Neural Network Language Models for Conversational Speech Recognition

Holger Schwenk and Jean-Luc Gauvain

LIMSI - CNRS, bat 508, BP 133, 91403 Orsay cedex, FRANCE
{schwenk,gauvain}@limsi.fr

Abstract

Recently there is growing interest in using neural networks for language modeling. In contrast to the well known backoff n-gram language models (LM), the neural network approach tries to limit problems from the data sparseness by performing the estimation in a continuous space, allowing by these means smooth interpolations. Therefore this type of LM is interesting for tasks for which only a very limited amount of in-domain training data is available, such as the modeling of conversational speech.

In this paper we analyze the generalization behavior of the neural network LM for in-domain training corpora varying from 7M to over 21M words. In all cases, significant word error reductions were observed compared to a carefully tuned 4-gram backoff language model in a state of the art conversational speech recognizer for the NIST rich transcription evaluations. We also apply ensemble learning methods and discuss their connections with LM interpolation.

1. Introduction

Language modeling of conversational speech is known to be a challenging task due to the unconstrained speaking style, frequent grammatical errors, hesitations, start-overs, etc. In addition, language modeling for conversational speech suffers from a lack of adequate training data since the main source is audio transcriptions, in contrast to the broadcast news (BN) task for which other news sources are readily available. Unfortunately, collecting large amounts of conversational data and producing detailed transcriptions is very costly. One possibility is to increase the amount of training data by selecting conversational-like sentences in BN material and on the Internet, or by transforming other sources to be more conversational-like.

Recently, a new approach has been developed that proposes to carry out the estimation task in a *continuous space* [1, 2], with the goal of making better use of the limited amount of training material. The basic idea is to project the word indices onto a continuous space and to use a probability estimator operating on this space. Since the resulting probability functions are smooth functions of the word representation, better generalization to unknown n-grams can be expected. A neural network can be used to simultaneously learn the projection of the words onto the continuous space and the n-gram probability estimation.

The first evaluation of such an approach in a conversational speech recognizer has shown that it can be used to reduce the word error [3]. These results were later confirmed with an improved speech recognizer for the DARPA Switchboard (SWB) task [4]. A neural network LM has also been used with a Syntactical Language Model showing perplexity and word error improvements on the Wallstreet Journal corpus [5]. In our previous work the neural network LM was trained on about 6.1M words and the word error rates were in the range of 25%. Since then, large amounts of conversational data have been collected in the DARPA EARS program and quick transcriptions were made available (total of about 1800h, 21.7M words), helping to build better speech recognizers. This amount of LM training data enables better training of the backoff 4-gram LM and one may wonder if the neural network LM, a method developed for probability estimation of sparse data, still achieves an improvement. In this paper we show that the neural network LM continues to achieve consistent word error reductions with respect to a carefully tuned backoff 4-gram LM. Detailed results are provided as a function of the size of the LM training data. We also compare the behavior on the training and test data. Finally we investigate training multpile neural networks in order to achieve better convergence.

2. Architecture of the Neural Network LM

The architecture of the neural network n-gram LM is shown in Figure 1. A standard fully-connected multi-layer perceptron is used. The inputs to the neural network are the indices of the $n-1$ previous words in the vocabulary $h_j = w_{j-n+1}, ..., w_{j-2}, w_{j-1}$ and the outputs are the posterior probabilities of *all* words of the vocabulary:

$$P(w_j = i|h_j) \qquad \forall i \in [1, N] \tag{1}$$

where N is the size of the vocabulary. This can be contrasted to standard language modeling where each n-gram probability is calculated independently. The input uses the so-called 1-of-n coding, i.e., the *i-th* word of the vocabulary is coded by setting the *i-th* element of the vector to 1 and all the other elements to 0. This coding substantially simplifies the calculation of the projection layer since only the *i-th* line needs to be copied of the $N \times P$ dimensional projection matrix, where N is the size of the vocabulary and P the size of the projection.

