COMPARISON OF FEEDFORWARD AND RECURRENT NEURAL NETWORK LANGUAGE MODELS

M. Sundermeyer[1], *I. Oparin*[2,*], *J.-L. Gauvain*[2], *B. Freiberg*[1], *R. Schlüter*[1], *H. Ney*[1,2]

[1]Human Language Technology and Pattern Recognition,
Computer Science Department, RWTH Aachen University, Aachen, Germany
[2]Spoken Language Processing Group, LIMSI CNRS, Paris, France
{sundermeyer,schluter,ney}@cs.rwth-aachen.de {oparin,gauvain}@limsi.fr

ABSTRACT

Research on language modeling for speech recognition has increasingly focused on the application of neural networks. Two competing concepts have been developed: On the one hand, feedforward neural networks representing an n-gram approach, on the other hand recurrent neural networks that may learn context dependencies spanning more than a fixed number of predecessor words.

To the best of our knowledge, no comparison has been carried out between feedforward and state-of-the-art recurrent networks when applied to speech recognition. This paper analyzes this aspect in detail on a well-tuned French speech recognition task. In addition, we propose a simple and efficient method to normalize language model probabilities across different vocabularies, and we show how to speed up training of recurrent neural networks by parallelization.

Index Terms— Automatic speech recognition, feedforward neural networks, recurrent neural networks

1. INTRODUCTION

Apart from the acoustic signal, today's speech recognition systems also make use of prior knowledge about the underlying human language. Such information is usually incorporated in the form of a probabilistic language model (LM) which assigns non-zero probabilties to arbitrary word sequences w_1^N. By assigning higher probabilities $p(w_1^N)$ to word sequences more commonly found in a natural language, a speech recognizer can identify those words that have most likely been spoken.

For state-of-the-art systems, usually a set of hypotheses for the spoken word sequence is generated using a backing-off language model ([1]). In a rescoring step, additional techniques are applied to obtain more accurate estimates for $p(w_1^N)$ on this limited set of hypotheses only. Based on the updated language model probabilities, the final recognition result is obtained.

Large improvements have been reported when applying neural network language models (NNLMs) [2] during rescoring (see e. g. [3], [4], and [5]). However, there are fundamental differences in the way neural networks have previously been applied to speech recognition tasks.

When a feedforward neural network (FFNN) is used, only the direct $(n-1)$ predecessor words w_{i-n+1}^{i-1} are used to predict the probability of the current word w_i. Although it is possible to include words from the previous sentence, most of the time the history is trunacted at the beginning of the sentence in the n-gram approach. When a recurrent neural network is used, the full sequence of predecessor words w_1^{i-1} is considered for predicting w_i, see [5].

On the other hand, it is not possible to apply recurrent neural networks to the rescoring of standard word lattices. Instead, only a subset of the hypotheses encoded in a word lattice can be considered and needs to be converted into a linear, non-branching format denoted as m-best list.

Furthermore, recurrent models cannot be consistently evaluated in rescoring: A speech recognizer only gives multiple hypotheses for a single sentence-like portion of the speech signal. To cover context lengths spanning e. g. two consecutive sentences, any two hypotheses for the two sentences have to be concatenated and rescored with the recurrent NNLM. For larger m-best lists and longer contexts, such an approach is computationally infeasible. As a result, in practice only the best scoring hypothesis of a sentence is used when rescoring the following sentence. Eventually, the perplexities computed with recurrent models may not fully reflect the corresponding performance in terms of word error rate (WER) because when computing perplexities, the problem of exponential growth of the hypothesis space does not exist, and the recurrent model can be evaluated in an exact manner.

For these reasons, it seems interesting to compare both approaches in more detail. In this work, we want to find out whether the increased complexity of recurrent models compared to feedforward networks pays off in performance.

2. RELATION TO PRIOR WORK

In recent works, the comparison of feedforward and recurrent neural networks has already been investigated to some extent. First experiments along this direction were presented

[*]Now with LNE, the French National Metrology and Testing Laboratory.

in [6], where a recurrent NNLM attained a significantly lower perplexity than a feedforward network trained on one million running words of Wall Street Journal (WSJ) data.

