



Multi-Stream Long Short-Term Memory Neural Network Language Model

Ebru Arisoy¹, Murat Saraclar²

¹MEF University, Istanbul, Turkey

²Bogazici University, Istanbul, Turkey

ebruarisoy.saraclar@mef.edu.tr, murat.saraclar@boun.edu.tr

Abstract

Long Short-Term Memory (LSTM) neural networks are recurrent neural networks that contain memory units that can store contextual information from past inputs for arbitrary amounts of time. A typical LSTM neural network language model is trained by feeding an input sequence. i.e., a stream of words, to the input layer of the network and the output layer predicts the probability of the next word given the past inputs in the sequence. In this paper we introduce a multi-stream LSTM neural network language model where multiple asynchronous input sequences are fed to the network as parallel streams while predicting the output word sequence. For our experiments, we use a sub-word sequence in addition to a word sequence as the input streams, which allows joint training of the LSTM neural network language model using both information sources.

Index Terms: Language modeling, long short-term memory, sub-word-based language modeling

1. Introduction

Neural network language models (NNLMs) have achieved very good performance in automatic speech recognition systems. Feedforward NNLMs [1, 2, 3, 4], recurrent NNLMs (RNNLMs) [5, 6] and long short-term memory (LSTM) NNLMs [7, 8] have been shown to yield both perplexity and word error rate (WER) improvements as compared to conventional n -gram language models.

In a feedforward NNLM, the input to the neural network is the words in the n -gram history and the output is a probability distribution over the predicted word. Whereas in a RNNLM, the history is not restricted to a fixed number of previous words, i.e., $n-1$ words. RNNLMs utilize the hidden layer activations, with the help of the recurrent connections at the hidden layer, as well as the current word to predict the next word. This corresponds to using a very long-range history while predicting the probability of the next word. However, in practice recurrent neural networks cannot effectively use information beyond about 5–10 time steps back, due to the well-known “vanishing gradient” problem [9]. The gradient of the error function decays exponentially over time, reducing the influence of inputs far back in time. 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 [10]. LSTM neural networks have been shown to yield superior performance both for acoustic and language modeling. In acoustic modeling, LSTM models outperform state-of-the-art deep neural networks [11, 12]. In language modeling, LSTM models give better performance than conventional n -gram and RNN language models [7, 8, 13].

An RNN language model, as well as an LSTM neural network language model, is typically trained by feeding a word sequence to the input layer. In this paper we propose a new LSTM neural network language model architecture which allows using multiple arbitrary length asynchronous input sequences while predicting the probability of the next word. We call this model *multi-stream LSTM language model*. In the proposed model, multiple input sequences are fed to the network as parallel streams and inputs from each stream are propagated to parallel hidden layers. Multiple parallel hidden layers are then connected to the same output layer to predict the probability of the next word. Since the streams fed to the input layer are asynchronous, the hidden layers are connected to the output layer only when the next word is predicted. The proposed multi-stream model allows using multiple information sources in LSTMs and gives the freedom of using arbitrary length asynchronous sequences as the information sources. For instance, in our experiments we use words and sub-words as two parallel streams where a word may correspond to several sub-words.

The idea of using multiple information sources in neural network language modeling has been implemented before and it was shown that the neural network models benefitted from the extra information provided. Syntactic features were investigated in feedforward NNLMs [14, 15]. Contextual features such as Part-of-Speech (PoS) tags, lemmas, topics and the socio-situational setting of a conversation were investigated in RNNLMs [16]. Topic features extracted by Latent Dirichlet Allocation were also explored in RNNLMs [17]. In all these works, each input word has the same number of associated features so that word and feature inputs fed to the network can be considered as synchronous sequences. Our proposed model distinguishes from the previous work in the sense that it does not restrict a word to have the same number of associated features. In fact, a word can have an arbitrary length feature sequence associated with it and our proposed multi-stream model allows these asynchronous sequences to be utilized in the neural network language model. In our multi-stream LSTM neural network language model approach, a sub-word sequence as well as a word sequence is fed to the LSTM network to predict the output word sequence. Using sub-word inputs in LSTMs to predict word sequences was also explored in [18] by representing a word as the sum of the fixed length input vectors representing its morphemes and the word itself. Note that this approach merges word and sub-word features at the feature-level, whereas our approach jointly trains a model using asynchronous word and sub-word sequences.