Let us denote c_k these projections, d_j the hidden layer activities, o_i the outputs, p_i their softmax normalization, and m_{jk}, b_j, v_{ij} and k_i the hidden and output layer weights and the corresponding biases. Using matrix/vector notation the neural network performs the following operations:

$$\mathbf{d} = \tanh\left(\mathbf{M} * \mathbf{c} + \mathbf{b}\right) \tag{2}$$

$$\mathbf{o} = \tanh\left(\mathbf{V} * \mathbf{d} + \mathbf{k}\right) \tag{3}$$

$$\mathbf{p} = \exp(\mathbf{o}) \ / \ \sum_{k=1}^{N} e^{o_k} \tag{4}$$

where lower case bold letters denote vectors and upper case bold letters denote matrices. The tanh and exp function as well as the division are performed element wise. The value of the output neuron p_i corresponds directly to the probability

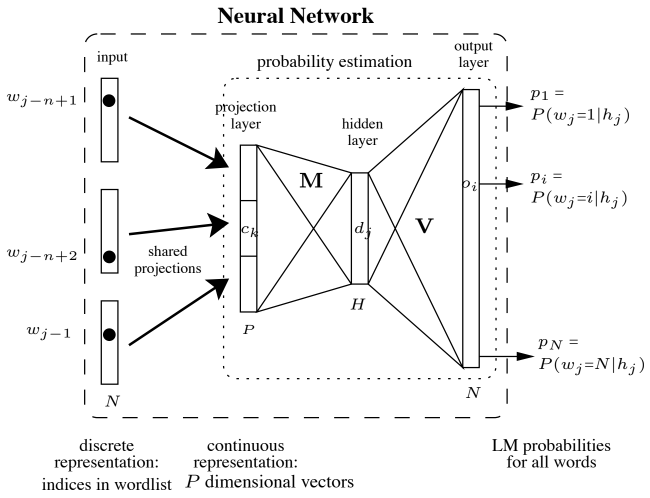

Figure 1: Architecture of the neural network language model. h_j denotes the context $w_{j-n+1}, ..., w_{j-1}$. P is the size of one projection and H and N is the size of the hidden and output layer respectively. When shortlists are used the size of the output layer is much smaller then the size of the vocabulary.

$P(w_j = i|h_j)$. Training is performed with the standard back-propagation algorithm using the cross-entropy as error function, and a weight decay regularization term. The targets are set to 1.0 for the next word in the training sentence and to 0.0 for all the other ones. It can be shown that the outputs of a neural network trained in this manner converge to the posterior probabilities. Therefore, the neural network directly minimizes the perplexity on the training data. Note also that the gradient is back-propagated through the projection-layer, which means that the neural network learns the projection of the words onto the continuous space that is best for the probability estimation task.

The complexity of this basic architecture during training and recognition is quite high, in particular due to the large size of the output layer. Therefore, we chose to limit the output of the neural network to the most frequent words, referred to as a *shortlist* in the following discussion. All words in the word list are still considered at the input of the neural network. In all the experiments reported in this paper a shortlist length of 2000 was used, which covers about 89% of the LM probability requests during lattice rescoring. The LM probabilities of the words in the shortlist ($\hat{P}_N$) are calculated by the neural network and the LM probabilities of the remaining words (P_B) are obtained from a standard 4-gram backoff LM:

$$P(w_j|h_j) = \begin{cases} \hat{P}_N(w_j|h_j)\, P_S(h_j) & \text{if } w_j \in \text{shortlist} \\ P_B(w_j|h_j) & \text{else} \end{cases} \quad (5)$$

$$\text{with } P_S(h_j) = \sum_{w \in shortlist(h_j)} P_B(w|h_j) \quad (6)$$

In other words, one can say that the neural network redistributes the probability mass of all the words in the shortlist.[1] This probability mass is precalculated and stored in the data structures of the standard 4-gram LM. A backoff technique is used if the probability mass for a requested input context is not directly available.

[1]Note that the sum of the probabilities of the words in the shortlist for a given context is normalized $\sum_{w \in shortlist} \hat{P}_N(w|h_j) = 1$.

In our current implementation one training epoch through 21M words takes about 7h on a PC with a 2.8GHz Intel Xeon processor running under Linux. During recognition, the neural network language model is used to rescore word lattices after acoustic MLLR adaptation which typically takes less than 0.1xRT. Details of these algorithms are given in [6].