In [7], deep feedforward NNLMs were analyzed. In addition, they were also compared to recurrent networks on a selection of WSJ training data consisting of about 24 million running words. Again, the recurrent network obtained consistently lower perplexities than the feedforward architecture. However, for the recurrent network, the full vocabulary was used whereas the feedforward network was trained on a shortlist vocabulary only. Therefore, it is not obvious whether the improvements stem from the network architecture itself or the use of all vocabulary words (which was shown to be beneficial in [8]).

Contrasting these results, in [9] a 10-gram feedforward NNLM performed even slightly better in perplexity than a recurrent network on a large-scale English to French translation task. Performance in terms of BLEU was identical for both network architectures. Here, the recurrent model was integrated into a feedforward training scheme by applying several approximations to speed up the training procedure.

To the best of our knowledge, no comparative analysis has been carried out that takes into account the impact of neural network LMs on speech recognition performance. On the other hand, it is well known that improvements in perplexity do not necessarily carry over to improvements in WER, see e. g. [10]. Furthermore, no prior work has compared feedforward networks to recurrent Long Short-Term Memory (LSTM) NNLMs that were shown to improve over standard recurrent NNLMs in [5]. In this paper, we address these open issues.

3. NEURAL NETWORK LANGUAGE MODELS

As an example for the feedforward NNLM architecture used here, a trigram NNLM is depicted in Fig. 1. At the input layer, the predecessor words u and v are fed into the network, where the words are represented as 1-of-V encoding for a vocabulary size V. The values of the nodes $x^{(\ell)}$ of a layer ℓ are given by the equation

$$x^{(\ell)} = f\big(W^{(\ell)} x^{(\ell-1)} + b^{(\ell)}\big)$$

for a weight matrix $W^{(\ell)}$, a bias $b^{(\ell)}$, and an activation function f. For the projection layer, most of the time f is set to the identity function. For the hidden layer, the sigmoid function is used, and for the output layer, f is set to the softmax function to obtain correctly normalized probabilities. The weight matrix between the input and the projection layer is tied for all history words.

For the comparative analysis in this work, we stick to two variants of this NNLM topology: (1) shortlist feedforward NNLMs, where the output layer is limited to the most-frequent words only, and (2) clustered feedforward NNLMs, where the full vocabulary is used at the output layer. In this

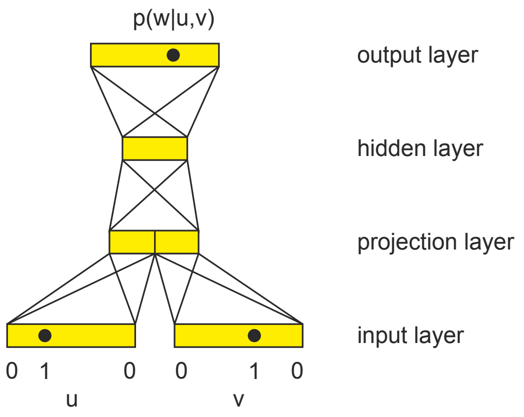

Fig. 1. Architecture of a feedforward NNLM.

case, each word is attributed to a unique class, and instead of directly predicting the probability of a word, this quantity is factorized into two individual terms:

$$p(w|u,v) = p\left(w|c(w);u,v\right) \cdot p\left(c(w)|u,v\right).$$

Here, first the class $c(w)$ of the word w is predicted, and then, given the class $c(w)$ and the predecessor words u, v, the probability of the next word is derived. As only a limited number of words occurs in a given class, and the number of word classes can be chosen to be much smaller than the vocabulary size, substantial speed-ups in training can be achieved by applying this equation, see [11] and [12].

Recurrent neural networks follow a similar topology, but as input they only receive the direct predecessor word, and the hidden layer contains recurrent connections, thereby implicitly taking into account multiple predecessor words that were presented to the network before. In this work, we use the more advanced LSTM node type for the hidden layer. Details on LSTM NNLMs can be found in [5].

3.1. Parallelization

For any kind of NNLM, the computational costs for training are considerable.

Therefore, the training computations are split between several CPU cores that independently generate intermediate results. In case of feedforward NNLMs, a fixed number of n-grams from the training data is propagated through the network without updating its weights. The number of training samples is referred to as bunch size, and, for language modeling, reasonable values may e. g. lie between 2 and 128 ([13]).

