USING NEURAL NETWORK LANGUAGE MODELS FOR LVCSR

Holger Schwenk and Jean-Luc Gauvain

LIMSI-CNRS
BP 133, bat 508, 91436 Orsay cedex, FRANCE

ABSTRACT

In this paper we describe how we used a neural network language model for the BN and CTS task for the RT04 evaluation. This new language modeling approach performs the estimation of the n-gram probabilities in a continuous space, allowing by this means probability interpolation for any context, without the need to back-off to shorter contexts. Details are given on training data selection, parameter estimation and fast training and decoding algorithms. The neural network language model achieved word error reductions of 0.5% for the CTS task and of 0.3% for the BN task with an additional decoding cost of 0.05xRT.

1. INTRODUCTION

There has been a lot of progress in acoustic modeling for large vocabulary speech recognition (LVCSR) during the last years, in particular various techniques for discriminative training and linear transformations for adaptation. These techniques usually achieve word error reductions of several points when deployed in a large vocabulary speech recognizer. It seems more complicated to propose alternative approaches to the widely adopted n-gram back-off language models (LM). These models are quite difficult to beat in a LVCSR, in particular when there are several million words of language model training data available. The only new approach we are aware of that achieves a word error reduction, is the SuperARV language model used by SRI in their BN and CTS systems [1, 2].

In this paper we describe the application of a new approach that uses a neural network to estimate the LM n-gram probabilities [3, 4]. 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 densities are continuous 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 to estimate the n-gram probabilities. This is still a n-gram approach, but the LM n-gram probabilities are "interpolated" for any possible context of length n-1 instead of backing-off to shorter contexts.

The first evaluation of such an approach in a conversational speech recognizer demonstrated that it can effectively reduce the word error rate [5]. In the following sections we show how we used the neural network language LM to reduce the word error rates for both CTS and BN RT04 tasks while keeping the decoding time about the same.

2. ARCHITECTURE

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 $h_j = w_{j-n+1}, ..., w_{j-2}, w_{j-1}$ and the outputs are the n-gram probabilities of *all* words of the vocabulary:

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

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:

$$\mathbf{d} = \tanh(\mathbf{M} * \mathbf{c} + \mathbf{b}) \qquad (1)$$

$$\mathbf{o} = \mathbf{V} * \mathbf{d} + \mathbf{k} \qquad (2)$$

$$\mathbf{p} = \exp(\mathbf{o}) \ / \ \sum_{k=1}^{N} e^{o_k} \qquad (3)$$

where lower case bold letters denote vectors and upper case bold letters denote matrices. The tanh and exp functions as well as the division are performed element wise. The value

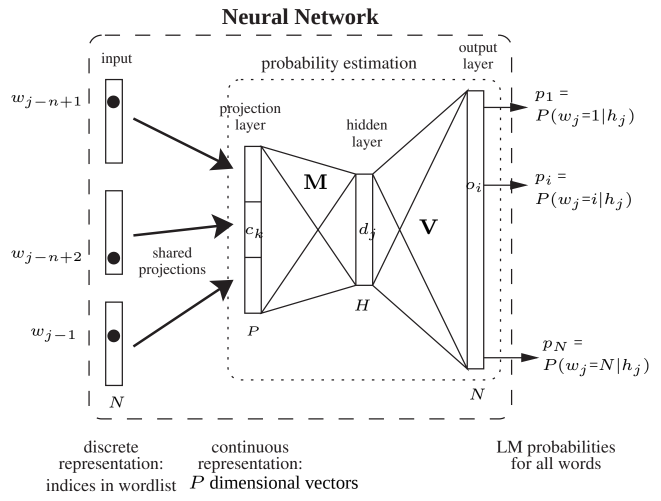

Fig. 1. Architecture of the neural network language model. h_j denotes the context $w_{j-n+1}, ..., w_{j-1}$. P is the size of one projection and H and N is the size of the hidden and output layer respectively. When shortlists are used the size of the output layer is much smaller then the size of the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

of the output neuron p_i corresponds directly to the probability $P(w_j = i|h_j)$. 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.

2.1. Fast Recognition

It is known that language models play an important role in decoding continuous speech. Using the neural 4-gram LM directly during decoding imposes an important burden on search space organization since a context of three words must be kept. This led to long decoding times in our first experiments when the neural LM was used directly during decoding [5]. In order to make the model tractable for LVCSR the following techniques have been applied:

1. *Lattice rescoring*: decoding is done with a standard back-off LM and a lattice is generated. The neural network LM is then used to rescore the lattice.