3. Baseline Speech Recognizer

In this paper the neural network language model is evaluated in a state of the art speech recognizer for conversational telephone speech (CTS)[4]. The word recognizer uses continuous density HMMs with Gaussian mixture for acoustic modeling and n-gram statistics estimated on large text corpora for language modeling. Each context-dependent phone model is a tied-state left-to-right CD-HMM with Gaussian mixture observation densities where the tied states are obtained by means of a decision tree. The acoustic feature vector has 39-components including 12 cepstrum coefficients and the log energy, along with their first and second derivatives. Cepstral mean and variance normalization are carried out on each conversation side. The acoustic models are MMI trained on 860 hours of data.

Decoding is carried out in 3 passes. In the first pass the speaker gender for each conversation side is identified using Gaussian mixture models, and a fast trigram decode is performed to generate approximate transcriptions. These transcriptions are used to compute the VTLN warp factors for each conversation side and to adapt the SAT models that are used in the second pass. Passes 2 and 3 make use of the VTLN-warped data to generate a trigram lattice per speaker turn which is expanded with the 4-gram baseline backoff LM and converted into a confusion network with posterior probabilities. The best hypothesis in the confusion network is used in the next decoding pass for unsupervised MLLR adaptation of the acoustic models (constraint and unconstrained). The third pass is similar to the second one but more phonemic regression classes are used and the search space is limited to the word graph obtained in the second pass. The overall run time is about 19xRT.

4. Language model training

The main source for training a LM for conversational speech are the transcriptions of the audio training corpora. Three different *in-domain* data corpora have been used:

7.2M words Our first experiments were carried out with the initial release of transcriptions of acoustic training data for the SWB task, namely the *careful* transcriptions of the SWB corpus distributed by LDC (2.7M words) and by ISIP (2.9M words), the Callhome corpus (217k words), some SWB cellular data (230k words) and *fast* transcriptions of a previously unused part of the SWB2 corpus (80h, 1.1M word).

12.3M words In 2003 LDC changed the data collection paradigm of conversational data to the so called "Fisher-protocol". Fast transcriptions of 520h of such data were available for this work.

21.7M words In a second release, fast transcriptions of additional 1300h have been made available, almost doubling the total amount of language model training data.

In addition to these in-domain corpora the following texts have been used to train the 4-gram backoff LM:

- 240M words of commercially produced BN transcripts,
- 80M words of CNN television BN transcriptions,
- 180M words of conversational like data that was collected from the Internet.[2]

We refer to these 500M words as BN corpus. Adding other sources, in particular newspaper text, did not turn out to be useful. The LM vocabulary contains 51077 words. The baseline LM is constructed as follows: Separate backoff n-gram LMs are estimated on all the above corpora using the modified version of Kneser-Ney smoothing as implemented in the SRI LM toolkit [7]. A single backoff LM was built by merging these models, estimating the interpolation coefficients with an EM procedure. Table 1 gives some statistics about the reference LMs.

in-domain data [words] (+500M words BN data)	7.2M	12.3M	21.7M
number of 2-grams	20.1M	20.1M	20.2M
3-grams	38.8M	31.5M	33.0M
4-grams	24.0M	24.3M	28.4M
Perplexity on Eval03 test	53.0	51.5	49.8

Table 1: Statistics for the backoff 4-gram reference LM built using in-domain training data of varying sizes and 500M words of BN data.

5. Experimental results

The neural network LM was trained only on the in-domain corpora. Two experiments have been conducted:

1. The neural network LM is interpolated with a backoff LM that was trained only on the in-domain copora and compared to this LM,
2. The neural network LM is interpolated with the full backoff LM (in-domain and BN data) and compared to this full LM.

The first experiment allows us to assess the real benefit of the neural LM since the two smoothing approaches (backoff and neural network) are compared on the same data. In the second experiment all the available data is used to obtain the overall best results. The perplexities of the neural network and the backoff LM are given in Table 2.

in-domain data [words]:	7.2M	12.3M	21.7M
In-domain data only:			
backoff LM	62.4	55.9	53.3
neural LM	57.0	50.6	48.5
Interpolated with all data:			
backoff LM	53.0	51.1	49.8
neural LM	50.8	48.0	46.6

Table 2: Eval03 test set perplexities for the backoff and neural LM as a function of the size of the in-domain training data.

