On the Use of Continuous Space Neural Network Language Models

Holger Schwenk,[*] **Abdelkhalek Messaoudi, Jun Luo** and **Jean-Luc Gauvain**
Spoken Language Processing Group
CNRS-LIMSI, BP 133
91403 Orsay cedex, France
{schwenk,abdel,luo,gauvain}@limsi.fr

Abstract

This paper reports on using continuous space language models for large vocabulary speech transcription systems. This approach is used to overcome the data sparseness problem which always arises for STT systems as there are never enough speech transcriptions to build representative large N-gram models. The architecture of these models is described, along with the methods that were deployed to make model training and decoding effective for an LVCSR system. Experimental results report significant word error reductions on the GALE data for Arabic and Mandarin on top of a well-tuned STT system. Some machine translation results are also reported to demonstrate the effectiveness of the approach.

1 Introduction

During the last years there has been growing interest in using neural networks for language modeling. Instead of relying on a back-off component, the neural network approach attempts to overcome the data sparseness problem by performing the estimation in a continuous space. In early work, this type of language model was mostly used for tasks for which only a very limited amount of in-domain training data is available. In the context of the EARS and GALE programs new algorithms were developed to train a neural network language model on very large text corpora. This made possible the use of the approach in domains where several hundreds of millions words of texts are available. The neural network language model has been evaluated in a state-of-the-art continuous speech recognizers for several languages and tasks. Word error reductions on the order of 0.5% to 1% absolute have been obtained.

The basic idea of the neural network LM, also called continuous space LM, is to project the word indices onto a continuous space and to use a probability estimator operating on this space (Bengio and Ducharme, 2001). Since the resulting probability functions are smooth functions of the word representation, better generalization to unknown n-grams can be expected. A neural network can be used to simultaneously learn the projection of the words onto the continuous space and to estimate the n-gram probabilities. This is still a n-gram approach, but the LM posterior probabilities are interpolated for any possible context of length n-1 instead of backing-off to shorter contexts. The architecture of this approach is described in detail in section 2.

Until today, this new language modeling method has been applied to large vocabulary speech recognition in six different languages (English, French, Spanish, Arabic, Mandarin and Dutch) achieving systematically consistent improvements in the word error rate. We describe here the work performed in the framework of the GALE project to use this technology in our Arabic and Mandarin speech recognition systems. In addition, the continuous space language model was used in a statistical machine translation system based on the open-source Moses decoder. Improvements of about 1 point BLEU were observed in in a large Arabic/English translation systems (Schwenk and Estève, 2008).

[*]Now with LIUM, University of Le Mans, France, Holger.Schwenk@lium.univ-lemans.fr

2 Architecture of the continuous space LM

The basic idea of the continuous space LM (CSLM) is to project the word indices onto a continuous space and to use a probability estimator operating on this space (Bengio and Ducharme, 2001). Since the resulting probability functions are smooth functions of the word representation, better generalization to unknown n-grams can be expected. A neural network can be used to simultaneously learn the projection of the words onto the continuous space and to estimate the n-gram probabilities. This is still a n-gram approach, but the LM posterior probabilities are "interpolated" for any possible context of length n-1 instead of backing-off to shorter contexts.

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

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

where N is the size of the vocabulary. The input uses the so-called 1-of-n coding, i.e., the *i-th* word of the vocabulary is coded by setting the *i-th* element of the vector to 1 and all the other elements to 0. The *i-th* line of the $N \times P$ dimensional projection matrix corresponds to the continuous representation of the i-th word. Let us denote c_k these projections, d_j the hidden layer activities, o_i the outputs, p_i their softmax normalization, and m_{jl}, b_j, v_{ij} and k_i the hidden and output layer weights and the corresponding biases. Using these notations the neural network performs the following operations:

$$d_j = \tanh\left(\sum_l m_{jl}\, c_l + b_j\right) \tag{2}$$

$$o_i = \sum_j v_{ij}\, d_j + k_i \tag{3}$$

$$p_i = e^{o_i} \;/\; \sum_{k=1}^{N} e^{o_k} \tag{4}$$

The value of the output neuron p_i corresponds directly to the probability $P(w_j = i|h_j)$. Training is performed with the standard back-propagation algorithm minimizing the following error function:

$$E = \sum_{i=1}^{N} t_i \, log \, p_i + \beta(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2) \tag{5}$$

where t_i denotes the desired output, 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 this equation is the cross-entropy between the output and the target probability distributions, and the second part is a regularization term that aims to prevent the neural network from overfitting the training data (weight decay). The parameter β has to be determined experimentally.