The rest of the paper is organized as follows: Section 2 explains the multi-stream LSTM language model. Experiments and results are described in Section 3, and Section 4 presents conclusions.

2. Multi-Stream LSTM Language Model

In this section we will first explain the LSTM neural network language model and then give the details of the proposed multi-stream LSTM approach.

2.1. LSTM language model

Long Short-Term Memory neural networks are recurrent neural networks proposed to solve the vanishing gradient problem of RNNs. The upper part of Figure 1, after removing the hidden to output layer connections coming from the lower part, shows a basic RNN/LSTM architecture. Here, the neural network language model is composed of input, projection, hidden and output layers. Each word in the vocabulary is encoded by 1-to- V coding where V is the size of the input vocabulary. In 1-to- V coding, each word in the vocabulary is represented by a V dimensional sparse vector where only the index of that word is 1, shown with a dot at the input layer in Figure 1, and the rest of the entries are 0. Then the input vector is mapped into a linear projection layer. Projection layer is followed by a recurrent hidden layer and the hidden layer is connected to the output layer. Each target at the output layer corresponds to a word in the vocabulary so that output layer produces a probability distribution over the predicted word. For instance, the dot at the output layer in Figure 1 represents the probability of the i 'th word in the vocabulary given the previous words in the input sequence as the history.

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

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

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. Note that in Equation 1, x_t represents the projection layer vector of w_t , the word at time t in the input word sequence.

In RNN language modeling, the conditional word probabilities $P(w|h)$ 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 i th element of the output vector y_t . Note that using the previous word as well as the previous hidden layer state while predicting the probability of the next word in RNNs translates to predicting the n -gram probability using a long-range of previous words in the history.

Even though RNNs potentially utilize arbitrarily long histories, in practice the effective context length of an RNN is quite limited. LSTMs remedy this limitation by replacing 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 hidden layer activations of an LSTM neural network is

computed as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (4)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (5)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (6)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (7)$$

$$h_t = o_t \tanh(c_t) \quad (8)$$

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 at the output layer are calculated using Eqs. (2) and (3). Since LSTMs solve the vanishing gradient problem in RNNs, the n -gram probabilities in LSTM neural network language models contain a much longer history context than RNN language models.

2.2. Multi-stream LSTM language model

We propose a multi-stream LSTM neural network for jointly modeling multiple arbitrary length sequences. In language modeling, the main sequence corresponds to a word sequence and the other sequence is a feature sequence associated with the given word sequence. Note that the word and the feature sequences will be asynchronous if each word has different number of sequential features. The output word sequence is predicted using these multiple asynchronous input streams together in LSTM neural network language model framework.

The architecture of the proposed model is shown in Figure 1. The upper part of the figure, after removing the hidden to output layer connections coming from the lower part, shows a typical LSTM neural network architecture. This upper part takes a sequence of words as input and predicts the probability of the next word using Equations 2-8. We propose adding a similar architecture containing input, projection and recurrent hidden layers to the original LSTM model for another input stream, e.g., a sub-word stream, obtained by splitting words into meaningful units. The parallel input, projection and hidden layers are not directly connected; instead, the hidden layers both connect to the same output layer. Therefore, each parallel network takes a different input stream and combines the information coming from these streams at the output layer. Here it is important to note that the input streams, e.g., the word stream and the sub-word stream, are not synchronous but both streams predict the output sequence at the word level. Therefore, in our multi-stream model, the connections from the hidden layer of the sub-word stream to the output layer, shown with dashed lines in Figure 1, are active only when a word output is predicted.