A perplexity reduction of about 9% relative is obtained independently of the size of the LM training data. This gain decreases to approximatively 6% after interpolation with the backoff LM trained on the additional 500M word of out-of domain data. It can been seen that the perplexity of the neural network LM trained only on the in-domain data is better than that of the backoff reference LM trained on all data (48.5 with respect to 49.8). Despite these rather small gains in perplexity, consistent word error reductions were observed (see Figure 2). The first system is that described in [4]. The second system has a much lower word error rate than the first one due to several improvements of the acoustics models, and the availability of more acoustic training data. The third system differs from the second one only by the amount of LM training data, the acoustic models have not been modified. The small decrease in word error in this case suggests that more LM training data would be of little help.[3]

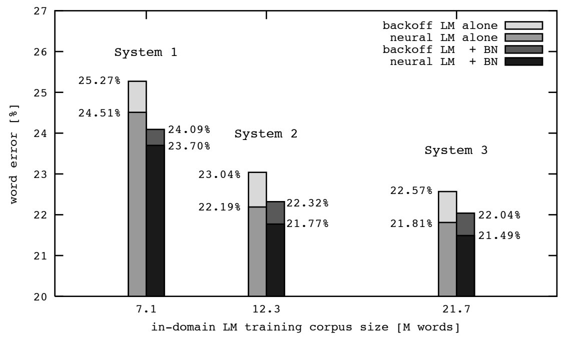

Figure 2: Word error rates on the Eval03 test set for the backoff LM and the neural network LM, trained only on in-domain data (left bars for each system) and interpolated with 500M word BN LM (right bars for each system).

Although the size of the LM training data has almost tripled from 7.2M to 21.7M words, a consistent absolute word error reduction of 0.55% can be observed. In all these experiments, it seems that the word error reductions brought by the neural network LM are independent of the other improvements, in particular those obtained by better acoustic modeling and by adding more language model training data. When only in-domain data is used (left bars for each system in Figure 2) the neural network LM achieves an absolute word error reduction of about 0.8%. Note also that the neural network LM trained on at least 12.3M words is better than the backoff LM that uses in addition 500M words of the BN corpus. In a contrastive experiment, backoff and neural LMs were trained on a random 400k word subset of the 21.7M corpus, simulating for instance a domain specific task with very little LM training data. In this case, the neural network LM decreased the perplexity from 84.9 to 75.6 and the word error rate from 25.60% to 24.67%.

In order to get more insight in the generalization behavior of the neural network LM the word error on the training corpus has been calculated. This has been done by rescoring 2-gram lattices that were generated for MMI acoustic model training (see Table 3, 12.3M words). Words that are in the training data, but not in the recognition lexicon were replaced by silence. It is not surprising to see that the backoff LMs have a very low

	backoff LM		neural LM
	3-gram	4-gram	4-gram
perplexity	33.3	17.7	41.9
word error	23.6%	18.6%	20.1%

Table 3: Performance on the training data for both LM.

[2]This data has been provided by the University of Washington.

[3]Going from 7.2M to 12.3M words of LM data gave a word error reduction of about 0.5% without changing the acoustic models.

perplexity on the training data, since these are basically "memory based" models that store the complete training data.[4] The 4-gram neural network LM, which is a more a compact representation of the data, has a much higher training perplexity than the 4-gram and even the 3-gram backoff LM. The perplexity is almost comparable to that obtained on the test data, which may be seen as an indicator for good generalization behavior.

6. Interpolation and ensembles

When building backoff LM for a collection of training corpora, better results are usually obtained by first building separate LMs for each corpus and then merging them together using interpolation coefficients obtained by optimizing the perplexity on some development data [8]. This procedure has been used for all our backoff LMs. It is, however, not so straightforward to apply this technique to a neural network LM since individually trained LMs can not be merged together, and they need to be interpolated on the fly. In addition, it may be sub-optimal to train the neural network LMs on subparts of the training corpus since the continuous word representation is not learned on all data. For this reason, in the above described experiments only one neural network was trained on all data.