This approach is not directly transferrable to recurrent NNLMs because there is no fixed context length and, thus, the probabilities of subsequent words in a sentence are all interdependent. Nevertheless, a similar idea can be applied to the case of recurrent neural networks. Instead of parallelizing over a set of individual words from a sentence, the bunch can be defined to contain full sentences. As a result, each core calculates the gradient with respect to a single sentence while the weights are kept fixed. Afterwards, the gradients for the individual sentences are accumulated and the result is used for updating the weight parameters.

The standard approach would be for each CPU core to maintain its individual gradient vector, and to add up the gradient vectors one by one (or pair-wise in a recursive fashion, if this step shall be parallelized, too) after processing all elements from the bunch.

One of the principal problems of bunch mode parallelization lies in the interaction with the clustering speed-up technique. The basic concept of the clustering approach is that only a small portion of the weight matrix is needed for calculating probabilities. Consequently, only a small number of components of the gradient has to be computed, and it would be inefficient to consider the full gradient matrix.

We resolve this problem by keeping track of all word classes that were observed within a given training sequence. When updating the weight parameters, only those weights need to be modified that correspond to words belonging to one of the previously seen classes. The weights related to the class posterior probability $p\big(c(w)|h\big)$, given the preceding history words h, need to be updated in all cases.

Using four CPU cores, we were able to reduce the training time down to 60 % of the original single-threaded version.

3.2. Probability Normalization

In many practical applications, an NNLM cannot be trained on the full set of vocabulary words. For shortlist NNLMs, by definition only the most frequent words are covered.

Even for clustered NNLMs, usually not all vocabulary words can be predicted. The reason is that normally, the NNLM is interpolated with a backing-off model. Such a model can easily be trained on data sizes that are too large for NNLM training, especially when the NNLM architecture is more complex. This leads to the case where the backing-off model has seen relevant words in training that did not occur in the reduced subset of NNLM training data at all.

In [13] it was proposed to compute normalized NNLM probabilities $\tilde{p}_{\text{NN}}$ according to

$$\tilde{p}_{\text{NN}}(w|h) = \begin{cases} \gamma(h) p_{\text{NN}}(w|h) & w \in V_{\text{NN}} \\ p_{\text{BO}}(w|h) & \text{otherwise} \end{cases}$$

for an NNLM vocabulary V_{NN}, NNLM probabilities p_{NN}, and for backing-off probabilities p_{BO}. The normalization term $\gamma(h)$ is obtained by summing up the backing-off LM probabilities over all words from the shortlist vocabulary.

The disadvantage of this method is that the computation of the sum over all shortlist OOV words requires a lot of computation time, especially when the backing-off model is large. Furthermore, no stand-alone perplexity can be computed for the NNLM as an additional backing-off model is needed, too. Nevertheless this approach is widely used in practice. Further normalization variants suffering from at least one of these problems are listed in [14].

As an alternative, an NNLM can be trained on the vocabulary $V_{\text{NN}} \cup \{\langle\text{unk}\rangle\}$, where $\langle\text{unk}\rangle$ denotes the unknown word token. We then propose to obtain probabilities $\tilde{p}_{\text{NN}}$ for the full backing-off vocabulary $V_{\text{BO}} \supset V_{\text{NN}}$ by setting

$$\tilde{p}_{\text{NN}}(w|h) = \begin{cases} p_{\text{NN}}(w|h) & w \in V_{\text{NN}} \\ \dfrac{p_{\text{NN}}(\langle\text{unk}\rangle|\text{h})}{|V_{\text{BO}} \setminus V_{\text{NN}}| + 1} & \text{otherwise.} \end{cases}$$

In this way, extending the NNLM probabilities to the full vocabulary can be done at virtually no computational extra costs. At the same time, zero probabilities can be avoided which is also interesting for comparing perplexities between different LMs. In addition, no dependence on the backing-off model itself is introduced.

4. EXPERIMENTAL RESULTS

The experimental comparison of different NNLM architectures presented in this work is based on the five state-of-the-art French speech recognition systems of RWTH that were trained for the 2012 evaluation of the Quaero research project[1]. These obtained the best results among those systems evaluated on this task.

The amount of acoustic training data comprised 350 hours of manually transcribed broadcast news and broadcast conversational speech. The baseline backing-off LM was trained on 1.6 billion running words for a vocabulary of size 200 K using Kneser-Ney smoothing [1]. The resulting speech recognition systems included state-of-the-art techniques like speaker adaptation, cross adaptation, Multi-Layer Perceptron (MLP) Tandem features and discriminative training (or a hybrid MLP acoustic model instead). By applying Confusion Network Combination (CNC, cf. [15]), all five systems were combined to further improve the recognition result.

