirectional language model and multiply them by themselves. In $P(w_t|w_{t-1}, h_{t-2}, w_t + {}_1, h_t + {}_2)$ contrast to models that only go in one way, this one goes in both directions. Focusing on optimizing the pseudolikelihood of training data as opposed to its likelihood is a fundamental method for training models. This is a straightforward technique. Nevertheless, our primary objective is to increase the amount of applicable training data that is readily available.

3. EXPERIMENTS

3.1. Experimental Set-up

In order to do the task that was required of us, we conducted research with the help of English Broadcast News.

The core of the system is comprised on IBM's GALE speech transcription software, which was first released in 2007. It took 430 hours' worth of audio from broadcast news to build and train the speaker-adaptive audio model that is capable of discriminative listening. The model was trained using audio from to broadcast the news. Using a total of 350 million words from various sources, the baseline language model is based on the traditional word 4-gram model. The size of the vocabulary and the baseline language model are around 84,000 words in length. There is about 48,000 words' worth of material included inside the held-out set. The language model that was used as the baseline for this particular dataset made the discovery that there was a WER (Word Error Rate) of 13.1%.

3.2. Training of Language model via RNNs and LSTMs.

To train the language Models such as RNN and LSTM, just a tiny fraction of the original 350 million words included in the corpus, particularly 12 million words, is used. Our team developed a training method of unidirectional and bidirectional Language model of RNN and LSTM by making use of the Theano toolkit's extensive feature set. Mini-batch training is feeding the number of phrases into the network in an unsupervised fashion and then modifying the network parameters based on how the words in those sentences are processed. This is done in order to learn how best to train the network. This is done so that the network may be trained without the monitoring of human beings. In order to make the most of the lightning-fast matrix operations that can be performed on a GPU, it is essential that each phrase in a minibatch have the same length. This is the only way to ensure optimal performance. It is possible to carry out these procedures. On the other hand, particularly in the realm of broadcast news, you'll discover that the length of a sentence may vary quite a bit from one report to the next. This is especially true when it comes to the length of the sentence. The potential speedup that may be realized by deploying a GPU is decreased when phrases that are shorter are extended using special symbols to make them the same length as phrases that are longer. This is done in order to make the phrases seem more uniform in length.

The RNN and LSTM language models successfully works on GPUs, in the training data into a single long word sequence, and then we partition this sequence into words of equal length. This allows us to train the RNN and LSTM language models effectively on the GPUs. Because of this, we are able to train these models on GPUs in a way that is both effective and efficient. Since an average phrase contains 18 words, the length of the sequence will be equal to that of the average phrase in the training data. During the course of our research, we divided the data into smaller batches of eight sequences each. To signify the start of the learning process, the initial hidden state vector is made up completely of zeros when it is assessed at time *t = 0*. Once a mini-batch of word sequences is processed, adjustments are made to the initial hidden state vector and other network parameters. These modifications are then incorporated into the process of populating hidden state vectors for the subsequent data batch. Prior to concluding the evaluation of a model, the value of the hidden state vector is updated to align with the latest version of the test data for each phrase. This occurs before the evaluation is complete. This takes place before it is determined that the assessment is finished. It is important to keep in mind that our solution may not provide the speed that was promised if the word sequence boundaries are set in an arbitrary way. The findings of our research, on the other hand, suggest that the aforementioned algorithm achieves a satisfactory compromise between the two opposing concerns of training speed and accuracy. Separating sentences into fragments of about the same length is an additional strategy that may be used in order to accomplish rapid RNNLM training on GPUs. This strategy involves breaking down phrases into fragments. In order to keep the number of batches at a constant level, the strategy's primary objective is to cut down on the amount of padding that has to be employed.
Table 1 Complexities and word error rates (WERs) are provided for linearly interpolated LSTM and RNN models using the baseline model.

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

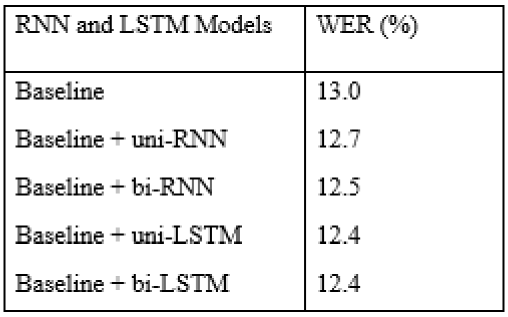
When training or evaluating a neural network language model, it is usual for the necessary computation to be dominated by the recurrence that are required to make the output layer of the model. This is because these multiplications are essential to generate the layer. This holds true not just throughout the phase of the model devoted to training but also during the whole of the phase devoted to evaluation. In order to minimize our expenses to a bare minimum while producing output, we force ourselves to stick to the 20,000 words that are most often found in the dictionary. The "out-of-shortlist" class contains all of the terms that were considered for inclusion in the eventual output vocabulary; however, it was ultimately decided not to include any of these words. It is presumable that the probabilities of occurrence of any individual phrase included in this category are comparable to one another. If the output of the output layer of a neural network language model is categorized and separated, this may help to reduce the amount of computer complexity that the model needs. In order to reduce the overall number of parameters, we have made the incorporation of a projection layer into the input process part of its overall structure. Using a weight matrix that is shared by all of the lexicon locations, a position on a d-dimensional continuous feature vector has been determined for each and every word that is included in the lexicon. In Equation 1, the variable x_t represents the continuous properties of the word w_t, which is the focus of the discussion at the moment. Back propagation is used to learn continuous feature vector representations of words, together with their associated weight matrices and biases. This process takes place over a period of time and takes place during the course of back propagation.