Training large neural networks (2M parameters) on a lot of data (21.4M examples) with stochastic backpropagation can be quite slow and poses convergence problems. We have developed very fast algorithms (see [6] for details), but we believe that the neural network underfit the training data. Unfortunately, more sophisticated techniques than stochastic gradient descent are not tractable for problems of this size. On the other hand, several well known ensemble learning algorithms used in the machine learning community can be applied to neural networks, in particular Bagging and AdaBoost. Following to the bias/variance decomposition of error, Bagging improves performance by minimizing the variance of the classifiers [9], while AdaBoost sequentially constructs classifiers that try to correct the errors of the previous ones [10]. In this work we have evaluated another variance reduction method: several networks are trained on all the data, but after each epoch the training data is randomly shuffled.

#hidden units	1024	1280	1560	1600	2000
One network	22.19	22.22	22.18	-	-
Ensembles:	2x500 22.15	3x400 22.12	3x500 22.07	4x400 22.09	4x500 22.09

Table 4: Word error rates for one large neural network and ensembles, trained on the 12.4M word in-domain corpus only.

As can been seen in Table 4 increasing the number of parameters of one large neural network does not lead to improvements for more than 1000 hidden units, but using ensembles of several neural networks with the same total number of parameters results in a slight decrease in the word error rate.

7. Conclusions

This paper has extended our recent work on language modeling using neural networks. The discrete word indices are projected onto a continuous space, allowing by these means "smooth interpolations" of the LM probabilities. The neural network LM has been evaluated using in-domain corpora of 7.2M to 21.4M words of acoustic transcriptions. When compared to a carefully tuned backoff 4-gram LM trained on the same data, the new approach achieves consistent word error reductions of about 0.8% in a state-of-the-art conversational speech recognizer on the NIST RT03 evaluation data. Even after adding 500M words of out-domain broadcast news data to the backoff LM only, the neural network LM still achieves better results. Our best system with the neural network LM has a word error rate of 21.5% on the Eval03 test set while the system with the backoff reference LM only achieves 22.0%. Both systems run in 19xRT.

The neural network LM learns a distributed representation of the LM probability distributions, achieving by these means good generalization to unseen n-grams. Backoff LMs are memory based models and have a tendency to overfit the training data. Thus much lower perplexities and word error rates on the training data were observed, but the results are worse on independent test data. Future work will concentrate on (word) error corrective training algorithms like AdaBoost and training criteria that directly minimize the estimated word error when the LM is used in a speech recognizer.

[4] If no cut-offs are used.

8. Acknowledgment

The authors would like to recognize the contributions of L. Lamel, G. Adda, and F. Lefèvre for their involvement in the development of the LIMSI conversational speech recognition system on top of which this work is based.

9. References

[1] Y. Bengio and R. Ducharme, "A neural probabilistic language model," in *Advances in Neural Information Processing Systems*, vol. 13. Morgan Kaufmann, 2001.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. 2, pp. 1137–1155, 2003.

[3] H. Schwenk and J.-L. Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," in *International Conference on Acoustics, Speech, and Signal Processing*, 2002, pp. I: 765–768.

[4] J.-L. Gauvain, L. Lamel, H. Schwenk, G. Adda, L. Chen, and F. Lefèvre, "Conversational telephone speech recognition," in *International Conference on Acoustics, Speech, and Signal Processing*, 2003, pp. I:212–215.

[5] A. Emami, P. Xu, and F. Jelinek, "Using a connectionist model in a syntactical based language model," in *International Conference on Acoustics, Speech, and Signal Processing*, 2003, pp. I:272–375.

[6] H. Schwenk, "Efficient training of large neural networks for language modeling," in *IEEE joint conference on neural networks*, 2004.

[7] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *International Conference on Speech and Language Processing*, 2002, pp. II: 901–904.

[8] F. Jelinek and R. Mercer, "Interpolated estimation of markov source parameters from sparse data," *Pattern Recognition in Practice*, pp. 381–397, 2000.

[9] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1994.

[10] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and Computation*, vol. 121, no. 2, pp. 256–285, 1995.