Throughout this paper, three different NNLM implementations were used: The LIMSI shortlist feedforward NNLM software, the RWTH clustered feedforward NNLM and the RWTH clustered recurrent LSTM NNLM implementation. For NNLM training, compared to the backing-off model only a limited amount of training data could be used which comprised a subset of 27 M running words of in-domain data. From the original 200 K vocabulary words of the backing-off LM, 180 K words were seen in the reduced NNLM data set. Based on relative frequencies, 200 classes were precomputed for the clustering of the NNLMs. The hidden layer sizes varied between 300 and 500 nodes, depending on the performance on the development data.

Perplexity results of baseline and neural network models are depicted in Table 1. We observe that in particular the LSTM achieves perplexities similar to those of the backing-off model, even though the LSTM was trained on much less data. Using a combination of two LSTMs, the backing-off model is even outperformed on the test data.

[1]http://www.quaero.org

The context-length of the FFNN was increased until no further improvement was observed, resulting in 8-gram NNLMs of perplexity 146.9 on the test set. In spite of the long context size of the FFNN, the LSTM perplexities were lower by 15 to 17 % relative compared to the feedforward architecture. Improvements by interpolating additional NNLMs are rather small when a large backing-off model has already been included for interpolation.

BO	FFNN		LSTM		Perplexity	
	LM_1	LM_2	LM_1	LM_2	dev12	test12
●	·	·	·	·	102.9	122.0
·	●	·	·	·	137.3	146.9
·	●	●	·	·	130.8	139.4
●	●	·	·	·	92.8	106.9
●	●	●	·	·	91.9	105.5
·	·	·	●	·	113.8	124.6
·	·	·	●	●	106.2	116.3
●	·	·	●	·	84.4	97.4
●	·	·	●	●	82.8	95.2

Table 1. Perplexity results of the RWTH NNLMs for the Quaero French 2012 corpora.

Table 2 shows WER results for the same task. All NNLMs were interpolated with the baseline backing-off model. On 100-best lists, an 8-gram FFNN was used, while on lattices only a 4-gram FFNN was applied due to lattice expansion which becomes difficult for long context-sizes. (On the test data, the clustered 4-gram NNLM achieves a perplexity which is larger by 4 points compared to the 8-gram from Table 1.)

We find that huge gains of 1.4 % absolute on the development and 1.3 % absolute on the test data can be obtained with LSTMs on top of our single-best baseline system with a backing-off LM and Viterbi decoding. It should be noted that the LSTM results may be slightly improved by a more accurate rescoring procedure. During rescoring, the LSTM history is not reset when shifting from one audio recording to another. Besides, the context-length is limited to eight consecutive sentences. The rescoring could be carried out such that a sliding window of eight sentences is moved over the development or test data. Instead, we use non-overlapping windows of eight sentences to simplify the rescoring process. On the other hand, preliminary experiments showed that, by taking both aspects into account, the WER can just be improved by 0.1 % absolute.

We see that none of the feedforward NNLMs performs as well as the recurrent networks. On the test data, LSTM networks improve by 0.4 % absolute over feedforward NNLMs, even after system combination. Interestingly, all FFNNs perform similarly after system combination: When rescoring on 100-best lists with longer context sizes, the gains after Viterbi decoding are larger compared to the lattice version, but the system combination on lattices gives larger improvements than on 100-best lists, so both approaches end up the same. Similar observations can be made when using Confusion Network Decoding for a single system.

LM Type	Hypoth.	Decoding	WER	
			dev12	test12
Backing-off	lattices	Viterbi 1x	15.6 %	17.4 %
		CNC 5x	14.5 %	16.7 %
+1x short-list FFNN	lattices	Viterbi 1x	15.1 %	17.1 %
		CNC 5x	14.2 %	16.2 %
+1x FFNN	100-best	Viterbi 1x	14.6 %	16.8 %
		CNC 5x	14.2 %	16.4 %
	lattices	Viterbi 1x	15.0 %	16.9 %
		CNC 5x	14.1 %	16.2 %
+2x FFNN	100-best	Viterbi 1x	14.6 %	16.7 %
		CNC 5x	14.1 %	16.3 %
	lattices	Viterbi 1x	14.9 %	16.9 %
		CNC 5x	14.1 %	16.2 %
+1x LSTM	100-best	Viterbi 1x	14.4 %	16.3 %
		CNC 5x	13.9 %	16.0 %
+2x LSTM	100-best	Viterbi 1x	14.2 %	16.1 %
		CNC 5x	13.8 %	15.8 %

Table 2. WER results on the Quaero French 2012 development and test corpora.