2. *Shortlists*: the neural network is only used to predict the LM probabilities of a subset of the whole vocabulary.

3. *Regrouping*: all LM probability requests in one lattice are collected and sorted. By these means all LM probability requests with the same context h_t lead to only one forward pass through the neural network.

4. *Block mode*: several examples are propagated at once through the neural network, allowing the use of faster matrix/matrix operations.

5. *CPU optimization*: machine specific BLAS libraries are used for fast matrix and vector operations.

It can be easily seen that most of the calculations are due to the large size of the output layer of the neural network. Remember that all outputs need to be calculated in order to perform the softmax normalization even though only one LM probability is needed. Experiments using lattice rescoring with unnormalized LM scores led to much higher word error rates. One may argue that it is not very reasonable to spend a lot of time to get the LM probabilities of words that do not appear very often. Therefore, we chose to limit the output of the neural network to the s most frequent words, $s \ll |N|$, referred to as a *shortlist* in the following discussion. All words of the word list are still considered at the input of the neural network. The LM probabilities of words in the shortlist ($\hat{P}_N$) are calculated by the neural network and the LM probabilities of the remaining words ($\hat{P}_B$) are obtained from a standard 4-gram back-off LM:

$$\hat{P}(w_t|h_t) = \begin{cases} \hat{P}_N(w_t|h_t) \cdot P_S(h_t) & \text{if } w_t \in \text{shortlist} \\ \hat{P}_B(w_t|h_t) & \text{else} \end{cases} \quad (4)$$

$$P_S(h_t) = \sum_{w \in shortlist(h_t)} \hat{P}_B(w|h_t) \quad (5)$$

It can be considered 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 back-off technique is used if the probability mass for a requested input context is not directly available. Table 1 gives the coverage, i.e. the percentage of LM probabilities that are effectively calculated by the neural network when evaluating the perplexity on a development set of 56k words or when rescoring lattices.

During lattice rescoring the LM probabilities with the same context h_t are often requested several times on potentially different nodes in the lattice. Collecting and regrouping all these calls prevents unnecessary forward passes since

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

shortlist size	1024	2000	4096	8192
dev. set	89.3%	93.6%	96.8%	98.5%
lattice	88.5%	89.9%	90.4%	91.0%

Table 1. Coverage for different shortlist sizes, i.e. percentage of 4-grams that are actually calculated by the neural LM. The vocabulary size is about 51k.

all LM predictions for the same context are immediately available. Further improvements can be obtained by propagating several examples at once though the network, also know as bunch mode [6]. In comparison to equation 1 and 2, this results in using matrix/matrix instead of matrix/vector operations:

$$\mathbf{D} = \tanh(\mathbf{M} * \mathbf{C} + \mathbf{B}) \quad (6)$$

$$\mathbf{O} = \mathbf{V} * \mathbf{D} + \mathbf{K} \quad (7)$$

where $\mathbf{B}$ and $\mathbf{K}$ are obtained by duplicating the bias $\mathbf{b}$ and $\mathbf{k}$ respectively for each line of the matrix. The tanh-function is performed element wise.

These matrix/matrix operations can be aggressively optimized on current CPU architectures, e.g. using SSE2 instructions for Intel processors [7, 8]. Although the number of floating point operations to be performed is strictly identical to single example mode, an up to five times faster execution can be observed in function of the sizes of the matrices.

The NIST Eval01 test set consists of 6h of speech with 5895 conversations sides. The lattices generated by the speech recognizer for this test set contain on average 511 nodes and 1481 arcs per conversation side. In total 3.8 million 4-gram LM probabilities are requested out of which 3.4 million (89.9%) are processed by the neural network, i.e. when the word is among the 2000 most frequent words. After collecting and regrouping all LM calls in each lattice, only 1 million forward passes though the neural network have been performed, giving a cache hit rate of about 70%. Using a bunch size of 128 examples, the total processing time takes less than 9 minutes on a Intel Xeon 2.8GHz processor, i.e. 0.03 times real time. This corresponds to about 1.7 billion floating point operations per second (1.7 GFlops). Lattice rescoring without bunch mode and regrouping of the calls in one lattice is about ten times slower.

