Available online at www.sciencedirect.com

ScienceDirect

ELSEVIER

Computer Speech and Language 21 (2007) 492–518

COMPUTER SPEECH AND LANGUAGE

www.elsevier.com/locate/csl

Continuous space language models [☆]

Holger Schwenk

Spoken Language Processing Group, LIMSI-CNRS, BP 133, 91403 Orsay cedex, France

Received 19 December 2005; received in revised form 15 September 2006; accepted 15 September 2006
Available online 9 October 2006

Abstract

This paper describes the use of a neural network language model for large vocabulary continuous speech recognition. The underlying idea of this approach is to attack the data sparseness problem by performing the language model probability estimation in a continuous space. Highly efficient learning algorithms are described that enable the use of training corpora of several hundred million words. It is also shown that this approach can be incorporated into a large vocabulary continuous speech recognizer using a lattice rescoring framework at a very low additional processing time. The neural network language model was thoroughly evaluated in a state-of-the-art large vocabulary continuous speech recognizer for several international benchmark tasks, in particular the NIST evaluations on broadcast news and conversational speech recognition. The new approach is compared to four-gram back-off language models trained with modified Kneser–Ney smoothing which has often been reported to be the best known smoothing method. Usually the neural network language model is interpolated with the back-off language model. In that way, consistent word error rate reductions for all considered tasks and languages were achieved, ranging from 0.4% to almost 1% absolute.
© 2006 Elsevier Ltd. All rights reserved.

1. Introduction

Language models play an important role in many applications such as character and speech recognition, translation and information retrieval. In this paper the use of a neural network language model for large vocabulary continuous speech recognition is investigated. Given the acoustic signal x, it is today common practice to introduce the acoustic model $P(x|w)$ and the language model $P(w)$ when searching for the best word sequence w^*:

$$w^* = \arg\max_w P(w|x) = \arg\max_w P(x|w)P(w). \tag{1}$$

A similar decomposition is used in statistical machine translation (Brown et al., 1990). Suppose we want to translate a French source sentence f to an English target sentence e:

$$e^* = \arg\max_e P(e|f) = \arg\max P(f|e)P(e), \tag{2}$$

[☆] This work was partially financed by the European Commission under the projects CORETEX, TC-STAR and CHIL, and by DARPA.
E-mail address: schwenk@limsi.fr.

doi:10.1016/j.csl.2006.09.003

where $P(f|e)$ is the so-called translation model and $P(e)$ the target language model. Statistical machine translation is not considered in this work, but initial studies have shown that the proposed neural network language model can also be used for this task (Schwenk et al., 2006).

Let us denote the word sequence $w_1,\ldots,w_i$ by w_1^i. A statistical language model has to assign a non-zero probability to all possible word strings w_1^n:

$$P(w_1^n) = P(w_1)\prod_{i=2}^{n} P(w_i|w_1^{i-1}). \tag{3}$$

In practice, data sparseness is an important problem, making it impossible to consider the full word history w_1^{i-1} to make the prediction of the word w_i, and equivalence classes are used. Many approaches were proposed that differ in how these equivalence classes are defined. For example in back-off n-gram word language models the context is limited to the $n-1$ previous words (Katz, 1987), or in class language models several words are grouped together into syntactical, semantical or statistical word classes (Brown et al., 1992).

In the domain of language modeling for continuous speech recognition, a large body of interesting work has been carried out and many new approaches have been developed, for instance structured language models (Chelba and Jelinek, 2000) or maximum entropy language models (Rosenfeld, 1996). To the best of our knowledge, these models are, however, not used in large vocabulary continuous speech recognizers that achieve state-of-the-art word error rates. In practice, these speech recognizers are based on back-off n-gram language models, a technique that was introduced almost twenty years ago (Katz, 1987). Of course, the performance of these language models has improved a lot, but this is mainly due to the availability of large text corpora of more than one billion words for training, and to the development of computer infrastructure to deal with such amounts of data. Many new smoothing techniques were developed, see for example Chen and Goodman (1999) for an overview, but the basic procedure is still the same, i.e., backing-off to lower order n-grams for unseen events. Only very recently have new language model approaches been successfully used in large vocabulary continuous speech recognizers, for example the SuperARV language model (Wang et al., 2004) and Random Forest language models (Xu and Mangu, 2005).

In standard back-off n-gram language models words are represented in a discrete space, the vocabulary. This prevents "true interpolation" of the probabilities of unseen n-grams since a change in this word space can result in an arbitrary change of the n-gram probability. Recently, a new approach was proposed based on a *continuous representation* of the words (Bengio et al., 2000, 2001; Bengio et al., 2003). The basic idea is to convert the word indices to a continuous representation and to use a probability estimator operating in this space. Since the resulting distributions are smooth functions of the word representation, better generalization to unknown n-grams can be expected. Probability estimation and interpolation in a continuous space is mathematically well understood and numerous powerful algorithms are available that can perform meaningful interpolations even when only a limited amount of training material is available. Bengio and Ducharme (2001) evaluated their model on two text corpora and achieved significant reductions in perplexity, but to the best of our knowledge, they did not test it on a speech recognition task.

The first application of a neural network language model to large vocabulary continuous speech recognition was performed by Schwenk (2001), Schwenk and Gauvain (2002), showing word error reductions on the DARPA HUB5 conversational telephone speech recognition task. The only other application of a neural network language model to speech recognition we are aware of was later done by Emami et al. (2003) and follow up work of the same authors. Emami et al. (2003) used the neural network language model with a syntactical language model showing perplexity and word error improvements on the Wall Street Journal corpus.

This paper summarizes and extends our continued research on neural network language models for large vocabulary continuous speech recognition (Schwenk and Gauvain, 2003; Schwenk, 2004; Schwenk and Gauvain, 2004a; Schwenk and Gauvain, 2004b; Schwenk and Gauvain, 2005a; Schwenk and Gauvain, 2005b) and provides extensive discussion and experimental evaluation. In the next section we discuss the relationship between the neural network language model as proposed by Bengio and Ducharme (2001) and similar works. Section 2 introduces the architecture of the neural network language model. Several highly efficient training and lattice rescoring algorithms are presented and discussed in detail. We also discuss resampling methods to train the neural network on corpora of several hundred million words. Section 3 presents a thorough evaluation of the neural network language model in state-of-the-art large vocabulary continuous speech recognizers. It was successfully applied to three different languages and several tasks that exhibit different speaking styles, ranging from prepared speech to meetings and conversations. The paper concludes with a discussion of future research.

1.1. Relation to other work

The idea of learning distributed representations for symbolic data was already presented in the early days of connectionism (Hinton, 1986; Elman, 1990) and more recently pursued by Paccanaro and Hinton (2000). There is also other work which describes the use of neural networks for language modeling, but on a much smaller scale and without incorporation into a speech recognition system. In Nakamura and Shikano (1989), a neural network is used to predict word categories. Experiments are reported on the Brown corpus using 89 different categories. Another approach to connectionist natural language processing, based on hierarchically organized modular subnetworks and a central lexicon, was developed by Miikkulainen and Dyer (1991). Xu and Rudnicky (2000) proposed language models using a simple neural network. The input consists of only one word and no hidden units are used, which limits the model to essentially capturing uni-gram and bigram statistics. Bi- and trigram models using neural networks were studied by Castro and Polvoreda (2001); Castro and Prat (2003). A training corpus of 800 k examples was used, but the vocabulary was limited to 29 words.

Similar ideas were also formulated in the context of character-based text compression (Schmidhuber, 1996). Neural networks were used to predict the probability of the next character. The cost function of the neural network language model is also similar to that used in maximum entropy models (Berger et al., 1996), which corresponds to a neural network without a hidden layer. The idea of using a vector space representation of words has also been used in the area of information retrieval. In latent semantic indexing feature vectors of documents and words are learned on the basis of their probability of co-occurences (Deerwester et al., 1990). This idea was also applied to statistical language modeling, for example in (Bellegarda, 1997).

2. Architecture of the neural network language model

The neural network language model has to perform two tasks: first, project all words of the context $h_j = w_{j-n+1}^{j-1}$ onto a continuous space, and second, calculate the language model probability $P(w_j = i|h_j)$ for the given context. It will be shown in the following discussion that both tasks can be performed by a neural network using two hidden layers. The first one corresponds to the continuous representation of all the words in the context and the second hidden layer is necessary to achieve non-linear probability estimation (see Fig. 1). The network is trained by stochastic back-propagation to minimize the perplexity of the training data.

Let N be the size of the vocabulary (typically 40 k,…,200 k) and P the dimension of the continuous space (50,…,300 in the experiments). The inputs to the neural network are the indices of the $n-1$ previous words in the vocabulary. The so-called 1-of-n coding is used, i.e., a N dimensional binary vector where the ith word of the vocabulary is coded by setting the ith element of the vector to 1 and all the other elements to 0. This coding substantially simplifies the calculation of the projection layer since we only need to copy the ith line of the $N \times P$ dimensional projection matrix. The projection matrix is shared for the different word positions in the context. The continuous projections of all the words in the context are concatenated and form the first hidden layer of the neural network (see Fig. 1). A linear activation function is used for this layer. Let us denote the activities of this layer c_l with $l = 1,\ldots,(n-1)\cdot P$. In the continuous space, each word corresponds to a point in R^P, and the learning task consists of estimating the joint probability distribution of a sequence of $n-1$ points corresponding to the context h_j. This is done by the remaining two layers of the neural network. The second hidden layer uses hyperbolic tangent activation functions:

$$d_j = \tanh\left(\sum_l m_{jl}c_l + b_j\right) \quad \forall j = 1,\ldots,H, \tag{4}$$

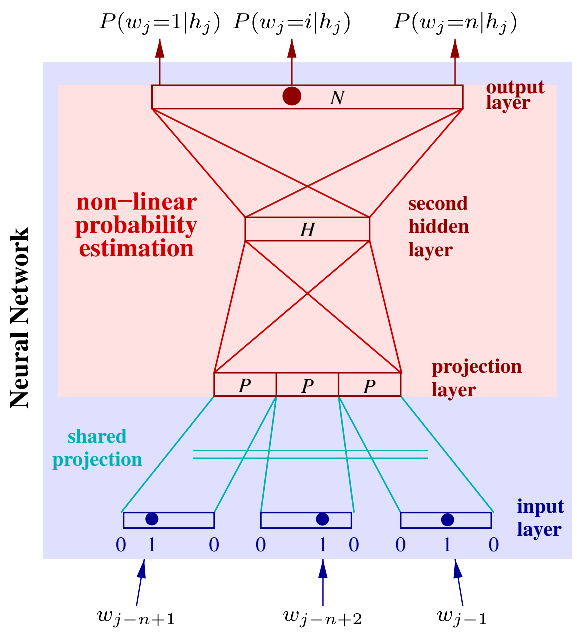

Fig. 1. Architecture of the neural network language model. h_j denotes the context w_{j-n+1}^{j-1}. P is the size of one projection and H and N is the size of the second hidden and output layer, respectively. When short-lists are used the size of the output layer is much smaller than the size of the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

where m_{jl} are the weights between the projection layer and the second hidden layer and b_j are the biases of the second hidden layer.