It can be shown that the outputs of a neural network trained in this manner converge to the posterior probabilities. Therefore, the neural network directly minimizes the perplexity on the training data. Note also that the gradient is back-propagated through the projection-layer, which means that the neural network learns the projection of the words onto the continuous space that is best for the probability estimation task.

This basic architecture has a quite high complexity. The following methods were deployed resulting in processing times of less than 0.1xRT:

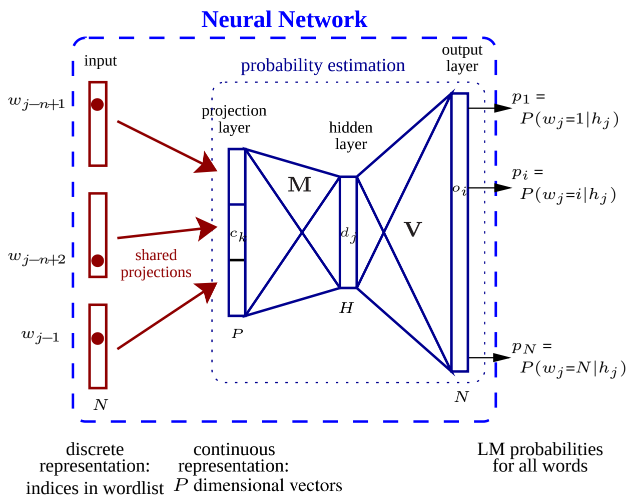

Figure 1: Architecture of the continuous space language model. h_j denotes the context $w_{j-n+1}, ..., w_{j-1}$. P is the size of one projection and H and N is the size of the hidden and output layer respectively. When shortlists are used the size of the output layer is much smaller then the size of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

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

2. *Lattice rescoring*: speech recognition is done with a standard back-off LM and a word lattice is generated. The CSLM is then used to rescore the lattice.

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

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

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

6. *resampling*: instead of cycling sequentially through all the data, we resample examples from the large corpora.

The idea behind shortlists is to use the neural network only to predict the s most frequent words, $s \ll |V|$, reducing by these means drastically the complexity. All words of the word list are still considered at the input of the neural network. The LM probabilities of words in the shortlist ($\hat{P}_N$) are calculated by the neural network and the LM probabilities of the remaining words ($\hat{P}_B$) are obtained from a standard 4-gram back-off LM:

$$\hat{P}(w_t|h_t) = \begin{cases} \hat{P}_N(w_t|h_t) P_S(h_t) & \text{if } w_t \in \text{shortlist} \\ \hat{P}_B(w_t|h_t) & \text{else} \end{cases} \quad (6)$$

$$P_S(h_t) = \sum_{w \in shortlist(h_t)} \hat{P}_B(w|h_t) \quad (7)$$

It can be considered that the neural network redistributes the probability mass of all the words in the shortlist. This probability mass is precalculated and stored in the data structures of the back-off LM. A back-off technique is used if the probability mass for a requested input context is not directly available. A detailed description of the speed-up techniques can be found in (Schwenk, 2004; Schwenk and Gauvain, 2005b).

Although some fast training algorithms were already proposed (Schwenk, 2004) , using a standard fully connected multi-layer perceptron, training on very large corpora can be time consuming. Therefore, an algorithm was proposed in (Schwenk and Gauvain, 2005b) to enable training the neural network on any arbitrarily large corpus. The basic idea is quite simple and consists of selecting small random subsets of the data for each epoch rather than using all the available data. This procedure has the advantages that there is no limit on the amount of training data; that after several epochs, most of the examples have been seen at least once; and that changing the examples after each epoch may potentially increase the generalization performance. This procedure has been used with more than 3 billion words of training data.