2.2. Fast Training

Language models are usually trained on text corpora of several hundred million words. With a vocabulary size of 51k words, standard back-propagation training would take several weeks. Parallel implementations and resampling techniques [4, 9] that result in important speedups have been proposed. Parallel stochastic back-propagation of neural networks needs low latency links between the processors which are very costly. Optimized floating point operations are much more efficient if they are applied to data that is stored in continuous locations in memory, making a better use of cache and data prefetch capabilities of the processors. This is not the case for resampling techniques. Therefore, a fixed size output layer was used and the words in the shortlist were rearranged in order to occupy continuous locations in memory.

In our initial implementation we used standard stochastic back-propagation and double precision for the floating point operations in order to ensure good convergence. Despite careful coding and optimized BLAS libraries for the matrix/vector operations [7, 8], one epoch through a training corpus of 12.4M examples takes about 47 hours on a Pentium Xeon 2.8 GHz processor. This processing time was reduced by more than a factor of 30 using the following techniques:

- Floating point precision (1.5 times faster). Only a slight decrease in performance was observed due to the lower precision.
- Suppression of intermediate calculations when updating the weights (1.3 times faster).
- Bunch mode: forward and back-propagation of several examples at once (up to 10 times faster).
- Multi-processing: use of SMP-capable BLAS libraries for off-the-shelf bi-processor machines (1.5 times faster).

Most of the improvement was obtained by using bunch mode in the forward and backward pass. After calculating the derivatives of the error function $\Delta\mathbf{K}$ at the output layer, the following equations were used (similar to [6]):

$$\mathbf{k} = \mathbf{k} - \lambda\Delta\mathbf{K} * \mathbf{i} \quad (8)$$

$$\Delta\mathbf{B} = \mathbf{V}^T * \Delta\mathbf{K} \quad (9)$$

$$\mathbf{V} = -\lambda\Delta\mathbf{K} * \mathbf{D}^T + \alpha\mathbf{V} \quad (10)$$

$$\Delta\mathbf{B} = \Delta\mathbf{B}. * (1 - \mathbf{D}. * \mathbf{D}) \quad (11)$$

$$\mathbf{b} = \mathbf{b} - \lambda\Delta\mathbf{B} * \mathbf{i} \quad (12)$$

$$\Delta\mathbf{C} = \mathbf{M}^T * \Delta\mathbf{B} \quad (13)$$

$$\mathbf{M} = -\lambda\Delta\mathbf{B} * \mathbf{C}^T + \alpha\mathbf{M} \quad (14)$$

where $\mathbf{i} = (1, 1, ...1)^T$, with the dimension of the bunch size. The back-propagation and weight update step, including weight decay, is done in one operation using the GEMM function of the BLAS library (eqn. 10 and 14). For this, the weight decay factor ϵ is incorporated into $\alpha = 1 - \lambda\epsilon$. The update step of the projection matrix is not shown for clarity.

Table 2 summarizes the effect of the different techniques to speed up the training. Extensive experiments were first done with a training corpus of 1.1M words and then applied

size of training data	double prec.	float prec.	bunch mode 2	4	8	16	32	128	SMP 128
1.1M words	2h	1h16	37m	31m	24m	14m	11m	8m18	5m50
12.4M words	47h	30h	10h12	8h18	6h51	4h01	2h51	2h09	1h27

Table 2. Training times reflecting the different improvements (on a Intel Pentium CPU at 2.8 GHz).

to a larger CTS corpus of 12.4M words. Bilmes et al. reported that the number of epochs needed to achieve the same MSE increases with the bunch size [6]. In our experiments the convergence behavior also changed with the bunch size, but after adapting the learning parameters only small losses in perplexity were observed, and there was no impact on the word error rate when the neural LM was used in lattice rescoring.

3. APPLICATION TO CTS

The neural network LM has already been used in LIMSI's CTS system for the RT02 and RT03 evaluations, achieving word error reduction of about 0.4%. Since then, the word error rates of the overall system have been significantly reduced, mainly due to the effective use of large amounts of data for acoustic and language modeling. In the following sections contrastive results are provided, showing the impact of the neural network LM in function of the available training data.

3.1. LM training data

The following corpora were used for language modeling in the RT04 system:

- CTS transcripts with breath noise (6.1M words): 2.7M words of the SWB transcriptions from LDC, 1.1M words of CTRAN transcriptions of SWB data, 230k words of cellular training data, 215k word of the Call-Home corpus transcriptions, 1.7M words of Fisher data transcribed by LDC and transcripts of the 1997 to 2001 test sets.
- CTS transcripts without breath noise (21.2M words): 2.9M words of SWB transcriptions from ISIP, 18.3M words of Fisher data transcribed by WordWave and distributed by BBN.
- BN transcriptions from LDC (years 92-95) and from PSMedia (years 96 and 97, and Jan-Nov 1998): 260.3M words.
- CNN transcripts from the CNN archive (01/2000-31/12/2003): 115.9M words.
- up to 525M words of web data from the University of Washington [10].