In contrast to standard language modeling where we want to know the probability of a word i given its context, $P(w_j = i|h_j)$, the neural network simultaneously predicts the language model probability of *all* words in the word list:

$$o_i = \sum_j v_{ij} d_j + k_i, \tag{5}$$

$$p_i = \frac{e^{o_i}}{\sum_{l=1}^N e^{o_l}} \quad \forall i = 1, \ldots, N, \tag{6}$$

where v_{ij} are the weights between the second hidden and the output layer and k_i are the biases of the output layer. The value p_i of the ith output neuron corresponds directly to the probability $P(w_j = i|h_j)$. The so-called softmax normalization (Eq. (6)) can be seen as a smoothed version of the *winner-takes-all* activation model in which the unit with the largest input has output +1 while all other units have output 0 (Bridle, 1990; Bishop, 1995). It is clear that this normalization for each language model probability evaluation is very costly in processing time due to the large size of the output layer, but in the following sections several improvements will be developed so that the approach can be used for large vocabulary continuous speech recognition. For n-gram back-off language models the normalization is performed at the time of construction of the model.

Using matrix/vector notation the operations performed by the neural network can be summarized as follows:

$$\mathbf{d} = \tanh(\mathbf{M} * \mathbf{c} + \mathbf{b}), \tag{7}$$

$$\mathbf{o} = \mathbf{V} * \mathbf{d} + \mathbf{k}, \tag{8}$$

$$\mathbf{p} = \exp(\mathbf{o}) \Bigg/ \sum_{l=1}^{N} e^{o_l}, \tag{9}$$

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.

Training is performed with the standard back-propagation algorithm minimizing the following error function:

$$E = \sum_{i=1}^{N} t_i \log p_i + \epsilon \left(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2 \right), \tag{10}$$

where t_i denotes the desired output of the neural network (target vector), i.e., the probability should be 1.0 for the next word in the training sentence and 0.0 for all the other ones. The first part of the equation is the cross-entropy between the output and the target probability distributions, and the second part is a regularization term that aims at preventing the neural network from overfitting the training data (weight decay). The parameter ϵ has to be determined experimentally using a validation set. Weight decay is not used for the bias terms b_j and k_i.

Following classical learning theory we should minimize the sum of E over all examples, but this generally leads to very slow convergence with training corpora of several thousand or even a million examples (see for examples Bishop (1995) for a general description of neural networks and learning theory). Therefore so-called stochastic back-propagation is used:

- select randomly one four-gram training example $w_1w_2w_3w_4$, i.e., we want to learn the task $P(w_4 = i|w_1w_2w_3)$,
- put the context $h = w_1w_2w_3$ at the input of the neural network and calculate the outputs p_i,
- set the desired outputs to $t_j = \delta_{ij}$, $j = 1,\ldots,N$, supposing that w_4 is the ith word in the vocabulary,
- calculate the gradient of the error function E with respect to all weights. This is done by back-propagating the errors from the outputs to the inputs,
- update all the weights:

$$\omega^{\text{updated}} = \omega^{\text{previous}} - \lambda \frac{\partial E}{\partial \omega} \tag{11}$$

where ω stands for one of the weights or bias of the neural network and λ is the learning rate that has to be chosen experimentally.

This procedure is repeated for several epochs, i.e., presentations of all examples in the training corpus. Note 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. It can be shown that the outputs of a neural network trained in this manner converge to the posterior probabilities (see for example Bishop (1995)). Therefore, the neural network minimizes the perplexity of the training data, under the constraint given by its architecture and the limited computational resources available for optimizing its parameters.

Let us analyze the complexity of calculating the probability $P(w_j = i|h_j)$ of only one n-gram. The activities of the projection layer are obtained by a simple table look-up and can be neglected in the complexity analysis. The calculation of the hidden- and output-layer activities corresponds to a matrix/vector multiplication followed by the application of a non-linear function. The addition of the bias terms is not explicitly considered as this can usually be done in one "multiply-and-add" operation. This gives the following number of floating point operations (Flops):

$$(n-1)P \times H + H + (H \times N) + N. \tag{12}$$

Since N is usually much larger than H, the complexity is dominated by the calculation at the output layer. For typical values of $n = 3$, $N = 64$ k, $P = 50$ and $H = 1024$, about 67 MFlops are needed for one forward pass. It is impractical to train networks of this size on several million examples and to use them in a large vocabulary continuous speech recognizer. In the following sections several improvements are presented to drastically reduce this complexity (Schwenk, 2004).

2.1. Fast recognition

To the best of our knowledge, all state-of-the-art large vocabulary continuous speech recognizers use back-off n-gram language models, and the decoding algorithms are heavily tuned to this type of language model. It is for instance common to use the knowledge whether a particular n-gram exists or if it is backed-off to a lower order n-gram during the construction of the search graph. In addition, it is assumed that the evaluation of the language model probabilities takes a negligible amount of time in comparison to the computations involved for the acoustic model. This is reasonable for a back-off language model, for which a language model request corresponds basically to a table look-up using hashing techniques, but this led to long decoding times in our first experiments when the neural language model was used directly during decoding (Schwenk and Gauvain, 2002). In order to make the neural network language model tractable for large vocabulary continuous speech recognition, several techniques were developed that are detailed below.

2.1.1. Lattice rescoring

Normally, the output of a speech recognition system is the most likely word sequence given the acoustic signal, but it is often advantageous to preserve more information for subsequent processing steps. This is usually done by generating a lattice, i.e., a graph of possible solutions where each arc corresponds to a hypothesized word with its acoustic and language model scores. Fig. 2 shows an example of a lattice that was generated using a trigram back-off language model during decoding. Each word has a unique two word context so that the language model probability that is attached to each arc can be calculated. It can for instance be seen that the arcs leaving from the node at $t = 0.64$ at the bottom of the example lattice have a common context (`not a`), but that the preceding nodes at $t = 0.59$ can not be joined into one node, since the bigram contexts are not unique. Additional simplifications are possible using the knowledge whether a particular n-gram exists or if it is backed-off to a lower order n-gram.

In the context of this work the unchanged large vocabulary continuous speech recognition decoder is used to generate lattices using an n-gram back-off language model. These lattices are then processed by a separate tool and all the language model probabilities on the arcs are replaced by those calculated by the neural network language model (lattice rescoring). In practice, processing is performed in several steps: first a lattice is generated using a bigram back-off language model, these lattices are then rescored using a trigram back-off language model and are pruned in order to reduce their size. Finally these trigram lattices are rescored using the neural network language model. Each time a lattice is rescored with a higher order language model than the one used to generate the lattice, it may be necessary to add some nodes and arcs in order to ensure the longer unique context (lattice expansion). This is in particular the case for the neural network language model since all language model requests use the full $n - 1$ context without ever backing-off to a shorter context. In our experiments, rescoring with a neural network four-gram language model resulted in an increase of the lattice size of about 30% in comparison to the lattice expanded with a four-gram back-off language model.

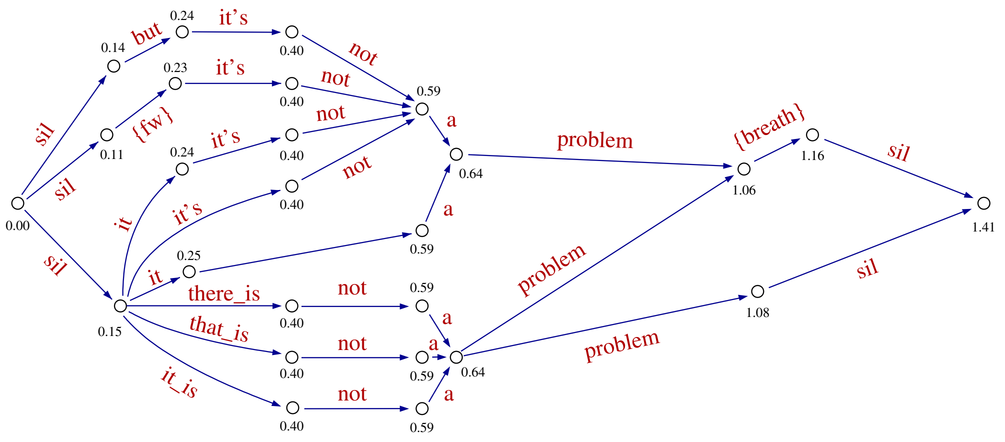

Fig. 2. Example of a lattice produced by the speech recognizer using a trigram language model. A lattice is a graph of possible solutions where each edge corresponds to a hypothesized word with its acoustic and language model scores (for clarity these scores are not shown in the figure). {*fw*} stands for a filler word and {*breath*} is breath noise.

2.1.2. Short-lists

It was demonstrated in Section 2 that most of the calculations are necessary to compute the activities of the output layer, due to its large size. Remember that all outputs need to be calculated in order to perform the softmax normalization even though only one language model probability is needed.[1] Therefore it is suggested to decrease the size of the output layer. One may argue that the language model probabilities of frequent words are likely to be well estimated by a back-off language model, and that the neural network language model should only be used to improve the estimation of rare words. On the other hand, it does not seem to be useful to improve the language model probabilities of words that do not appear very often since this probably won't have a notable effect on the word error rate. Taking this into consideration, we chose to limit the output of the neural network to the s most frequent words, $s \ll N$, referred to as a *short-list* in the following discussion. It is important to remember that all words in the vocabulary are still considered for the input of the neural network. The language model probabilities of words in the short-list ($\widehat{P}_N$) are calculated by the neural network and the language model probabilities of the remaining words ($\widehat{P}_B$) are obtained from a standard n-gram back-off language model. Defining the random variable L for the event that the to be predicted word is in the short-list, this can be formalized as

$$\widehat{P}(w_t|h_t) = \begin{cases} \widehat{P}_N(w_t|h_t, L) \cdot P_B(h_t|L) & \text{if} \quad w_t \in \text{ short-list} \\ \widehat{P}_B(w_t|h_t) & \text{else} \end{cases} \tag{13}$$

where $P_B\ (h_t|L)$ is given by:

$$P_B(h_t|L) = \sum_{w \in \text{short-list}} \widehat{P}_B(w|h_t). \tag{14}$$

It can be considered that the neural network redistributes the probability mass of all the words in the short-list.[2] This probability mass is precalculated and stored in the data structures of the standard four-gram back-off language model. A back-off technique is used if the probability mass for a requested input context is not directly available. Coverages for different short-list sizes are given in the following result sections, i.e., the percentage of language model probability requests that are actually performed by the neural network.

It is also possible to use the neural network to predict this probability mass. In this case an additional output is used to group together all words that are not in the short-list. During lattice rescoring the value of this output is used to rescale the probabilities obtained from the back-off language model. The other outputs of the neural network correspond directly to a language model probability.

In summary, the neural network is never used alone for large vocabulary tasks, i.e., with a vocabulary larger than 16 k words, but we always need a back-off language model in order to process the language model predictions of rare words. We will call this the short-list back-off language model. In addition, perplexity and word error improvements were observed when interpolating the neural network language model with a back-off language model. These two back-off language models can be identical, but this is not necessary. We do not want the short-list back-off language model to be too big since the probability mass for all three-grams must be precalculated and stored. Usually this language model is trained on in-domain data only (less than 40 M words), eventually using a cut-off on the four-grams. The back-off language model used for interpolation, however, can be of any size and it is trained on all available data.

2.1.3. Sorting and reordering

During lattice rescoring, language model probabilities with the same context h_t are often requested several times on potentially different nodes in the lattice (for instance the trigram "`not a problem`" in the lattice shown in Fig. 2). Collecting and grouping together all these calls prevents multiple forward passes since all language model predictions for the same context are immediately available at the output layer (see Fig. 1).