Figure 2 shows the multi-stream LSTM model architecture unfolded in time for several time steps. Here, w_i represents projection layer vector of the words in the input word sequence and sw_{ij} represents projection layer vector of the sub-words in the associated sub-word sequence, where sw_{ij} is the vector input for the j 'th sub-word of the i 'th word. Sub-words are meaningful segmentations of words into smaller units, e.g., morphological segmentations, so that they also produce a sequential input for LSTM modeling. In Figure 2, word and sub-word input vectors are fed into different recurrent hidden layers. The word stream produces outputs at every time step, whereas

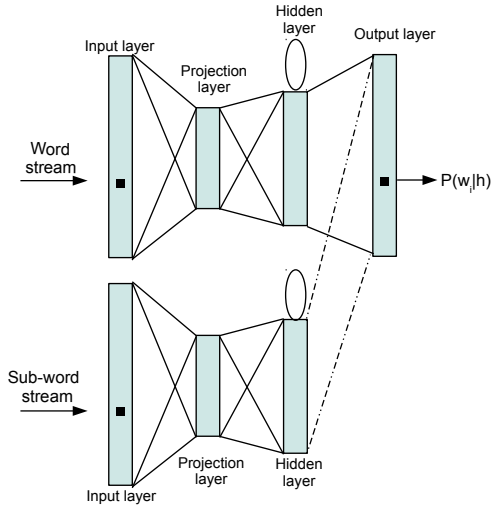


Figure 1: Multi-stream LSTM architecture

the sub-word stream produces outputs only at the time steps corresponding to the word boundaries. However, the sub-word hidden layer recursively accumulates information coming from previous time steps even without producing outputs.

Note that there are several ways of combining multiple information sources in neural network modeling, such as feature-level combination, model-level combination and joint training. In feature-level combination, features associated with a given word are directly appended to the input vector of that word to form a fixed-length input vector for neural network training as in [14, 15, 16, 17]. In case of different number of features for each word, *e.g.*, when morphological segmentations of a word are used as features, a fixed-length input vector is formed by summing vector representations of the features as in [18]. In model-level combination, separate models trained on multiple information sources are combined. For instance, a neural network language model trained on a word sequence can be interpolated with another model trained on sub-word sequences. In joint training, a single model is trained using multiple information sources together. Our multi-stream LSTM framework is a joint training approach which provides a novel LSTM neural network language model.

3. Experiments

We evaluate the performance of single-stream and multi-stream LSTM neural network language models in both automatic speech recognition and key word search tasks. Following sections describe the experimental set-up and the results.

3.1. Experimental Set-up

Experiments are performed on the IARPA Babel Turkish limited language pack (LimitedLP) dataset which contains conversational telephone speech. Automatic speech recognition (ASR) and keyword search (KWS) systems are built using the Kaldi toolkit [19] following the Babel recipe¹. The acoustic model is a DNN with p -norm activations [20]. The 3-gram lan-

¹<http://svn.code.sf.net/p/kaldi/code/trunk/egs/babel/s5c>

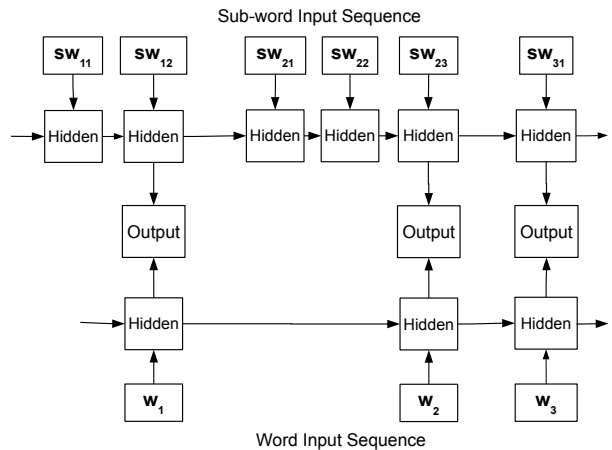


Figure 2: Multi-stream LSTM architecture unfolded in time.

guage model is trained using SRILM toolkit [21] on the reference transcriptions of the acoustic training data consisting of approximately 10K utterances and 74K words. The language model vocabulary contains 10K words. The performance of the ASR and KWS systems are evaluated on a development set containing 10 hours of acoustic data.