In the following we use the term *BN corpus* for the last three corpora (more than 500M words), in contrast to the 27.2M words of the CTS corpus (first two items of the above list). The LM vocabulary contains 51077 words. The baseline LM is constructed as follows: Separate back-off n-gram LMs are estimated for all the above corpora using the modified version of Kneser-Ney smoothing as implemented in the SRI LM toolkit [11]. A single back-off LM was built by merging these models, estimating the interpolation coefficients with an EM procedure.

3.2. Development results

In this section the neural network LM is compared to the back-off LM for different amounts of LM training data. Starting with 7.2M words (SWB data only), the first release of Fisher data was added (12.3M words in total), until all Fisher data was available (27.2M words in total). The neural network LM was trained only on these CTS corpora. Two experiments have been conducted:

1. The neural network LM is interpolated with a back-off LM that was also trained only on the CTS corpora and both LMs are compared.
2. The neural network LM is interpolated with the full back-off LM (CTS 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 (back-off and neural network) are compared on the same data. In the second experiment all the available data is used for the back-off LM to obtain the best results. The perplexities of the neural network and the back-off LM are given in Table 3.

CTS corpus [words]:	7.2M	12.3M	27.2M
In-domain data only:			
back-off LM	62.4	55.9	50.1
neural LM	57.0	50.6	**45.5**
Interpolated with all data:			
back-off LM	53.0	51.1	**47.5**
neural LM	50.8	48.0	44.2

Table 3. Eval03 test set perplexities for the back-off and neural LM as a function of the size of the CTS 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 back-off LM trained on the additional BN corpus. It can been seen that the perplexity of the neural network LM trained only on the CTS data is better than that of the back-off reference LM trained on all data (45.5 with respect to 47.5). Despite these rather small gains in perplexity, consistent word error reductions were observed. The results are summarized in Figure 2 for the three different systems. The first one is that described in [12]. The second system has a lower word error rate than the first one due to several improvements of the acoustic models, and the availability of more acoustic training data. The third system differs from the second one by the amount of training data used for the acoustic and the language model.

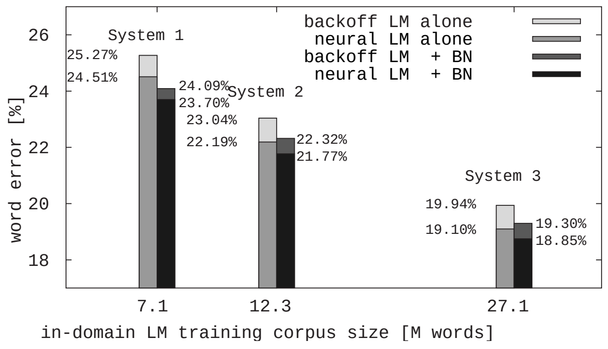

Fig. 2. Word error rates on the Eval03 test set for the back-off LM and the neural network LM, trained only on CTS data (left bars for each system) and interpolated with the BN LM (right bars for each system).

Although the size of the LM training data has almost quadrupled from 7.2M to 27.2M words, a consistent absolute word error reduction of 0.55% can be observed [13]. In all these experiments, it seems that the word error reduction brought by the neural network LM is independent of the other improvements, in particular those obtained by better acoustic modeling and by adding more language model training data. If only the CTS 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 back-off LM that uses in addition 500M words from the BN corpus (22.19% with respect to 22.32% for system 2, and 19.10% with respect to 19.30% for system 3).

3.3. RT04 results

The BBN/LIMSI CTS 2004 English STT evaluation system uses a tightly integrated combination of 5 BBN and 3 LIMSI speech recognition systems, all of LIMSI's systems use the neural network LM (more details are provided in[14]). Table 4 summarizes the word error rates of the three LIMSI components with and without the neural network LM.

	dev04		eval04	
system	back-off	NN	back-off	NN
L1	15.99%	15.52%	18.76%	18.32%
L2	14.71%	14.45%	17.18%	16.86%
L3	14.94%	14.64%	17.33%	17.06%

Table 4. Dev04 word error rates for the LIMSI components of the 2004 BBN/LIMSI CTS evaluation system

It can be seen that the neural network LM gives a 0.5% absolute improvement for the first system. There is a gain of about 0.3% for the L2 and L3 systems, although they are based on the hypothesis of the L1 system, including system combination with two BBN systems.