[1] Experiments using lattice rescoring with unnormalized language model scores led to much higher word error rates.

[2] Note that the sum of the probabilities of the words in the short-list for a given context is normalized: $\sum_{w \in \text{short-list}} \widehat{P}_N(w|h_t) = 1$.

2.1.4. Bunch mode

Further improvements can be obtained by propagating several examples at once through the network, which is also known as bunch mode (Bilmes et al., 1997). In comparison to Eq. (7) and (8) this results in using matrix/matrix instead of matrix/vector operations:

$$\mathbf{D} = \tanh\,(\mathbf{MC} + \mathbf{B}), \tag{15}$$

$$\mathbf{O} = \mathbf{VD} + \mathbf{K}, \tag{16}$$

where **B** and **K** are obtained by duplicating the bias **b** and **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, for instance using SSE instructions for Intel processors (Intel Math Kernel Library, 2005). Although the number of floating point operations to be performed is strictly identical to single example mode, an up to ten-fold execution speed-up can be observed depending on the sizes of the matrices. Typical execution times to calculate the perplexity on a data set of 27 k words are as follows: 164 s when no bunch mode is used, and 77, 32, 23, 19 and 16 s with a bunch of size 2, 8, 16, 32 and 128 examples, respectively.

The development data of the 2004 Nist conversational speech recognition evaluation consists of more than 3 h of speech comprised of 3779 conversations sides. The lattices generated by the speech recognizer for this test set contain on average 282 nodes and 759 arcs per conversation side. In total 2.4 million four-gram language model probabilities were requested out of which 2.3 million (84.7%) were processed by the neural network, i.e., the word to be predicted is among the 2000 most frequent words (the short-list). After collecting and grouping all language model calls in each lattice, only 359 thousand forward passes through the neural network were performed, giving a cache hit rate of about 84%. Using a bunch size of 128 examples, the total processing time took less than 9 minutes on an dual Intel Xeon 2.8 GHz server, i.e., in 0.05 times real time (xRT).[3] This corresponds to about 2.3 billion floating point operations per second (2.3 GFlops). The neural network used in this benchmark has a hidden layer of 1560 neurons and the short-list is of length 2000. Lattice rescoring without bunch mode and grouping of all calls in one lattice is about ten times slower.

It is important to note that the complexity of the neural network language model increases only *linearly* with the order of the *n*-gram and with the size of the vocabulary (or alternatively the length of the short-list). Although we consider only four-grams in this paper, longer span language models result in a negligible increase of the memory and time complexity, both with respect to training and with respect to lattice rescoring. This is a major advantage in comparison to back-off language models whose complexity increases *exponentially* with the context length. In practice, five-grams or even higher order back-off language models are rarely used due to the data sparseness problem inflicted on such model.

2.2. Fast training

One of the major concerns was to achieve fast probability evaluations so that the neural network language model can be used in a large vocabulary continuous speech recognizer. The time to train the neural network does not affect the recognition process, but long training times prevent fast system development, the exploration of several alternative architectures and the use on large training corpora.

In speech recognition applications, language models are usually trained on text corpora of several million words. The neural networks used for these tasks have about 1000 neurons in the last hidden layer and the size of the short-list is at least 2000. This results in more than 2 million parameters. Training such large networks on several million examples is a significant challenge and a straightforward implementation of the stochastic back-propagation algorithm would take a considerable amount of time. Therefore several speed-up techniques have been proposed in the literature, in particular parallel implementations (Bengio et al., 2003), resampling techniques (Bengio and Sénécal, 2003) and hierarchical decomposition (Morin and Bengio, 2005). Parallel stochastic back-propagation of neural networks requires connections between the processors with very low latency, 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

[3] In speech recognition, processing time is measured in multiples of the length of the speech signal, the real time factor xRT. For a speech signal of 2 h, a processing time of 3xRT corresponds to 6 h of calculation.

use of cache and data prefetch capabilities of processors. This is not the case for resampling techniques. Therefore, a fixed size output layer was used and the words in the short-list were rearranged in order to occupy continuous locations in memory.

In an initial implementation, standard stochastic back-propagation and double precision for the floating point operations were used in order to ensure good convergence. Despite careful coding and optimized libraries for the matrix/vector operations (Intel Math Kernel Library, 2005), one epoch through a training corpus of 12.4 M examples took about 47 h on a Pentium Xeon 2.8 GHz processor. This time was reduced by more than a factor of 30 using:

- *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 (SMP)*: parallel matrix algebra using two CPUs in an off-the-shelf PC (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 operations were performed (similar to Bilmes et al., 1997):

(1) Update the bias at the output layer:

$$\mathbf{k} = \mathbf{k} - \lambda\Delta\mathbf{K} * \mathbf{i}, \tag{17}$$

where $\mathbf{i} = (1,1,\ldots,1)^{\mathrm{T}}$, with a dimension of the bunch size.

(2) Back-propagate the errors to the hidden layer:

$$\Delta\mathbf{B} = \mathbf{V}^{\mathrm{T}} * \Delta\mathbf{K}. \tag{18}$$

(3) Update the weights from the hidden to the output layer:

$$\mathbf{V} = -\lambda\Delta\mathbf{K} * \mathbf{D}^{\mathrm{T}} + \alpha\mathbf{V}. \tag{19}$$

(4) Calculate the derivatives of the tanh-activation function at the hidden layer:

$$\Delta\mathbf{B} = \Delta\mathbf{B} \odot (1 - \mathbf{D} \odot \mathbf{D}), \tag{20}$$

where the sign $\odot$ denotes element-wise multiplication.

(5) Update the bias at the hidden layer:

$$\mathbf{b} = \mathbf{b} - \lambda\Delta\mathbf{B} * \mathbf{i}. \tag{21}$$

(6) Back-propagate the errors to the projection layer:

$$\Delta\mathbf{C} = \mathbf{M}^{\mathrm{T}} * \Delta\mathbf{B}. \tag{22}$$

(7) Update the weights from the projection to the hidden layer:

$$\mathbf{M} = -\lambda\Delta\mathbf{B} * \mathbf{C}^{\mathrm{T}} + \alpha\mathbf{M}. \tag{23}$$

Note that the back-propagation and weight update step, including weight decay, is performed in one operation using the GEMM function of the BLAS library (Eq. (19) and (23)). For this, the weight decay factor ϵ is incorporated into $\alpha = 1 - \lambda\epsilon$. This simple modification speeds up training by a factor of 1.3.

Bunch mode is not used to update the projection matrix since we can not group together several vectors in continuous locations in memory. $\Delta\mathbf{C}$ in Eq. (22) is a matrix of dimension $(n-1)\cdot P$ times the number of examples in one bunch. Each column corresponds to the concatenated continuous representations of the words in a given context. Let us denote c_r, $r = 1,\ldots,(n-1)$ the continuous representation of one word in this context (c_r is a P dimensional vector), and i the index of the corresponding word in the vocabulary. Then, the weight update step for this word is given by:

$$x_i = x_i + \Delta c_r \tag{24}$$

Table 1
Training times reflecting the different improvements (on an Intel Pentium Xeon CPU at 2.8 GHz)

Size of training data	Precision		Bunch mode						SMP
	Dbl	Float	2	4	8	16	32	128	128
1.1 M words	2 h	1 h16	37 m	31 m	24 m	14 m	11 m	8 m18	5 m50
12.4 M words	47 h	30 h	10 h12	8 h18	6 h51	4 h01	2 h51	2 h09	1 h27

where x_i is the ith line of the $N \times P$ dimensional projection matrix. This operation is repeated for all the words in one context, and all the contexts in the bunch.

Table 1 summarizes the effect of the different techniques to speed up training. Extensive experiments were first performed with a training corpus of 1.1 M words and then applied to a larger corpus of 12.4 M words. Bilmes et al. (1997) reported that the number of epochs needed to achieve the same mean squared error increased with the bunch size. In our experiments the convergence behavior also changed with the bunch size, but after adapting the learning parameters of the neural network only small losses in perplexity were observed, and there was no impact on the word error rate when the neural language model was used in lattice rescoring.

2.3. Grouping of training examples

Stochastic back-propagation consists in cycling through the training corpus, in random order, and to perform a forward/backward pass and weight update for each example. This can be potentially optimized using the fact that many four-grams appear several times in a large training corpus, and that many four-grams share a common trigram context. The idea is to group together these examples and to perform only one forward and backward pass through the neural network. The only difference is that there are now multiple output targets for each input context. Furthermore, four-grams appearing multiple times can be learned at once by multiplying the corresponding gradients by the number of occurrences. This is equivalent to using bunch mode where each bunch includes all examples with the same trigram context. Alternatively, the cross-entropy targets can also be set to the empirical posterior probabilities, rather than 0 or 1. Formally, all four-grams with a common context $w_1w_2w_3$ are collected and one forward/backward pass is performed using the following targets:

$$t_j = \frac{\text{number of four-grams } w_1w_2w_3w_4 = j}{\text{number of three-grams } w_1w_2w_3} \quad \forall j = 1, \ldots, N. \tag{25}$$

In stochastic back-propagation with random presentation order, the number of forward and backward passes corresponds to the total number of four-grams in the training corpus which is roughly equal to the number of words.[4] With the new algorithm the number of forward and backward passes is equal to the number of *distinct trigrams* in the training corpora. One major advantage of this approach is that the expected gain, i.e., the relation between the total number of four-grams and distinct trigrams, increases with the size of the training corpus. Although this approach is particularly interesting for large training corpora (see Table 2), we were not able to achieve the same convergence as with random presentation of individual four-grams. Overall, the perplexities obtained with the grouping algorithm were higher and this technique was not used for the applications of the neural network language model described in the following sections.

2.4. Resampling the training data

The above described algorithms were used to train the neural network language model on corpora of up to 30 M words (see Section 3 for a detailed description). For several tasks, however, much more data is available, in particular when newspaper texts are representative of the style of language we want to model. As an extreme case, we could mention language modeling of American English broadcast news for which almost 2 billion

[4] The sentences were surrounded by begin and end of sentence markers. The first two words of a sentence do not form a full four-gram, e.g., a sentence of three words has only two four-grams.

Table 2
Training times for stochastic back-propagation using random presentation of all four-grams in comparison to grouping all four-grams that have a common trigram context (bunch size = 32)

Words in corpus	229 k	1.1 M	12.4 M
Random presentation			
# Four-grams	162 k	927 k	11.0 M
Training time	144 s	11 m	171 m
Grouping			
# Distinct trigram	124 k	507 k	3.3 M
Training time	106 s	6 m35 s	58 m

words are available. It is of course impractical to train a neural network on this much data. Even when processing time would not be an issue, the convergence behavior will certainly be suboptimal. It should be noted that building back-off language models on large amounts of data is in principle no problem, as long as powerful machines with enough memory are available in order to calculate the word statistics.

We propose an algorithm that makes it possible to train the neural network on arbitrarily large training corpora. The basic idea is quite simple: instead of performing several epochs over the whole training data, a different small random subset is used at each epoch. This procedure has several advantages:

- There is no limit on the amount of training data.
- After some epochs, it is likely that all the training examples were seen at least once.
- Changing the examples after each epoch adds noise to the training procedure. This potentially increases the generalization performance.