3 Experimental evaluation in ASR

Word error reductions using a continuous space LM for LVCSR were first reported on the EARS conversational telephone speech recognition task (Schwenk and Gauvain, 2002). Since then significant improvements have been reported on many speech recognition tasks and languages. Here results are reported on GALE broadcast data for both Arabic and Mandarin.

Continuous space language models were trained using a 290k word list for Arabic and a 56k word list for Mandarin. The short-list was of size 12k for Arabic and 8k for Mandarin. A traditional 4-gram language model was trained on all the available data for each language and pruned with high pruning threshold, and used as a short-list back-off model to calculate the probability masses and to cover the probabilities of words not included in the short-list.

Since training the neural network is quite long, the text data was sampled according to the size of the corpus and its importance. For Arabic, the manual transcriptions of the audio data and the web transcripts are used in their entirety. Across the text sources a total of about 26 million contexts are projected, and each training iteration takes about a day. Table 1 shows the approximate number of contexts by Arabic text source for a training iteration.

Text source	*#contexts*
trans+webstrans	16.6 M
nhr	2.0 M
albayan	1.5 M
hyt	1.5 M
aljaz	1.0 M
addustour	0.4 M
raya	0.4 M
levantine	0.3 M
ahram	0.3 M
akhbar	0.3 M
xin	0.3 M
al-watan	0.2 M
asb	0.1 M

Table 1: Number of contexts by Arabic text source for a CSLM training iteration.

Several neural networks were trained using different projection sizes, 200, 220 and 250. The evolution of the perplexity on the training and the full development data set (dev06+dev07) with a network having a projection size of 200 is given in Table 2. The first few iterations result in a large decrease in perplexity of the training data, and to a lesser degree on the dev data. After 8 or 9 iterations there is still a small gain on the training data, but this is not always the case for the dev data, for which the change in perplexity is seen to fluctuate.

Iteration	*Train*	*delta*	*Dev06+Dev07*	*delta*
1	364.8	-	791.2	-
2	236.0	128.8	717.9	73.2
3	208.0	28.0	686.6	31.4
4	193.5	14.4	671.3	15.3
5	184.2	9.3	662.0	9.3
6	177.3	6.9	657.7	4.3
7	172.4	4.9	648.7	9.0
8	168.5	4.0	647.9	0.8
9	165.0	3.5	640.1	7.8
10	162.2	2.8	640.7	-0.7
11	159.8	2.4	638.3	2.4
12	157.7	2.1	638.0	0.4

Table 2: Perplexity and change in perplexity of the Arabic training and development data as a function of training iteration for the CSLM.

The Mandarin continuous space model was trained on only a portion of the available text data as listed in Table 3. As for Arabic, all of the available transcripts of audio data (17.3M words) were used as well as other sources based on their recency and relevancy for a total of 1.3 billion words. The resampling coefficients of each individual source is also given in the table. It is 100% for the audio transcripts, and varies for the other sources in order to have on the order of 2 million words each at each iteration. These can be seen to range from 0.6% to 5% of the corpus size. Three neural networks were trained, having 200/250/300 hidden layers respectively. The corresponding perplexities on the combined development data set (dev07+eval07+dev08) are given in Table 4. A standard 4-gram language model was formed by interpolating component LMs trained on 28 subcorpora. The perplexity on the combined dev07+eval07+dev08 data with this language model is 192.6. Interpolating this model (interpolation weight 0.484) with the three continuous space LMs using the weights shown in Table 4 reduces the perplexity of the development data to 171.8. This interpolated language model is what is referred to as the 4-gram CSLM.

Corpus	*Size*	*Resampling Coeff.& Size*	
audio transcripts	17.3M	1.000	17.3M
giga_xin	319.9M	0.006	1.92M
giga_cna_2	247.7M	0.008	1.98M
agile_bitex_ce	178.0M	0.010	1.78M
VOARFA	35.4M	0.056	1.98M
phoenixtv	88.0M	0.023	2.02M
ibm_sina	283.3M	0.007	1.98M
bbn_webdata	187.3M	0.010	1.87M

Table 3: The Mandarin text corpora used to train the continuous space language model, with the resampling coefficient and corpus sizes.