In fact, two different versions of the neural network LM were used. For the L1 system, one large neural network with 1560 hidden units was trained on all the 27.2M words of CTS data.[2] The interpolation coefficients of this neural network LM with the back-off LM trained on all data was 0.5 (optimized by an EM procedure on dev04). Lattice rescoring with the neural network LM for this system takes about 0.05xRT. For the L2 and L3 system four smaller networks, of dimension 500 and 600 hidden units respectively, were trained on the same data, but with different random initialization of the weights and different random orders of the training examples. It was found that this gives better generalization behavior than one large neural network with the same number of parameters. These four neural network LMs are interpolated with the back-off LM (coefficients: 0.14, 0.14, 0.15, 0.15 and 0.62). Lattice rescoring for these two systems takes about 0.06xRT.

4. APPLICATION TO BN

Originally the neural network LM has been developed specifically for the CTS task where the need is more important due to the limited amount of in-domain data for this task. The situation is not the same for the BN task, for which large amounts of transcripts and newspaper text are available for language model training (more than a billion words). This makes the use of the neural network LM for the BN task questionable. First of all it is impossible to train it on so much data, and second, we are not faced with a data sparseness problem any more. It could be believed that training a back-off LM on so much data, should give a very good model, difficult to improve upon. In the following sections

[2]the word codes are of dimension 50 in all the experiments.

system:	B1	L1	R1	B2	R2	L2	R3
without NN LM	10.98	10.43	9.94	10.12	9.68	10.15	9.61
with NN LM		10.11	9.78	9.95	9.54	9.87	9.26

Table 5. Dev04 word error rates of the different components of the RT04 BBN/LIMSI BN system, System combinations: R1=B1+L1, R2=B1+L1+B2, R3=L1+B2+L2

results are reported showing that the neural network LM is nevertheless quite useful for the BN task.

4.1. LM training data

The reference interpolated 4-gram back-off LM was built from 9 component models trained on subsets of the available text materials including transcriptions of the acoustic BN data (1.8M words); transcriptions of the CTS data (27.2M words); TDT2, TDT3 and TDT4 closed captions (14.3M words); commercially produced BN transcripts from LDC and PSMedia (260M words); CNN web archived transcripts (112M words from Jan'2000-Nov'2003, excluding 15/01/01-28/02/01); and newspaper texts (1463M words). All data predates November 15, 2003 with the period 15/01/2001-28/02/2001 being excluded. The word list contains 65523 words and has an OOV rate of 0.48% on the dev04 set. The word list also contains compound words for about 300 frequent word sequences and about 1000 frequent acronyms. The interpolation coefficients were estimated using an EM procedure to optimize the perplexity on the Dev04 data set.

Despite the fast training algorithms it would take a long time to train the neural network LM on several hundred million words, so a small 27M word subset was selected that was expected to represent the evaluation period of the progress set: the BN transcriptions, TDT2, TDT3 and TDT4 closed captions and 4 months of CNN transcripts from 2001. Although this is less than 2% of the data, the corresponding component LMs account for more than 20% in the interpolation for the final LM. To further speed-up the training process, four small neural networks were trained on all the data (using the same randomization procedure than for the CTS components). The hidden layers were again of size 500 and 600 respectively.

data set	**back-off LM**	**neural LM**
subset (27M words)	148.44	130.70
full corpus (1.9G words)	109.93	105.44

Table 6. Perplexities on the dev04 set

As can be seen from table 6, the neural network LM gives a gain of 12% in perplexity with respect to the back-off 4-gram LM if only the small 27M words corpus is used. The interpolation coefficient of the neural LM is 0.6. Using all data for the back-off LM, the gain reduces to 4% relative, with an interpolation coefficient of 0.4.

4.2. RT04 results

The BBN/LIMSI 2004 English BN evaluation system uses a tightly integrated combination of the BBN and LIMSI speech recognition component systems. First BBN runs a a two pass decoding (system B1, 2.6xRT), followed by a full LIMSI decode that adapts on this result (system L1, 2.7xRT). BBN runs then another 2 pass decode (system B2, 2.1xRT) by adapting on the rover of L1 and B2. Finally LIMSI adapts to the rover of B1, L1 and B2 and runs another full decode (system L2, 1.8xRT). The final result is the combination of L1, B2, and L2. More details of this integrated system are provided in [15].