This algorithm is summarized in Fig. 3. It is important to note that we resample independently from several corpora. For example, we may choose to use all the examples of a small in-domain corpus, but to resample only small fractions of large complementary corpora. In that way, it is possible to focus on important data and still use large amounts of newspaper texts. The parameters of the resample algorithm are the size of the random subsets that are used at each epoch. If we chose the same resampling fraction for each corpus, the resampling algorithm would be almost identical to regular stochastic gradient descent, using a particular randomization of the order of the examples and testing after each sub-epoch.

3. Experimental evaluation

In this section, a thorough evaluation of the neural network language model on several large vocabulary continuous speech recognition (LVCSR) tasks and three different languages (English, French and Spanish) is presented. Evaluating a new language model smoothing technique like the neural network approach on different tasks is interesting since we are faced with different speaking styles, ranging from prepared speech (broadcast news task) to unconstrained conversational speech (Switchboard task, lecture and seminar transcription). In broadcast news speech, the grammar is usually well respected, but the vocabulary tends to be

```
Repeat
    Select training data:
      – Extract random subsets of examples from the different corpora
      – Shuffle data
    + Train network for one epoch (performing weight updates after each example)
    + Test performance on development data
Until convergence
```

Fig. 3. Training algorithm for large corpora.

rather large due to a large amount of proper names. The vocabulary for conversational speech, on the other hand, is probably more limited, but there are frequent syntactical errors that are difficult to model. The amount of available language model training data differs also quite a bit for the different tasks. The main question is whether newspaper and Web texts, two sources of large amounts of text, are representative of the speaking style of the considered task. The neural network language model was in fact first used for the modeling of conversational speech in the NIST Switchboard LVCSR task in order to see whether it can take better advantage of the limited amount of *in-domain* language model training data. The three considered languages (English, French and Spanish) also exhibit particular properties: French has for instance a large number of homophones in the inflected forms that can only be distinguished by a language model.

For each task the amount of language model training data is given, the training procedures are detailed and comparative results are presented. It should be mentioned that not all neural network training algorithms were systematically applied on all tasks. The recently developed resampling algorithm has not yet been used for the English broadcast news and conversational speech recognition systems. In general, details on the speech recognition systems are omitted since this is not the direct focus of this work and the reader is referred to the bibliography. A detailed description of the technology that is common to all speech recognizers can be found in Gauvain et al. (2002).

In all the described experiments, the neural network language model is compared to a four-gram back-off model. Modified Kneser–Ney smoothing is used, which has often been reported to be the best known single smoothing method (but see Goodman (2001) for a combination of many smoothing techniques). All back-off language models were trained using the SRI language model toolkit (Stolcke, 2002). When large amounts of language model training data are available, the resampling algorithm is used or the neural network model is trained on a subset of the data only. It is recalled that a back-off language model is always needed to predict the language model probabilities of words that are not in the short-list. It was also useful to interpolate the neural network with a back-off language model, even when the latter was trained on more data. This always led to reductions in perplexity and word error rate. In the following this interpolated model will be denoted by *hybrid language model*.

When calculating the perplexity of a sentence using a four-gram language model two particularities must be considered. First, the sentence is surrounded by begin and end of sentence special symbols, denoted $\langle \texttt{s} \rangle$ and $\langle \backslash \texttt{s} \rangle$, respectively. Second, we do not have a full three word context for the first n-grams. The sentence "`This is an example`", for instance, results in the following calculation:

$$\begin{aligned} P(\texttt{This is an example}) = {} & P(\texttt{This}|\langle \texttt{s} \rangle) \cdot P(\texttt{is}|\langle \texttt{s} \rangle \texttt{This}) \cdot P(\texttt{an}|\langle \texttt{s} \rangle \texttt{This is}) \\ & \cdot P(\texttt{example}|\texttt{This is an}) \cdot P(\langle \backslash \texttt{s} \rangle|\texttt{is an example}). \end{aligned}$$

The same observations apply when rescoring lattices. It is possible to train neural network language models with a one, two and three word context in order to process all n-grams appearing in a sentence or a lattice. Alternatively, we could repeat the begin of sentence symbol and only consider four-grams, for instance $P(\texttt{This}|\langle \texttt{s} \rangle \langle \texttt{s} \rangle \langle \texttt{s} \rangle)$ at the beginning of the sentence. In this work we decided to simply use the back-off bi- and trigram language model for these short context n-grams. It has in fact turned out that only a few bi- and trigram language model probabilities are requested when rescoring a lattice.

3.1. English conversational speech

International interest in research on the recognition of conversational telephone speech (CTS) started with the introduction of the Switchboard task in the NIST evaluation in 1997. The goal is to automatically transcribe discussions between two private people on the (cellular) telephone. Collection of the Switchboard data started in 1992 at Texas Instruments and was then pursued in several phases, mostly in the framework of the DARPA EARS project.[5] Language modeling of conversational speech is a very challenging task due to the unconstrained speaking style, frequent grammatical errors, hesitations, start-overs, etc. This is illustrated by the following examples that were extracted from the language model training material:

[5] *E*ffective *A*ffordable *R*eusable *S*peech-to-Text.

– *But yeah so that is that is what I do.*
– *I I have to say I get time and I get a few other.*
– *Possibly I don't know I mean I had one years ago so.*
– *Yeah yeah well it's it's.*

Language modeling of conversational speech suffers from a lack of adequate training data since the main source is audio transcriptions. In general, newspaper text, a source of large amounts of possible training data, is not representative for the language encountered in unconstrained human/human conversations. 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 broadcast news material (Iyer and Ostendorf, 1999) and on the Internet (Bulyko et al., 2003). In addition to these techniques, the neural network language model is used here in order to see if it can take better advantage of the limited amount of training data. In all experiments on this task, the neural network model was interpolated with a back-off model.

We present results corresponding to different stages in the development of a conversational telephone speech recognition system, in particular with respect to the amount of available acoustic and language model training data. This makes it possible to analyze the improvements obtained by the neural network language model for different amounts of training data (see Section 3.1.2). Finally, the speech recognizer developed for the Nist 2004 Rich Transcription evaluation is a complex combination of several components from two research institutes: BBN Technologies in Cambridge and LIMSI–CNRS in France (Prasad et al., 2004). Significant word error reductions were obtained for the combined system although the hybrid language model was only applied to the LIMSI components (see Section 3.1.3). All the neural networks have a 50 dimensional projection layer and the length of the short-list was set to 2000. The size of the hidden layer was adapted to the amount of training data (between 200 and 1500 neurons).

3.1.1. Language model training data

The following corpora were used for language modeling:

- CTS transcripts with breath noise (6.1 M words): 2.7 M words of the Switchboard transcriptions provided by LDC, 1.1 M words of CTRAN transcriptions of Switchboard-II data, 230 k words of cellular training data, 215 k word of the CallHome corpus transcriptions, 1.7 M words of Fisher data transcribed by LDC and transcripts of the 1997–2001 evaluation sets.
- CTS transcripts without breath noise (21.2 M words): 2.9 M words of Switchboard transcriptions provided by the Mississippi State University, 18.3 M words of Fisher data transcribed by WordWave and distributed by BBN (Kimball et al., 2004).
- Broadcast news transcriptions from LDC and from PSMedia: 260.3 M words.
- CNN transcripts from the CNN archive: 115.9 M words.
- up to 525 M words of web data, collected by the University of Washington (Bulyko et al., 2003).

The last three corpora are referred to as broadcast news (BN) corpus, in contrast to the 27.3 M words of the CTS corpus (first two items of above list). The baseline language model is constructed as follows: separate back-off n-gram language models are estimated for all the above corpora and then merged together estimating the interpolation coefficients with an EM procedure. This procedure usually achieves better results than directly building a language model on all the data.

3.1.2. Development results

In this section three systems were tested that differ in the amount of acoustic and language model training data used. These data became available as the Ears project advanced. The first system was trained on 229 h of transcribed speech. The language models were build using 7.2 M words of in-domain data (Switchboard data only) and 240 M words of broadcast news data. The word list consists of 42 k words. This system is described in detail in Gauvain et al. (2003). This second system was trained on more than 280 h of transcribed speech. The language models were build using 12.3 M words of in-domain data (addition of the first release of Fisher data)

Table 3
Perplexities on the 2003 evaluation data for the back-off and the hybrid LM as a function of the size of the CTS training data

CTS corpus (words)	7.2 M	12.3 M	27.3 M
In-domain data only			
Back-off LM	62.4	55.9	50.1
Hybrid LM	57.0	50.6	**45.5**
Interpolated with all data			
Back-off LM	53.0	51.1	**47.5**
Hybrid LM	50.8	48.0	44.2

and 347 M words of broadcast news data. The word list consists of 50 k words. All available data was used to train the language model of the third system: 27.3 M words of in-domain (complete release of Fisher data) and 901 M words of broadcast news. The acoustic model was trained on 450 h. The word list consists of 51 k words.

The neural network language model was trained on the in-domain data only (CTS corpora). Two types of experiments were conducted for all three systems:

(1) The neural network language model was interpolated with a back-off language model that was also trained on the CTS corpora only and compared to this CTS back-off language model.
(2) The neural network language model was interpolated with the full back-off language model (trained on CTS and BN data) and compared to this full language model.

The first experiment allows us to assess the real benefit of the neural language model since the two smoothing approaches (back-off and hybrid) are compared on the same data. In the second experiment all the available data was used for the back-off model to obtain the overall best results. The perplexities of the hybrid and the back-off language model are given in Table 3.

A perplexity reduction of about 9% relative is obtained independently of the size of the language model training data. This gain decreases to approximately 6% after interpolation with the back-off language model trained on the additional BN corpus of out-of domain data. It can be seen that the perplexity of the hybrid language model trained only on the CTS data is better than that of the back-off reference language model trained on all of the data (45.5 with respect to 47.5). Despite these rather small gains in perplexity, consistent word error reductions were observed (see Fig. 4).

Although the size of the language model training data has almost quadrupled from 7.2 M to 27.3 M words, use of the hybrid language model resulted in a consistent absolute word error reduction of about 0.5%. In all of these experiments, it seems that the word error reductions achieved by the hybrid language model are independent of the other improvements, in particular those obtained by better acoustic modeling and by adding

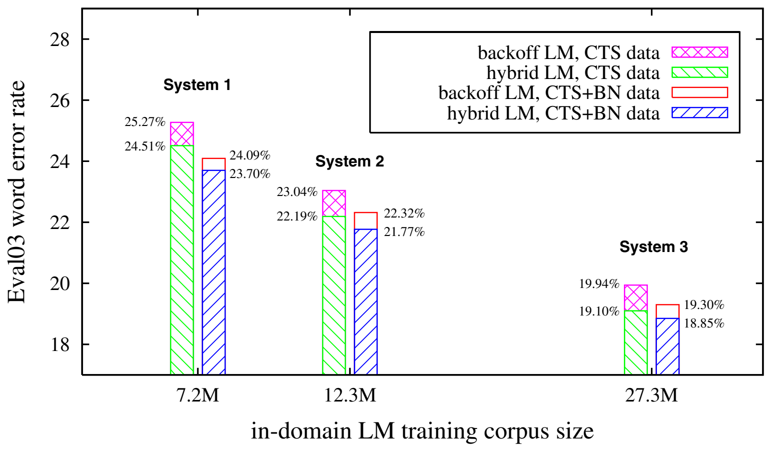

Fig. 4. Word error rates on the 2003 evaluation test set for the back-off LM and the hybrid LM, trained only on CTS data (left bars for each system) and interpolated with the broadcast news LM (right bars for each system).