We make use of a linear projection layer that has 180 dimensions, a hidden layer that has 300 dimensions, and an output layer that has 20K dimensions in order to train language models such as unidirectional and bidirectional RNNs and LSTMs. Despite the fact that all networks have the same size for their projection layer, hidden layer, and output layer accordingly, the number of available parameters differs from network to network. LSTMs need about four times the amount of parameters that are required by RNNs. This is because LSTMs store information in memory blocks rather than hidden units. In order to calculate the forward hidden state and the backward hidden state, bidirectional models need approximately twice as many parameters than their unidirectional counterparts do. Unidirectional models only need to account for one hidden state at a time. This is because bidirectional models need taking into account both the forward and backward movement of the vehicle.

3.3.Results

In spite of recent attempts to employ RNNLMs directly in speech recognition decoding, the majority of previous research has assessed RNNLMs via the use of N-best list rescoring because of the amount of processing power they demand. This is because, in bidirectional models, the probability of one word is conditionally determined by the likelihood of another word. However, rescoring N-best lists by using bidirectional models is a technique that is uncomplicated and simple to carry out.

Table 2. The WERs of the RNN And LSTM Models are inserted from the WERs of the baseline model.



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

It is possible to compare the efficacy of unidirectional and bidirectional Recurrent Neural Network and Long Short-Term Memory language models by rescoring the 50 top lists. By doing so, we will be able to establish which model is the more successful option. Before carrying out the assessment, the unidirectional models are given a linear interpolation using the default language model. The RNN was given a weight of 0.3 throughout the interpolation phase, while the LSTM was given a weight of 0.4. Both of these weights were considered to be appropriate. The held-out set was made to be less confusing by doing this, therefore it was done. The results of the study are shown in Table 1, which breaks down both the word mistake rate and the overall degree of confusion caused by the unidirectional models. The LSTM language model is around 15% more effective than its counterpart when measured against the baseline language model. On the other hand, the RNN language model is approximately 7% more successful when measured against the LSTM language model. When compared to the baseline, the LSTM language model obtains a WER improvement of 0.4% (12.6% versus 13.0%), which is a difference that is important in terms of statistics($p < 0.001$ for all comparisons). It also performs better than the RNN language model in terms of the word error rate (WER), and this difference is statistically significant ($p = 0.008$ for the test). The absolute margin of improvement is 0.2%.

The baseline model calculates conditional word probabilities, but it only considers data from the previous iteration of the task. On the other hand, the conditional word probabilities that are generated by a bidirectional model take into consideration not just previous data but also data from the future in addition to historical data. The wisdom of extrapolating probability in both directions from the word level is not immediately obvious to me. When you take into account the fact that the probabilities of the interpolated words will be multiplied together in order to derive the probability of the utterances using an equation that is equivalent to Eq. (4), you will see that this is particularly true. This is because likelihood and pseudolikelihood are not synonymous terms. The reason for this is because the probability does not take into account the fact that the model may be used in both directions.

When we are interpolating bidirectional models with the baseline model, we focus instead on the phrase level as our point of reference. This allows us to more accurately represent the data. As a consequence of this, the data might potentially be represented with a better degree of accuracy at this time. Utilizing the bidirectional model allows us to calculate each hypothesis' log pseudolikelihood and compare them. The log likelihood score from the baseline language model and the acoustical model score are added to this third score to get the final score, which is then interpolated logarithmically. Using the simplex method implementation that is included as a component of the SRILM toolbox, the log-linear interpolation weights are chosen in such a way as to guarantee that the word error rate on the *dev04* data set is brought down to its bare minimum. This is accomplished by ensuring that the weights are chosen in such a manner as to ensure that the WER is lowered to its bare minimum.