Both LIMSI systems use the neural network LM to rescore the lattices of the last decoding run, followed by consensus decoding and hypothesis extraction. The additional time required is less than 0.04xRT for each system. There is a word error reduction of 0.3% for the L1 system (10.43 $\rightarrow$ 10.11%) and of 0.2% for the L2 system (10.07 $\rightarrow$ 9.87%). After the submission of the official BBN/LIMSI RT04 BN evaluation system, a contrastive experiment was carried out without using the neural network LM in LIMSI's components. It is surprising to see that this affects in fact all the following runs, i.e. ROVER combinations and cross-site adaptations, resulting in an increase in the word error rate of the overall integrated system by 0.35% (see table 5). In summary, the neural network LM achieves significant reductions in the word error rate although it was trained on less than 2% of the available language model training data.

5. CONCLUSION

In this paper we presented a neural network language model for the RT04 BN and CTS transcription tasks. The main idea is to to perform the estimation of the n-gram probabilities in a continuous space, allowing by this means probability interpolation for any context, without the need to back-off to shorter contexts. Fast algorithms for training and recognition have been described: one training epoch through 27M examples takes about 10 hours and lattice rescoring is done in less than 0.05xRT.

The neural network LM was initially developed for the CTS task where the need is more important due to the limited amount of in-domain CTS LM training data. A consistent word error reduction of about 0.5% was observed altough the CTS LM training data was increased from 7M to more than 27M words. This gain also seems to be independent from other improvements of the system, in particular better acoustic modeling using all the Fisher data.

The BN task is characterized by a huge amount of LM training data since transcriptions and even newspaper text is believed to represent well the speaking style. Although the neural network was trained on less than 2% of the data, a word error reduction of about 0.3% was observed in the integrated BBN/LIMSI system that uses multiples cross-site adaptations and system combinations.

Future work will concentrate on other training criteria that are better related to word error than perplexity, and on unsupervised language model adaptation techniques.

6. ACKNOWLEDGMENT

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

7. REFERENCES

[1] W. Wang and M. Harper, “The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources,” in *Empirical Methods in Natural Language Processing*, 2002.

[2] W. Wang, A. Stolcke, and M. Harper, “The use of a linguistically motivated language model in conversational speech recognition,” in *International Conference on Acoustics, Speech, and Signal Processing*, 2004, pp. 261–264.

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

[4] 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.

[5] 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.

[6] J. Bilmes, K. Asanovic, C.-W. Chin, and J. Demmel, “Using phipac to speed error back-propagation learning,” in *International Conference on Acoustics, Speech, and Signal Processing*, 1997, pp. V:4153–4156.

[7] Intel’s MKL, “Intel math kernel library, http://www.intel.com/software/products/mkl/,” 2004.

[8] ATLAS, “Automatically tuned linear algebra software, http://www.netlib.org/atlas,” 2004.

[9] Y. Bengio and J.-S. Sénécal, “Quick training of probabilistic neural nets by importance sampling,” in *AISTATS Conference*, 2003.

[10] I. Bulyko, M. Ostendorf, and A. Stolcke, “Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures,” in *Human Language Technology Conference*, 2003.

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

[12] 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.

[13] H. Schwenk and J.-L. Gauvain, “Neural network language models for conversational speech recognition,” in *International Conference on Speech and Language Processing*, 2004, pp. 1283–1286.

[14] R. Prasad, S. Matsoukas, C.-L. Kao, J. Ma, D.-X. Xu, T. Colthurst, G. Thattai, O. Kimball, R. Schwartz, J.-L. Gauvain, L. Lamel, H. Schwenk, G. Adda, and F. Lefevre, “The 2004 20xrt bbn/limsi english conversational telephone speech system,” in *2004 Rich Transcriptions Workshop, Pallisades, NY*, 2004, p. in press.

[15] L. Nguyen, S. Abdou, M. Afify, J. Makhoul, S. Matsoukas, R. Schwartz, B. Xiang, L. Lamel, J.-L. Gauvain, G. Adda, H. Schwenk, and F. Lefevre, “The 2004 bbn/limsi 10xrt english broadcast news transcription system,” in *2004 Rich Transcriptions Workshop, Pallisades, NY*, 2004, p. in press.