Table 4
Word error rates on the 2004 development and evaluation set for the LIMSI components of the 2004 BBN/LIMSI CTS evaluation system

System	Dev04		Eval04	
	Back-off LM (%)	Hybrid LM (%)	Back-off LM (%)	Hybrid LM (%)
LIMSI 1	15.99	15.52	18.76	18.32
LIMSI 2	14.71	14.45	17.18	16.86
LIMSI 3	14.94	14.64	17.33	17.06

more language model training data. When only the CTS data is used (left bars for each system in Fig. 4), the hybrid language model achieves an absolute word error reduction of about 0.8%. Note also that the hybrid language model trained on at least 12.3 M words is better than the back-off language model that uses in addition 500 M words of the BN corpus (22.19 with respect to 22.32% for system 2, and 19.10 with respect to 19.30% for system 3).

3.1.3. Results in the 2004 NIST rich transcription evaluation

LIMSI and BBN in Cambridge developed a tightly integrated combination of five BBN and three LIMSI speech recognition systems for the 2004 NIST English CTS evaluation (Prasad et al., 2004). The second and third LIMSI system are not full decodes but perform word lattice rescoring of the lattices generated by the first LIMSI system. All of the LIMSI systems use the hybrid language model. Table 4 summarizes the word error rates of the three LIMSI components.

It can be seen that the hybrid language model gives an absolute improvement of about 0.5% for the first LIMSI system. There is an additional gain of about 0.3% for the second and third systems, although they are based on the hypothesis of the first system, including system combination with two BBN systems.

In fact, two different versions of the hybrid language model were used. For the first system, one large neural network with 1560 hidden units was trained on all 27.3 M words of the CTS data. The interpolation coefficient of this neural network language model with the back-off language model trained on all data was 0.5 (optimized by an EM procedure on the development data). Lattice rescoring with the hybrid language model for this system takes about 0.05xRT. For the second and third system four smaller networks, of dimension 500 and 600 hidden units, respectively, were trained on the same data, but with different random initializations 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 amount of parameters. These four neural network language models are interpolated with the back-off language model (coefficients: 0.14, 0.14, 0.15, 0.15 and 0.42). Lattice rescoring for these two systems takes about 0.06xRT.

3.2. English broadcast news

In the beginning, the neural network language model was developed specifically for the CTS task where the need for better smoothing techniques is important due to the limited amount of in-domain data for this task. The situation is not the same for the broadcast news task, for which commercially produced transcripts, CNN transcripts from the Web or even newspaper text seem to be appropriate for language model training, resulting in large amounts of available data. This made the use of the neural network language model for the broadcast news task questionable. First, it is quite difficult to train it on more than a billion words (the resampling algorithm was only developed later and it was not applied to this data), and second, data sparseness may be less of a problem, which is the major motivation for using the neural network language model. It is conceivable that training a back-off language model on so much data would give a very good model, that is difficult to improve upon. In the following sections results are reported showing that the neural network language model is nevertheless quite useful for the broadcast news task.

3.2.1. Language model training data

The reference interpolated four-gram back-off language model was built from nine component models trained on subsets of the available text materials including transcriptions of the acoustic BN data (1.8 M

Table 5
Perplexities on the 2004 development set

LM training data	Back-off LM	Hybrid LM
Subset (27 M words)	148.4	130.7
Full corpus (1.9 G words)	109.9	105.4

In the second line the full corpus is only used to train the back-off LM, the neural networks are always trained on the subset of 27 M words only.

words); transcriptions of the CTS data (27.4 M words); TDT2, TDT3 and TDT4 closed captions (14.3 M words); commercially produced BN transcripts from LDC and PSMedia (260 M words); CNN web archived transcripts (112 M); and newspaper texts (1463 M words). The word list contains 65 523 words and has an out-of-vocabulary (OOV) rate of 0.48% on the development set of the 2004 NIST evaluations. The interpolation coefficients were estimated using an EM procedure to optimize the perplexity on the development data. About 450 h of transcribed speech were used to train the acoustic models.

Since the resampling algorithm was developed after these experiments were conducted, we selected a small 27 M word subset that was expected to be representative of the period of the test data: BN transcriptions, TDT2, TDT3 and TDT4 closed captions and four months of CNN transcripts from 2001. Although this amounts to less than 2% of the data, the corresponding component language models account for more than 20% in the interpolation for the final language model. To further speed-up the training process, four small neural networks were trained on this data (using the same randomization procedure as for the CTS components). The hidden layers were again of size 500 and 600, respectively, and the short list length was set to 2000.

As can be seen from Table 5, the hybrid back-off/neural network language model gives a gain of 12% in perplexity with respect to the back-off four-gram language model if only the small 27 M word corpus is used. The interpolation coefficient of the neural language model is 0.6. Using all data for the back-off language model only, the gain reduces to 4% relative, with an interpolation coefficient of 0.4.

3.2.2. Results in the 2004 NIST rich transcription evaluation

LIMSI and BBN also developed a joint system for the English broadcast news task of the 2004 NIST evaluation. System combination is performed using cross-site adaptation and the ROVER algorithm (Fiscus, 1997). BBN runs 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 two 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 Nguyen et al. (2004).

Both LIMSI systems use the interpolated neural network/back-off language model to rescore the lattices. The results are summarized in Table 6. The hybrid language model achieved a word error reduction of 0.3% absolute for the L1 system (10.43 → 10.11%) and of 0.2% for the L2 system (10.07 → 9.87%). In a contrastive experiment, the same ROVER combination was performed, but without using the hybrid language model in LIMSI's components (see Table 6 first line). It is surprising to see that this affects in fact all of 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% (9.26 → 9.61%). In summary, the hybrid neural network back-off language model achieves significant reductions in the word error rate although it was trained on less than 2% of the available language model training data.

Table 6
Word error rates on the 2004 development set of the different components of the RT04 BBN/LIMSI BN system

System	B1	L1	R1	B2	R2	L2	R3
Back-off LM	10.98%	10.43%	9.94%	10.12%	9.68%	10.15%	**9.61**%
Hybrid LM		10.11%	9.78%	9.95%	9.54%	9.87%	**9.26**%

The hybrid LM is used in the LIMSI components only. ROVER System combinations: R1 = B1 + L1, R2 = B1 + L1 + B2, R3 = L1 + B2 + L2.

3.3. French broadcast news

Given the high flexional properties of the French language, transcribing French broadcast news is more challenging than English. This is in part due to the large number of homophones in the inflected forms. The speech recognizer for this task is based on the same core technology as the English system. The primary differences between the English and French broadcast news systems are: a 200 k vocabulary to overcome the lower lexical coverage in French (including contextual pronunciations to model liaisons) and a case sensitive language model. The acoustic models were trained on about 190 h of broadcast news data. Two versions of the speech recognizer are used: a classical 7xRT system and a fast 1xRT system. In both systems, the neural network language model is used to rescore the lattices of the last decoding pass. The word list includes 200 k words for the 7xRT and 65 k words for the real-time system. The OOV rate is 0.40% and 0.95%, respectively on a development set of 158 k words. Both systems are described in more detail in Gauvain et al. (2005). The 7xRT system was evaluated in the first French Technolangue-Ester automatic speech recognition benchmark test where it achieved the lowest word error rate by a significant margin (Galliano et al., 2005).

3.3.1. Language model training data

The following resources were used for language modeling:

- Transcriptions of the acoustic training data (4.0 M words).
- Commercial transcriptions (88.5 M words).
- Newspaper texts (508 M words).
- Web data (13.6 M words).

As before, language models were first built separately on each corpus and then merged together. Table 7 summarizes the characteristics of the individual text corpora. Although the detailed transcriptions of the audio data represent only a small fraction of the available data, they get an interpolation coefficient larger than 0.4. This shows clearly that they are the most appropriate text source for the task. The commercial transcripts and the newspaper and Web texts reflect less well the speaking style of broadcast news audio, but this is to some extent counterbalanced by the large amount of data. One could say that these texts are helpful for learning the general grammar of the language.

Several experiments were performed on this task, using the 7xRT as well as the real-time system. Following the above discussion, it seems natural to start with training a neural network language model on the transcriptions of the acoustic data only. In the next section the importance of the length of the short-list is first analyzed for the 7xRT speech recognizer with its 200 k word list. We then switch to the real-time system and describe a series of experiments with different amounts of training data, including the use of the data resampling algorithm. Finally, results with the 7xRT system using all data are presented.

3.3.2. Selection of the short-list

In the previous experiments on English BN and CTS, a 65 k vocabulary was used and the short-list was of a size of 2000. This resulted in a coverage of about 90%, i.e., nine out of ten language model requests were processed by the neural network. The French 7xRT BN system uses a word list of 200 k and consequently larger

Table 7
Characteristics of the text corpora used to build the baseline back-off LMs for the French BN systems

LM training data	# Words in training (in M)	65 k Word list		200 k Word list	
		Perpl.	Coeffs.	Perpl.	Coeffs.
Acoustic transcriptions	4	107.4	0.43	117.9	0.41
Commercial transcriptions	88.5	137.8	0.14	142.5	0.15
Newspaper texts	508	103.0	0.35	109.3	0.35
Web texts	13.6	136.7	0.08	141.7	0.09
All interpolated	614	70.2	–	74.2	–

Perplexity is measured on a development set of 158 k words. The interpolation coefficients were obtained using an EM procedure.

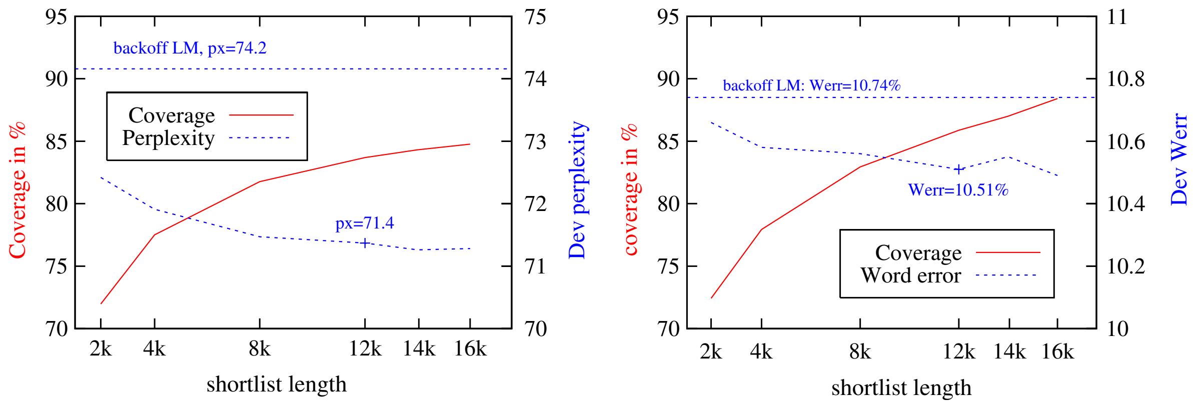

Fig. 5. Analysis of different short-list sizes (7xRT French BN system). Left figure: coverage and perplexity on the development data. Right figure: Coverage and word error on the development data when rescoring lattices.

short-list sizes were investigated. Fig. 5 left shows the coverage and the perplexity of the neural network language model on the development data as a function of the short-list size. In all experiments in this section, the neural network language model was trained on the transcriptions only and interpolated with the back-off language model trained on all data. The other parameters are: $P = 50$ and $H = 500$. The initial learning rate was set to 5×10^{-3} and the weight decay factor to 3×10^{-5}.