5. CONCLUSION

In this work, we compared feedforward and recurrent NNLMs on a French speech recognition task. We found that recurrent neural networks outperformed standard feedforward approaches on these data.

From the results obtained we also conclude that standard speech recognition technology somehow limits the possible improvements that can be obtained from applying recurrent NNLMs: Confusion Network based techniques give additional gains in WER, but favor lattices over m-best lists which themselves are the basis of any recurrent NNLM rescoring approach. Therefore it may be worthwhile to further investigate how to improve decoding on m-best lists (or conversely, converting m-best lists back to a lattice-like format).

Furthermore, for future work, it will be interesting to combine feedforward and recurrent NNLM architectures to see to which extent their improvements over baseline models are additive in WER.

6. ACKNOWLEDGEMENT

This work was partly realized as part of the Quaero program, funded by OSEO, French State agency for innovation. Furthermore, the research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreements nos. 287658 and 287755. H. Ney was partially supported by a senior chair award from DIGITEO, a French research cluster in Ile-de-France.

7. REFERENCES

[1] Kneser, R., and Ney, H., “Improved Backing-Off For M-Gram Language Modeling”, Proc. of ICASSP 1995, pp. 181–184

[2] Bengio, Y., and Ducharme, R., “A neural probabilistic language model”, Proc. of Advances in Neural Information Processing Systems (2001), vol. 13., pp. 932–938

[3] Schwenk, H., Gauvain, J.-L., “Connectionist Language Modeling for Large Vocabulary Continuous Speech Recognition”, Proc. of ICASSP 2002, pp. 765–768

[4] Mikolov, T., Karafiát, M., Burget, L., Černocký, J., and Khudanpur, S., “Recurrent neural network based language model”, Proc. of Interspeech 2010, pp. 1045–1048

[5] Sundermeyer, M., Schlüter, R., and Ney, H., “LSTM Neural Networks for Language Modeling”, Proc. of Interspeech 2012

[6] Mikolov, T., Kombrink, S., Burget, L., Černocký, J., and Khudanpur, S., “Extensions of Recurrent Neural Network Language Model”, Proc. of ICASSP 2011, pp. 5528–5531

[7] Arısoy, E., Sainath, T. N., Kingsbury, B., and Ramabhadran, B., “Deep Neural Network Language Models”, Proc. of NAACL-HLT 2012 Workshop, pp. 20–28

[8] Le, H. S., Oparin, I., Messaoudi, A., Allauzen, A., Gauvain, J.-L., and Yvon, F. “Large Vocabulary SOUL Neural Network Language Models”, Proc. of Interspeech 2011, pp. 1469–1472

[9] Le, H. S., Allauzen, A., and Yvon, F., “Measuring the Influence of Long Range Dependencies with Neural Network Language Models”, Proc. of NAACL-HLT 2012 Workshop, pp. 1–10

[10] Goodman, J. T., “A bit of progress in language modeling”, Computer Speech and Language 15 (2001), pp. 403–434

[11] Goodman, J. T., “Classes for fast maximum entropy training”, Proc. of the ICASSP 2001, pp. 561–564

[12] Morin, F., Bengio, Y., “Hierarchical Probabilistic Neural Network Language Model”, Proc. of the 10th Int. Workshop on Artificial Intelligence and Statistics 2005

[13] Schwenk, H., “Continuous Space Language Models”, Computer Speech and Language 21 (2007), pp. 492–518

[14] Park, J., Liu, X., Gales, M. J. F., and Woodland, P. C., “Improved Neural Network Based Language Modelling and Adaptation”, Proc. of Interspeech 2010, pp. 1041–1044

[15] Hoffmeister, B., Schlüter, R., and Ney, H., “iCNC and iROVER: The Limits of Improving System Combination with Classification?”, Proc. of Interspeech 2008, pp. 232–235