You may find a summary of the facts in Table 2, which is located lower down the page. Because it is difficult to accurately quantify perplexity while also using a bidirectional language model, the findings of this model have been ignored. This is due to the fact that the model incorporates both directions of language. We also disclose the results of log-linear interpolation using the unidirectional models so that the comparison between the two may be done in a fair and accurate manner. The prefix "*uni*" is given to models that are able to move in just one way, while the prefix "*bi*" is given to models that are able to move in either direction. We can see that when we use the linear interpolation technique, the results in WERs for the unidirectional models are lower when we use the log-linear interpolation approach rather than the linear interpolation method. This can be seen when we compare the log-linear interpolation approach to the linear interpolation method. Both Table 1 and Table 2 illustrate this point. It is feasible to calculate word-error rates by applying linear interpolation; however, choosing weights with the intention of decreasing ambiguity may not be the most efficient method.

The WER is improved by 0.3% when the baseline model is used in conjunction with the unidirectional RNN. This improvement is statistically significant when compared to the baseline; $p < 0.001$ indicates that it is significantly better than the baseline. When the baseline model and the bidirectional RNN model are coupled, the WER is increased by 0.5% in absolute terms, and this improvement is significant at the $p < 0.001$ level. When we first started, the WER was 13.0%. The current rate is 12.5%. The baseline, when combined with a unidirectional LSTM model, produced the most impressive results for us. This resulted in the highest degree of precision for us. The combination resulted in an increase that was statistically significant ($p < 0.001$) and which was 0.6% more than the group that was used as the control. On the other hand, a bidirectional LSTM does not provide any major benefits over its unidirectional equivalent in the sense that it may improve performance.

4. CONCLUSION

When it comes to modeling language for the sake of speech recognition, this article investigates the distinctions that may be made between unidirectional and bidirectional RNNs as well as LSTMs. LSTMs are taken into consideration as well. Finally, we show how to provide a Procedure for effectively training RNNs on a GPU. We also examine the difficulties encountered during training and assessing bidirectional language models. The following observations emerged from our training exercises including the duty of transcribing television news:

- Long short term memory language models perform much better than Recurrent Neural network language models, with LSTM language models having an absolute advantage of 0.4% in WER in comparison to RNN language models.
- The performance of bidirectional RNNs is much better than that of their unidirectional counterparts ($p < 0.05$). It is interesting because this result demonstrates the possibility of adopting bidirectional networks in language modeling.
- In terms of performance, it has not been shown that bidirectional LSTM models are clearly better than unidirectional LSTM models. To have an accurate comprehension of this trend, more research with more extensive data sets is required.

REFERENCES

[1] Mikolov, T., Karafiat, M., Burget, L., Cernocky, J. & Khudanpur, S. (2010). Recurrent neural network-based language model. *In Proceedings of Interspeech*, *20*(1), 1045– 1048.

[2] Mikolov, T., Kombrink, M., Burget, L., Cernocky, J. & Khudanpur, S. (2011). Extensions of recurrent neural network language model. *In Proceedings of ICASSP*, 5528–5531.

[3] Mikolov, T., Deoras, A., Povey, D., Burget, L., & Cernocky, J. (2011). Strategies for training large scale neural network language models. *In Proceedings of ASRU*, 196–201.

[4] Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 2(5), 157– 166.

[5] Hochreiter S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 8*(9), 1735–1780.

[6] Graves, A., Jaitly, N. & Mohamed, A. (2013). Hybrid speech recognition with deep bidirectional LSTM. *In Proceedings of ASRU*, *20*(1), 273–278.

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

[8] Martin Sundermeyer, Ralf Schluter,¨ and Hermann Ney, LSTM neural networks for language modelling. *In Proceedings of Interspeech*, 2012.

[9] Soutner, D. & Muller, L. (2013). Application of LSTM neural net-¨ works in language modelling. *Lecture Notes in Computer Science*, 105–112.

[10] Schuster, M. & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

[11] Graves, A. & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *5*(6), 602–610.

[12] Graves, A., Liwicki, R., Fernandez, S., Bertolami, R., Bunke, H. & Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(31), 855–868.

[13] Xiong, D., Zhang, M. & Li, H. (2011). Enhancing language models in statistical machine translation with backward n-grams and mutual information triggers. *In Proceedings of ACL-HLT*, 1288–1297.

[14] Chen, S. F., Kingsbury, B., Mangu, L., Povey, D., Saon, G., Soltau, H. & Zweig, G. (2006). Advances in speech transcription at IBM under the DARPA EARS program. *IEEE Transactions on Audio, Speech, and Language Processing*, *14(*5),1596–1608.

[15] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D. & Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. *In Proceedings of the Python for Scientific Computing Conference (SciPy),*

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

[17] Jeff Kuo, H. K., Arisoy, E., Emami, A. & Vozila, P. (2012). Large scale hierarchical neural network language models. *In Proceedings of Interspeech*.

[18] Huang, Z., Zweig, G. & Dumoulin, B. (2014). Cache based recurrent neural network language model inference for first pass speech recognition. *In Proceedings of ICASSP*, *23*(1), 6354–6358.

[19] Stolcke, A. (2002). SRILM–An extensible language modeling toolkit. *In Proceedings of ICSLP*, *2*, 901–904.