It can be clearly seen that a short-list of a length of 2000 is insufficient for this task (the coverage is only 72%) and that better results can be obtained with larger output layers. However, the curves flatten out with increasing size and a short-list of a length of 12 k was used in most of the experiments. In this case the perplexity decreases from 74.2 with the back-off model to 71.4. The right part of Fig. 5 shows the results of rescoring the lattices with the hybrid model. The coverage, i.e., the number of language model probability requests in the lattice performed by the neural network, increases from 72% (short-list length = 2 k) to 88% (length = 16 k). With 12 k short-lists the word error decreases from 10.74% to 10.51% with an additional decoding time of 0.11xRT. Longer short-lists do not lead to further word error reductions.

The real-time systems use only a 65 k word list and the short-list length was fixed at 8 k, in particular to fulfill the severe time constraints. The other parameters of the neural network were the same as the ones used with the 200 k word list. The neural network language model alone achieves a perplexity of 103.0 which is only a 4% relative reduction with respect to the back-off model (perplexity = 107.4, see Table 7). If this neural network model is interpolated with the back-off model trained on the entire training set, the perplexity decreases from 70.2 to 67.6. The interpolation coefficient for the neural network is 0.28. Despite this small improvement in perplexity a notable word error reduction was obtained from 14.24% to 14.02%, with the lattice rescoring taking less than 0.05xRT. In the next sections, it is shown that larger improvements can be obtained by training the neural network on more data. The following experiments were all performed with the real-time system.

3.3.3. Adding selected data

Training the neural network language model with stochastic back-propagation on all of the available text corpora would take quite a long time. The estimated time for one training epoch with the 88 M words of commercial transcriptions is 58 h, and more than 12 days if all the 508 M words of newspaper texts were used. This is of course not very practical. One solution to this problem is to select a subset of the data that seems to be most useful for the task. This was done by selecting six months of the commercial transcriptions that minimize the perplexity on the development set. This gives a total of 22 M words and the training time is about 14 h per epoch. One can ask if the capacity of the neural network should be augmented in order to deal with the increased number of examples. Experiments with hidden layer sizes from 400 to 1000 neurons were performed (see Table 8).

Although there is a small decrease in perplexity and word error when increasing the dimension of the hidden layer, this is at the expense of a higher processing time. The training and recognition time is in fact almost linear with respect to the size of the hidden layer. An alternative approach to augment the capacity of the

Table 8
Performance of the neural network LM and training time per epoch as a function of the size of the hidden layer (1xRT system, fixed six months subset of commercial transcripts)

Hidden layer size	400	500	600	1000[a]
Training time per epoch	11 h20	13 h54	16 h15	11 h + 16 h
Perplexity NN LM alone	100.5	100.1	99.5	94.5
Perplexity hybrid LM	68.3	68.3	68.2	68.0
Word error rate hybrid LM	13.99%	13.97%	13.96%	13.92%

[a] Interpolation of networks with 400 and 600 hidden units.

Table 9
Performance of the neural network LM and training time per epoch as a function of the dimension of the projection layer (1xRT system, fixed six months subset of commercial transcripts)

Projection layer dim.	50	60	70	100	**120**	150
Training time/epoch	13 h54	13 h54	14 h21	14 h34	**14 h32**	14 h50
Perplexity NN LM	99.5	99.4	98.7	98.0	**97.5**	97.7
Perplexity hybrid LM	68.2	68.2	68.1	68.0	**67.9**	67.9
Word error rate hybrid LM	13.97%	13.97%	13.98%	13.92%	**13.88%**	13.91%

neural network is to modify the dimension of the continuous representation of the words. The idea behind this is that the probability estimation may be easier in a higher dimensional space (instead of augmenting the capacity of the non-linear probability estimator itself). This is similar in spirit to the theory behind support vector machines (Vapnik, 1998).

Increasing the dimension of the projection layer has several advantages as can be seen from the Table 9. First, the dimension of the continuous word representation has only a small effect on the training complexity of the neural network since most of the calculations are done to propagate and learn the connections between the hidden and the output layer (see Eq. (12)).[6] Second, perplexities and word error rates are slightly lower than those obtained when the size of the hidden layer is increased. The best result was obtained with a 120 dimensional continuous word representation. The perplexity is 67.9 after interpolation with the back-off language model and the word error rate is 13.88%. Finally, convergence is faster: the best result is obtained after about 15 epochs while up to 40 are needed with large hidden layers.

3.3.4. Training on all available data

In this section the random sampling algorithm described in Section 2.4 is used. We chose to use the full corpus of transcriptions of the acoustic data as this is the most appropriate data for the task. Experiments with different random subsets of the commercial transcriptions and the newspaper texts were performed (see Figs. 6 and 7). In all cases the same neural network architecture was used, i.e., a 120 dimensional continuous word representation and 500 hidden units. Some experiments with larger hidden units showed basically the same convergence behavior. The learning rate was again set to 5×10^{-3}, but with a slower exponential decay.

First of all, it can be seen that the results are better when using random subsets instead of a fixed selection of six months. Since the number of examples in one epoch depends on the parameters of the resampling algorithm, Fig. 6 shows the perplexity as a function of the number of examples seen during learning. The best results were obtained when resampling 10% of the commercial transcriptions. The perplexity is 67.1 after interpolation with the back-off language model and the word error rate is 13.82% (see summary in Table 10). Larger subsets of the commercial transcriptions lead to slower training, but do not give better results.

Encouraged by these results, we also included the 508 M words of newspaper texts in the training data. The sizes of the resampled subsets were chosen in order to use between 4 and 9 M words of each corpus. Fig. 7 summarizes the results. Using all the data gives a clear improvement in perplexity, but the particular values of the resampling coefficients do not appear to be important: the perplexities of the neural network language

[6] It is difficult to relate the measured run-time directly to the theoretical number of floating point operations since the performance depends also on cache size issues that seem to be more favorable for certain matrix sizes.

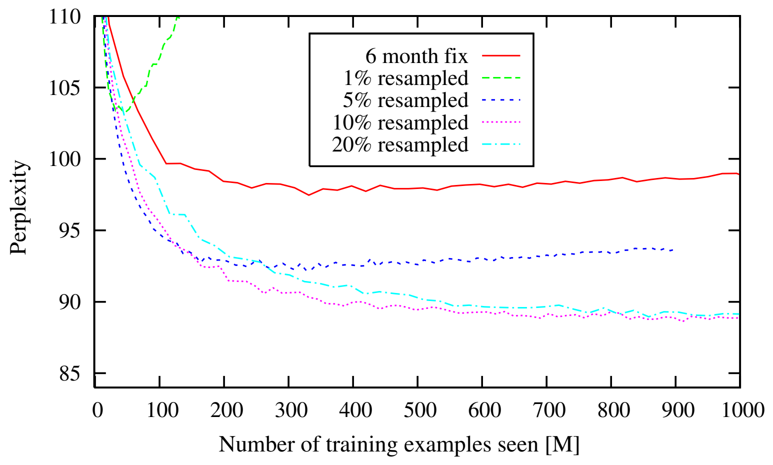

Fig. 6. Perplexity of the neural network LM alone when resampling different random subsets of the commercial transcriptions (65 k word list).

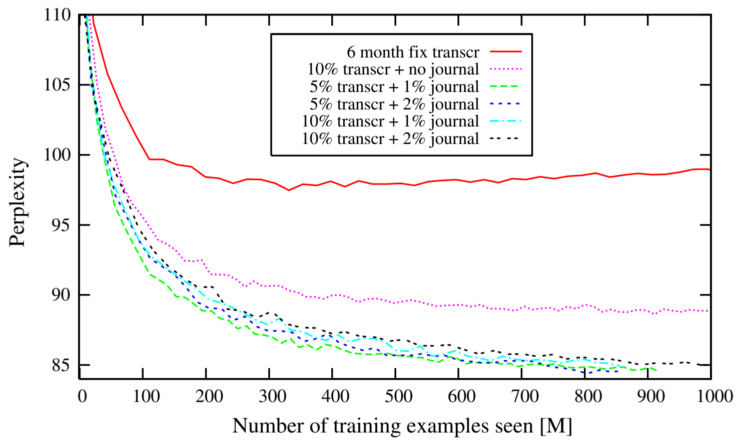

Fig. 7. Perplexity of the neural network LM alone when resampling different random subsets of the commercial transcriptions and the newspaper texts (65 k word list).

Table 10
Comparison of the back-off with the neural network LM using different amounts of training data (1xRT system)

Training data	Back-off	Neural network LM				
	600 M	4 M	22 M	92.5 M[a]	600 M[a]	
Training time						
Per epoch	–	2 h40	14 h	9 h40	12 h	3×12 h
Total (max)	–	80 h	250 h	500 h	500 h	3×500 h
Perplexity						
NN LM alone	–	103.0	97.5	84.0	80.5	76.5
Hybrid LM	70.2	67.6	67.9	66.7	66.5	65.9
Coefficient of NN	–	0.28	0.26	0.37	0.40	3×0.16
Word error rate						
Hybrid LM	**14.24**%	14.02%	13.88%	13.81%	**13.75**%	**13.61**%

The perplexities are given for the neural network LM alone and interpolated with the back-off LM trained on all the data (hybrid LM). The last column corresponds to three interpolated neural networks.
[a] By resampling different random parts at the beginning of each epoch.

models alone converge to a value of about 85. We chose to resample 10% of the commercial transcriptions and 1% of the newspaper texts. The neural network trained on the subset of 22 M words of the commercial transcripts converges after about 16 epochs, i.e., after having seen 350 M examples. The neural networks trained on all the data need about twice as much time (about 700 M examples). This means that in average each example of the large corpus was seen only once. However, in practice training can be stopped much earlier, i.e., after having seen 200–300 M examples, with a small impact on the word error rate (increase by about 0.05%).

Table 10 summarizes the results of the different neural network language models. It can be clearly seen that the perplexity of the neural network language model alone decreases significantly with the amount of training data used. The perplexity after interpolation with the back-off language model changes only by a small amount, but there is a notable improvement in the word error rate. This is more experimental evidence that the perplexity of a language model is not directly related to the word error rate.

The best neural network language model achieves a word error reduction of 0.5% absolute with respect to the carefully tuned back-off model. The additional processing time needed to rescore the lattices is less than 0.05xRT. This is a significant improvement, in particular for a fast real-time continuous speech recognition system. When more processing time is available a word error rate of 13.61% can be achieved by using three neural networks together (in 0.14xRT). If we do not interpolate these neural networks with the back-off model, the perplexity is 76.5 and the word error rate 14.30%. This means that it is not interesting to use the neural network alone for this task.

The experimental results were also validated using the 7xRT speech recognizer. The word error rate of the reference system using a back-off model is 10.74%. This can be reduced to 10.51% when interpolating the back-off with a neural network model trained on the fine transcriptions only and to 10.20% when the neural network model is trained on all data using the resampling algorithm. The interpolation coefficient for the back-off model is 0.53.

3.4. English and Spanish parliament speeches

The European project Tc-Star is concerned with speech-to-speech translation of European parliament speeches.[7] The main focus is on the translation directions European English to Spanish and vice versa. This section summarizes the efforts undertaken to build language models for the speech recognition systems for these languages. In both cases, the main source for language model training are the transcripts of the acoustic training data (about 700 k words) and the official translations of parliament speeches as distributed by the European Community (about 34 M words per language). Both systems achieved a very good ranking in the 2006 Tc-Star international evaluation (Lamel et al., 2006). We report here the results on the development and test set of this evaluation.