In LSTM language modeling, we use both the words and their sub-word segmentations. The sub-word segmentations can be obtained either with grammatical or statistical approaches. Grammatical sub-word units are obtained with rule-based morphological analyzers and statistical sub-word units are obtained with data-driven approaches. In our research, we prefer using a statistical approach to obtain the sub-word sequences since statistical approaches do not require any linguistic knowledge of the language which can be quite important in IARPA Babel project running on multiple under-resourced languages. We use the Morfessor algorithm [23, 24], with publicly available software, to segment words into statistical sub-word units. Morfessor algorithm utilizes the Minimum Description Length principle while learning a representation of the language from the given data. Segmenting the word vocabulary, 10K words, into sub-words using the Morfessor algorithm results in a sub-word vocabulary containing around 3K units.

We train two single-stream and one multi-stream LSTM neural network language models using the reference transcriptions of the IARPA Babel Turkish limitedLP dataset. We implemented training algorithms for single-stream and multi-stream LSTM language models using the Theano library [22]. The first two single-stream models are trained on word and sub-word sequences of the text data respectively. The third model is a multi-stream model where LSTM neural network is trained on parallel word and sub-word streams. Note that all the models predict output sequences at the word level even in the single-stream sub-word model. All three LSTM neural network language models are trained using a 60-dimensional linear projection layer, a 100-dimensional hidden layer and a 10K dimensional output layer. The input layer dimensions for word and sub-word input layers are 10K and 3K respectively.

Models	WER (%)	MTWV (IV)
Baseline	62.3	0.2783
Baseline + single-stream (word)	61.8	0.2839
Baseline + single-stream (sub-word)	61.8	0.2822
Baseline + multi-stream (word + sub-word)	61.8	0.2874

Table 1: ASR and KWS performances.

3.2. Results

The ASR performance of the LSTM neural network language models are evaluated in 500-best rescoring. LSTM neural network language models are linearly interpolated with the baseline 3-gram language model before rescoring. We use 0.5 as the interpolation weight. Interpolation weight is tuned to minimize the perplexity of a text data chosen from the development set containing 2 hours of acoustic data.

The word error rate (WER) results are given in Table 1. Both of the single-stream models and the multi-stream model yield 0.5% absolute improvement on top of the baseline language model. This improvement is statistically significant at $p < 0.001$.² However, multi-stream model does not give any further gains compared to single-stream models.

For the KWS evaluations, we convert the 500-best lists back into lattices. Note that this conversion results in significantly smaller lattices and degrades the KWS performance. Nevertheless, we report the KWS performance as a proof of concept for showing effectiveness of LSTM language models in KWS. The Maximum Term Weighted Value (MTWV) results for in-vocabulary (IV) queries are given in Table 1. The best MTWV is obtained with the multi-stream (word + sub-word) model resulting in 0.009 improvement over the 500-best baseline.

4. Conclusions

In this study, we introduce a multi-stream LSTM neural network language model which combines the information from multiple asynchronous input sequences to predict an output sequence. The combination is performed by connecting parallel hidden layers of LSTMs with the output layer. The advantage of the proposed model is to utilize arbitrary length input sequences in the LSTM framework. We evaluate the proposed model in ASR and KWS tasks. Both single-stream and multi-stream LSTM language models outperform the baseline in terms of ASR performance measured in WER. However, no gain in WER is obtained with the proposed multi-stream LSTM approach. On the other hand the best KWS performance is obtained using the multi-stream LSTM neural network language model. The improved KWS performance suggests that LSTM language models yield a better ranking of the N-best hypotheses for KWS.