3.4.1. European English system

The speech recognizer for the English parliament speeches has a similar architecture to the English broadcast news system. The incorporation of the neural network language model was again performed by rescoring the final lattices. The four-gram back-off model was trained on 688 k words of transcribed audio data, 33.8 M words of English parliament speeches and a total of 473 M words of broadcast news data. The neural network language model was trained on the same amount of data using the resampling algorithm (with weights 1.0, 1.0 and 0.01). A short-list of 4096 words was used, giving a coverage of 82.6% on the development data and of 88.7% when rescoring lattices. The vocabulary has 60 k words, resulting in an OOV rate on the development data of 0.28%.

The experiments on the French broadcast news system showed that it is advantageous to augment the dimension of the projection layer instead of the size of the hidden layer in order to increase the capacity of the neural network. Here this idea is pushed even further and four neural networks are trained independently and interpolated together. The dimensions of the projection layers were 200, 250, 300 and 350. This approach is related to Bagging (Breiman, 1994) and random class language models (Emami and Jelinek, 2005). Bagging aims to improve the generalization performance by combining classifiers trained on bootstrap replicates of the

[7] *Technology and Corpora for Speech to Speech Translation*, <http://www.tc-star.org>.

Table 11
Recognition results for English parliament speeches on the development and evaluation data

LM training data	Back-off LM	Neural LM alone	Hybrid LM
	507 M	507 M	507 M
Perplexity, Dev.	106.4	92.5	90.2
Word error rate, Dev.	10.84%	10.07%	9.95%
Word error rate, Eval.	9.02%	8.45%	8.18%
Lattice rescoring time	–	0.11xRT	0.15xRT

Table 12
Recognition results for Spanish parliament speeches on the development and evaluation data

LM training data	Back-off LM	Neural LM alone	Hybrid LM
	84 M	84 M	84 M
Perplexity, Dev.	87.7	81.0	78.8
Word error rate, Dev.	10.64%	10.09	10.00%
Word error rate, Eval.	11.17%	10.66%	10.59%
Lattice rescoring time	–	0.25xRT	0.28xRT

training data. In random class language models, randomization can be obtained by varying the number of classes. In our model the dimension of the projection layer is varied. Each of the four neural network achieves a perplexity of about 103 on the development set, which decreases to 92.5 when they are interpolated together. The initial learning rate was set to 5×10^{-3} and the weight decay factor to 3×10^{-5}. All the results are summarized in Table 11.

A relative perplexity reduction of 15% was obtained ($106.4 \rightarrow 90.2$) and the word error rate on the development data improved by as much as 0.89% absolute ($10.84 \rightarrow 9.95\%$). This is a significant improvement at this level of performance of the speech recognition system. Since the neural network language model was trained on the same data as the back-off model, there is only a small gain when both are interpolated (the coefficient of the neural network is 0.72). Our approach also showed a good generalization behavior: on the evaluation data the same absolute word error reduction was achieved (when interpolation was used).

3.4.2. Spanish system

The speech recognizer for the Spanish parliament speeches has the same structure as the English system. Data used for the language models are the transcriptions of the audio data (788 k words), the translated European Parliament speeches (36.1 M words) and proceedings of the Spanish Parliament (47.1 M words). The neural network was trained on all this data, using the resampling algorithm (weights 1.0, 0.2 and 0.2). The vocabulary consists of 64 k words and the OOV rate on the development data is 0.64%. A short-list of 2000 words was used, giving a coverage of 78.3% on the development data and 80.0% when rescoring lattices. Four neural networks were trained and interpolated together. The dimensions of the projection layers were 150, 170, 200 and 250. The other parameters of the neural networks are the same as for the English system. Table 12 summarizes the results.

The neural network language model achieves a relative improvement in perplexity of 10% and a word error reduction of 0.64% absolute. This gain is smaller than the one obtained on the English system. This may be explained by the smaller coverage during lattice rescoring and by the fact that the lattices themselves are smaller: for Spanish they have 307 arcs and 550 links in average, while there are 357 arcs and 698 links for the English system. Also, the gain obtained by interpolating the neural network and the back-off language model is quite small (coefficient of the back-off model is 0.24).

3.5. English lectures and meetings

In this section the design of language models for a speech recognizer for English lectures and seminars is investigated. This is the most challenging task considered in this paper with respect to several aspects. The

Table 13
Result summary for English lectures and meetings on the development data

LM training data	Back-off LM	Neural LM alone	Hybrid LM
	84 M	84 M	84 M
Perplexity	122.6	117.3	107.8
Word error rate	23.5%	23.2%	22.6%

research described here was performed in the context of the European FP6 Integrated Project Chil which is exploring new paradigms for human–computer interaction.[8]

The acoustic training material consisted of the Chil (6.2 h), Isl (10.3 h), Icsi (60 h) and Nist (17.2 h) meeting corpora and the Ted corpus (9.3 h), recordings of presentations made mainly by non-native speakers of English at the Eurospeech conference in Berlin 1993. See Lamel et al. (2005) and references therein for a detailed description of these corpora. Transcribing lectures and meetings is much more complicated than broadcast news recognition. Also, there is significantly less training data available for acoustic and language modeling in comparison to our work on conversational speech. The baseline back-off language models were trained on the transcriptions of the audio data (1.4 M words) and 20 thousand articles from speech-related workshops and conferences (46.6 M words). The use of transcriptions of conversational telephone speech or even broadcast news was not very useful. The word list has 58 k words and the OOV rate is 0.46% on the chil 2006 development data (29 k words).

The resampling algorithm was used to train the neural network on the transcriptions (coefficient 1.0) and the proceedings (coefficient 0.2). A short-list of 4096 words was used, giving a coverage of 77.1% on the development data and 87.1% when rescoring lattices. As before, several neural networks were trained and interpolated together, varying the dimension of the projection layer between 120 and 250. In the hybrid language model, i.e., the interpolation of the back-off and the neural language model, the back-off language model has a coefficient of 0.38. The hidden layer of all neural networks was of size 500. On this task, the hybrid language model achieves an absolute word error reduction of 0.9% (see Table 13).

4. Conclusion

In this paper, we have described the theory and an experimental evaluation of a new approach to language modeling for large vocabulary continuous speech recognition. The principal idea, based on work of Bengio and Ducharme (2001), is to project the words onto a continuous space and to perform the probability estimation in this space. An important aspect of the success of the new method are the fast algorithms for training and recognition. Lattice rescoring takes usually less than 0.1xRT so that the neural network language model can be used in a real-time speech recognizer. The described training algorithms are fast enough to train the neural network on several million words in a reasonable amount of time. The system was trained on text corpora of up to 600 million words using a resampling algorithm.

The necessary capacity of the neural network is also an important issue. Three possibilities were explored: increasing the size of the hidden layer, training several networks and interpolating them together, and using large projection layers. Increasing the size of the hidden layer gave only modest improvements in word error, at the price of very long training times. In this respect, the second solution is more interesting as the networks can be trained in parallel. Large projection layers appear to be the best choice as this has little impact on the complexity during training or recognition.

The neural network language model was thoroughly evaluated on several speech recognition tasks and languages. This has made it possible to cover different speaking styles, ranging from rather well formed speech with few errors (broadcast news) to very relaxed speaking with many errors in syntax and semantics (meetings and conversations). The three covered languages also exhibit particular properties, French has for example a large number of homophones in the inflected forms that can only be distinguished by a language model. Usually, the neural network language model is interpolated with a carefully optimized baseline back-off four-gram

[8] Computers in the Human Communication Loop, <http://isl.ira.uka.de/chil>.

model. In this way, significant reductions in the word error rate of almost 1% absolute were obtained. It is worthwhile to recall that the baseline speech recognizers were all state-of-the-art as demonstrated by their good rankings in international evaluations. It seems therefore reasonable to say that the combination of the developed neural network and a back-off language model can be considered as a serious alternative to the commonly used back-off language models alone. Despite these good results, we believe that the full potential of language modeling in the continuous domain is far from being attained. In the following discussion we suggest future directions for research.

Today, unsupervised acoustic model adaptation to the speaker and to the acoustic conditions is common practice and usually important reductions in the word error rate are observed. Unsupervised adaptation of the language model has also attracted much attention in the literature, but with far less success than acoustic model adaptation, at least when used in a large vocabulary continuous speech recognizer. The major problem is probably the mismatch between the amount of available adaptation data (several hundred words) and the number of parameters of the language model that need to be adapted (thousands to millions). Among others, the proposed methods include cache language models (Kuhn and de Mori, 1990), adaptive interpolation of several language models (Kneser and Steinbiss, 1993), maximum a posteriori (Federico, 1996) or minimum discrimination information (Federico, 1999), and adaptation methods based on information retrieval methods (Chen et al., 2001). An interesting summary can also be found in Bellegarda (2000).

The continuous representation of the words in the neural network language model offers new ways to perform constrained language model adaptation. For example, the continuous representation of the words can be changed so that the language model predictions are improved on some adaptation data, e.g., by moving some words closer together which appear often in similar contexts. The idea is to apply a transformation on the continuous representation of the words by adding an adaptation layer between the projection layer and the hidden layer. This layer is initialized with the identity transformation and then learned by training the neural network on the adaptation data. Several variants of this basic idea are possible, for example using shared block-wise transformations in order to reduce the number of free parameters. Note that the continuous representation of all words is transformed, even when they do not appear in the adaptation data. It may be necessary to use several adaptation layers in order to achieve a highly non-linear transformation.

Other interesting extensions of the neural network language model concern the resampling algorithm. We are going to investigate more sophisticated ways to select different subsets from the large corpus at each epoch (instead of a random choice). One possibility would be to use active learning, i.e., focusing on examples that are most useful to decrease the perplexity. One could also imagine associating a probability to each training example and using these probabilities to weight the random sampling. These probabilities would be updated after each epoch. This is obviously similar to boosting techniques which build sequentially classifiers that focus on examples wrongly classified by the preceding one.

Finally, we propose to investigate new training criteria for the neural network language model. Language models are almost exclusively trained independently from the acoustic model by minimizing the perplexity on some development data, and it is well known that improvements in perplexity do not necessarily lead to reductions in the word error rate. Unfortunately, it is quite difficult to use other criteria than perplexity with back-off language models, although several attempts have been reported in the literature, for example (Ito et al., 1999; Chen et al., 2000; Kuoi et al., 2002). The neural network language model on the other hand, uses a cost function that is minimized during training. We will explore a variety of new cost functions that are well related to the expected word error rate. Another interesting direction is to train the language model on output produced by the speech recognizer instead of on raw texts. A particularly interesting candidate are consensus networks since they actually include competing words. The goal would be to train a model to discriminate between the words in a consensus block so as to minimize the expected word error rate.

The underlying idea of the continuous space language model described here is to perform the probability estimation in a continuous space. Although only neural networks were investigated in this work, the approach is not inherently limited to this type of probability estimator. Other promising candidates include Gaussian mixture models and radial basis function networks. These models are interesting since they can be more easily trained on large amounts of data than neural networks, and the limitation of a short-list at the output may not be necessary. The use of Gaussians makes it also possible to structure the model by sharing some Gaussians using statistical criteria or high-level knowledge. On the other hand, Gaussian mixture models are a non-discriminative approach. Comparing them with neural networks could provide additional insight into the success of the neural network language model: is it the continuous representation or rather the discriminative probability estimation that is the key to improving upon the traditional back-off language models?

Acknowledgements

The author would like to recognize the contributions of G. Adda, E. Bilinski, F. Lefèvre, O. Galibert, J.-L. Gauvain and L. Lamel for their involvement in the development of the LIMSI speech recognition systems on top of which this current work is based. The author would also like to thank Yoshua Bengio for fruitful discussions and helpful comments.

References

Bellegarda, J., 1997. A latent semantic analysis framework in large-span language modeling. In: Proceedings of Eurospeech. pp. 1451–1454.

Bellegarda, J., 2000. Exploiting latent semantic information in statistical language modeling. Proceedings of IEEE 88 (8), 1279–1296.

Bengio, Y., Ducharme, R., 2001. A neural probabilistic language model. In: Proceedings of Advances in Neural Information Processing Systems, vol. 13. pp. 932–938.

Bengio, Y., Sénécal, J.-S., 2003. Quick training of probabilistic neural nets by importance sampling. In: Proceedings of AISTATS Conference.

Bengio, Y., Ducharme, R., Vincent, P., 2000. A neural probabilistic language model. Technical Report 1178, Département d'informatique et recherche opérationnelle, Université de Montréal.

Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C., 2003. A neural probabilistic language model. Journal of Machine Learning Research 3 (2), 1137–1155.

Berger, A., Della Pietra, S., Della Pietra, V.J., 1996. A maximum entropy approach to natural language processing. Computational Linguistics 22, 39–71.

Bilmes, J., Asanovic, K., Chin, C., Demmel, J., 1997. Using phipac to speed error back-propagation learning. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing. pp. V:4153–4156.

Bishop, C., 1995. Neural Networks for Pattern Recognition. Clarendon Press, Oxford.

Breiman, L., 1994. Bagging predictors. Machine Learning 24 (2), 123–140.

Bridle, J.S., 1990. Probabilistic interpretation of feedforward classification network outputs, with relationship to statistical pattern recognition. In: Neurocomputing: Algorithms, Architectures and Applicationss. Springer-Verlag, pp. 227–236.

Brown, P., Cocke, J., Della Pietra, S., Della Pietra, V.J., Jelinek, F., Lafferty, J., Mercer, R., Roossin, P., 1990. A statistical approach to machine translation. Computational Linguistics 16 (2), 79–85.

Brown, P.F., Della Pietra, V.J., deSouza, P.V., Lai, J.C., Mercer, R.L., 1992. Class-based *n*-gram models of natural language. Computational Linguistics 18 (4), 467–470.

Bulyko, I., Ostendorf, M., Stolcke, A., 2003. Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures. In: Proceedings of Human Language Technology Conference. pp. 7–9.

Castro, M.J., Polvoreda, V., 2001. Connectionist *n*-gram models by using MLPs. In: Proceedings of Workshop on Natural Language Processing and Neural Networks (Tokyo).

Castro, M.J., Prat, F., 2003. New directions in connectionist language modeling. In: Proceedings of International Work-conference on Artificial and Natural Neural Networks IWANN, Springer-Verlag LNCS, vol. 2686. pp. 598–605.

Chelba, C., Jelinek, F., 2000. Structured language modeling. Computer Speech and Language 13 (4), 283–332.

Chen, S.F., Goodman, J.T., 1999. An empirical study of smoothing techniques for language modeling. Computer Speech and Language 13 (4), 359–394.

Chen, Z., Lee, K.F., Li, M.J., 2000. Discriminative training on language model. In: Proceedings of International Conference on Speech and Language Processing. pp. I:493–496.

Chen, L., Gauvain, J.-L., Lamel, L., Adda, G., Adda, M., 2001. Using information retrieval methods for language model adaptation. In: Proceedings of Eurospeech. pp. 255–258.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R., 1990. Indexing by latent semantic analysis. Journal of the American Society for Information Science 41 (6), 391–407.

Elman, J., 1990. Finding structure in time. Cognitive Science 14, 179–211.

Emami, A., Jelinek, F., 2005. Random clusterings for language modeling. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing. pp. I:581–584.

Emami, A., Xu, P., Jelinek, F., 2003. Using a connectionist model in a syntactical based language model. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing. pp. I:372–375.

Federico, M., 1996. Bayesian estimation methods for *n*-gram language model adaptation. In: Proceedings of International Conference on Speech and Language Processing. pp. 240–243.

Federico, M., 1999. Efficient language model adaptation through mdi estimation. In: Proceedings of Eurospeech. pp. 1583–1586.

Fiscus, J.G., 1997. A post-processing system to yield reduced error word rates: Recognizer output voting error reduction (ROVER). In: Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding. pp. 347–354.

Galliano, S., Geoffrois, E., Mostefa, D., Choukri, K., Bonastre, J.-F., Gravier, G., 2005. The Ester phase II evaluation campaign for the rich transcription of French broadcast news. In: Proceedings of Eurospeech. pp. 1149–1152.

Gauvain, J.-L., Lamel, L., Adda, G., 2002. The LIMSI broadcast news transcription system. Speech Communication 37 (2), 98–108.

Gauvain, J.-L., Lamel, L., Schwenk, H., Adda, G., Chen, L., Lefèvre, F., 2003. Conversational telephone speech recognition. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing. pp. I:212–215.

Gauvain, J.-L., Adda, G., Adda-Decker, M., Allauzen, A., Gendner, V., Lamel, L., Schwenk, H., 2005. Where are we in transcribing BN French? In: Proceedings of Eurospeech. pp. 1665–1668.

Goodman, J.T., 2001. A bit of progress in language modeling. Computer Speech and Language 15, 403–434.

Hinton, G., 1986. Learning distributed representations of concepts. In: Proceedings of the Eighth Annual Conference of the Cognitive Science Society. Lawrence Erlbaum, Hillsdale, Amherst, pp. 1–12.

Intel Math Kernel Library, 2005. <http://www.intel.com/software/products/mkl/>.

Ito, A., Khoda, M., Ostendorf, M., 1999. A new metric for stochastic language model evaluation. In: Proceedings of Eurospeech. pp. 1591–1594.

Iyer, R., Ostendorf, M., 1999. Relevance weighting for combining multi-domain data for *n*-gram language modeling. Computer Speech and Language 13 (3), 267–282.

Katz, S.M., 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech, and Signal Processing 35 (3), 400–401.

Kimball, O., Kao, C.-L., Arvizo, T., Makhoul, J., Iyer, R., 2004. Quick transcription and automatic segmentation of the fisher conversational telephone speech corpus. In: Proceedings of 2004 Rich Transcriptions Workshop, Pallisades, NY.

Kneser, R., Steinbiss, V., 1993. On the dynamic adaptation of stochastic language modeling. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing. pp. 586–589.

Kuhn, R., de Mori, R., 1990. A cache-based natural language model for speech reproduction. IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (6), 570–583.

Kuoi, H.-K.J., Fosler-Lussier, E., Jiang, H., Lee, C.-H., 2002. Discriminative training of language models for speech recognition. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing. pp. I:325–328.

Lamel, L., Adda, G., Bilinski, E., Gauvain, J.-L., 2005. Transcribing lectures and seminars. In: Proceedings of Eurospeech. pp. 1657–1660.

Lamel, L., Gauvain, J.-L., Adda, G., Barras, C., Bilinski, E., Galibert, O., Pujol, A., Schwenk, H., Zhu, X., 2006. The LIMSI 2006 TC-STAR transcription systems. In: Proceedings of TC-STAR Speech to Speech Translation Workshop, Barcelona. pp. 123–128.

Miikkulainen, R., Dyer, M.G., 1991. Natural language processing with modular PDP networks and distributed lexicon. Cognitive Science 15, 343–399.

Morin, F., Bengio, Y., 2005. Hierarchical probabilistic neural network language model. In: Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics.

Nakamura, M., Shikano, K., 1989. A study of english word category prediction based on neural networks. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing. pp. 731–734.

Nguyen, L., Abdou, S., Afify, M., Makhoul, J., Matsoukas, S., Schwartz, R., Xiang, B., Lamel, L., Gauvain, J.-L., Adda, G., Schwenk, H., Lefevre, F., 2004. The 2004 BBN/LIMSI 10xRT english broadcast news transcription system. In: Proceedings of 2004 Rich Transcriptions Workshop, Pallisades, NY.

Paccanaro, A., Hinton, G., 2000. Extracting distributed representations of concepts and relations from positive and negative propositions. In: Proceedings of IEEE joint conference on neural networks. pp. 2259–2264.

Prasad, R., Matsoukas, S., Kao, C.-L., Ma, J., Xu, D.-X., Colthurst, T., Thattai, G., Kimball, O., Schwartz, R., Gauvain, J.-L., Lamel, L., Schwenk, H., Adda, G., Lefevre, F., 2004. The 2004 20xRT BBN/LIMSI english conversational telephone speech system. In: Proceedings of 2004 Rich Transcriptions Workshop, Pallisades, NY.

Rosenfeld, R., 1996. A maximum entropy approach to adaptive statistical language modeling. Computer Speech and Language 10 (3), 187–228.

Schmidhuber, J., 1996. Sequential neural text compression. IEEE Transactions on Neural Networks 7 (1), 142–146.

Schwenk, H., 2001. Language modeling in the continuous domain. Technical Report 2001-20, LIMSI-CNRS, Orsay, France.

Schwenk, H., 2004. Efficient training of large neural networks for language modeling. In: Proceedings of IEEE joint conference on neural networks. pp. 3059–3062.

Schwenk, H., Gauvain, J.-L., 2002. Connectionist language modeling for large vocabulary continuous speech recognition. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing. pp. I: 765–768.

Schwenk, H., Gauvain, J.-L., April 2003. Using Continuous Space Language Models for Conversational Speech Recognition. In: Proceedings of ISCA and IEEE Workshop on Spontaneous Speech Processing and Recognition. Tokyo, pp. 49–53.

Schwenk, H., Gauvain, J.-L., 2004a. Neural network language models for conversational speech recognition. In: Proceedings of International Conference on Speech and Language Processing. pp. 1215–1218.

Schwenk, H., Gauvain, J.-L., 2004b. Using neural network language models for LVCSR. In: Proceedings of 2004 Rich Transcriptions Workshop, Pallisades, NY.

Schwenk, H., Gauvain, J.-L., 2005a. Building continuous space language models for transcribing european languages. In: Proceedings of Eurospeech. pp. 737–740.

Schwenk, H., Gauvain, J.-L., 2005b. Training neural network language models on very large corpora. In: Proceedings of Empirical Methods in Natural Language Processing. pp. 201–208.

Schwenk, H., Déchelotte, D., Gauvain, J.-L., 2006. Continuous space language models for statistical machine translation. In: Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions. pp. 723–730.

Stolcke, A., 2002. SRILM - an extensible language modeling toolkit. In: Proceedings of International Conference on Speech and Language Processing. pp. II: 901–904.

Vapnik, V., 1998. Statistical Learning Theory. Wiley, New York.

Wang, W., Stolcke, A., Harper, M., 2004. The use of a linguistically motivated language model in conversational speech recognition. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing. pp. 261–264.

Xu, P., Mangu, L., 2005. Using random forest language models in the IBM RT-04 CTS system. In: Proceedings of Eurospeech. pp. 741–744.

Xu, W., Rudnicky, A., 2000. Can artificial neural networks learn language models? In: Proceedings of International Conference on Speech and Language Processing. pp. I:202–205.