In the future we are planning to investigate the multi-stream LSTM with alternative sub-word units, such as syllables, and to combine word and sub-word level information at earlier layers of the network, such as the projection and hidden layers. We are also planning to train multi-stream LSTM language models on larger corpora. Furthermore, the impact of LSTM language models in KWS performance should be investigated using a lattice rescoring framework.

²Statistical significance is measured by the NIST MAPSSWE test.

5. Acknowledgements

This study uses the IARPA Babel Program base period language collection release babel105b-v0.4, supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

6. References

- [1] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.
- [2] H. Schwenk and J.-L. Gauvain, "Training neural network language models on very large corpora," in *Proceedings of HLT-EMNLP*, 2005, pp. 201–208.
- [3] H. Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, no. 3, pp. 492–518, Jul. 2007.
- [4] H.-K. J. Kuo, E. Arisoy, A. Emami, and P. Vozila, "Large scale hierarchical neural network language models," in *Proceedings of Interspeech*, Portland, Oregon, USA, 2012.
- [5] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of Interspeech*, 2010, pp. 1045–1048.
- [6] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proceedings of ICASSP*, 2011, pp. 5528–5531.
- [7] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Proceedings of Interspeech*, 2012.
- [8] D. Soutner and L. Müller, "Application of LSTM neural networks in language modelling," *Lecture Notes in Computer Science*, pp. 105–112, 2013.
- [9] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 2, no. 5, pp. 157–166, 1994.
- [10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 8, no. 9, pp. 1735–1780, 1997.
- [11] A. Graves, N. Jaitly, and A. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proceedings of ASRU*, 2013, pp. 273–278.
- [12] 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.
- [13] E. Arisoy, A. Sethy, B. Ramabhadran, and S. Chen, "Bidirectional recurrent neural network language models for automatic speech recognition," in *Proceedings of ICASSP*, 2015.
- [14] A. Emami, "A neural syntactic language model," Ph.D. dissertation, Johns Hopkins University, Baltimore, MD, USA, 2006.
- [15] H.-K. J. Kuo, L. Mangu, A. Emami, I. Zitouni, and Y.-S. Lee, "Syntactic features for Arabic speech recognition," in *Proceedings of ASRU*, Merano, Italy, 2009, pp. 327–332.
- [16] Y. Shi, P. Wiggers, Catholijn, and M. Jonker, "Towards recurrent neural networks language models with linguistic and contextual features," in *Proceedings of Interspeech*, 2012, pp. 1664–1667.
- [17] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *Spoken Language Technologies*. IEEE, 2012.

- [18] D. Renshaw and K. B. Hall, “Long short-term memory language models with additive morphological features for automatic speech recognition,” in *Proceedings of ICASSP*, 2015.
- [19] D. Povey *et al.*, “The kaldi speech recognition toolkit,” in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011.
- [20] J. Trmal, G. Chen, D. Povey, S. Khudanpur, P. Ghahremani, X. Zhang, V. Manohar, C. Liu, A. Jansen, D. Klakow *et al.*, “A keyword search system using open source software,” in *Proceedings of the Workshop on Spoken Language Technologies (SLT)*, 2014.
- [21] A. Stolcke, “SRILM—An extensible language modeling toolkit,” in *Proceedings of ICSLP*, vol. 2, Denver, 2002, pp. 901–904.
- [22] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, “Theano: a CPU and GPU math expression compiler,” in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Jun. 2010.
- [23] M. Creutz and K. Lagus, “Unsupervised discovery of morpheme,” in *Proceedings of ACL*, 2002, pp. 21–30.
- [24] —, “Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0,” Helsinki University of Technology, Publications in Computer and Information Science Report A81, March 2005.