Language	*Hidden Layer Size*		
Model	200	250	300
CSLM	220.11	219.24	217.29
Interp. weight	0.164	0.169	0.183

Table 4: Perplexity of the combined dev07+eval07+dev08 data and interpolation weights of the 3 component CSLMs.

A two-pass decoding is used with PLP+MLP acoustic models (Fousek et al., 2008). The first pass generates a word lattice, followed by consensus decoding with 4-gram LM. This output is used for un-

System	train #words	LM size	Perplexity Dev06 all	Perplexity Dev06 Nwire	Perplexity Dev06 Web	BLEU Dev06 All	BLEU Dev06 NW	BLEU Dev06 Web	BLEU Eval08 All	BLEU Eval08 NW	BLEU Eval08 Web
Gigaword LM	3.4G	3.7G	128.1	104.7	206.5	44.40	47.27	34.90	42.13		
+ Google LM	169M 4-grams	5.5G	114.5	99.0	161.7	44.70	47.22	36.11	41.90	47.20	33.50
CSLM	3.4G	$\approx$ 1G	98.3	85.3	137.4	45.96	48.56	36.69	**42.98**	**48.30**	**34.31**

Table 6: Result summary for the baseline SMT system, adding parts of the Google n-grams and using the CSLM.

Arabic LM	*eval06*	*dev07*	*eval07*
4-gram	19.1	12.1	13.7
CSLM	18.4	11.6	13.0
Mandarin LM	*dev07*	*evrt07ns*	*dev08*
4-gram	11.3	10.3	10.5
CSLM	10.8	9.9	10.1

Table 5: Results comparing standard 4-gram backoff LM alone and interpolated with continuous space LMs on three data sets. Arabic WER (%) with automatic segmentation (top). Mandarin CER (%) with manual segmentation (bottom).

supervised acoustic model adaptation using CMLLR and MLLR, prior to a second decoding using the LM formed by interpolating the 4-gram LM with the 3 component CSLMs for the consensus decoding.

Table 5 reports recognition error rates for Arabic and Mandarin on three data sets. For Arabic, a WER reduction of 0.5 to 0.7% is obtained using the CSLM. The gain in Mandarin CER is a bit smaller, 0.4 to 0.5%. Continuous space models have been trained for other languages (French, Spanish, Dutch, English) and tasks (Schwenk, 2007). In all cases one or more CSLMs are interpolated with a standard 4-gram backoff language model and improvements of the same order of magnitude have been consistently obtained.

4 Experimental evaluation in SMT

The continuous space LM was also used in a state-of-the-art phrase-based statistical machine translation (SMT) system. It is well known that the LM on the target language plays a crucial role in phrase-based MT system. It helps choosing among different possible translations of a word or phrase, decides whether reordering of the words should be performed and assures of course the smoothness of the produced sentence. Here we report on results with the CSLM achieved by the first author in an Arabic/English system that participated in the 2008 NIST MT evaluation (Schwenk and Estève, 2008).

The translation model was trained on all provided data (about 165M words) and the baseline 4-gram LM on all the texts of the English LDC Gigaword corpus (more than 3.5G words). In addition, a large collection of 1 billion 5-grams were used, obtained from texts collected by Google on the Internet. It has been reported that significant improvements of the performance of an SMT system can be obtained using all this data (Brants et al., 2007), but this requires substantial hardware equipment that exceeded by far our possibilities. Therefore, we applied various steps of filtering with the goal to keep only the most important n-grams (about 139M 4-grams). The results of this SMT system are summarized in Table 6. Adding this "Google LM" reduced the perplexity by about 9% in average, but most of the improvement is obtained on the WEB part of Dev06, which is actually not surprising given the source of this data. The CSLM achieved an additional perplexity reduction of almost 14% on top of the large LM which includes the filtered Google data. The storage requirements of the CSLM are also lower since it use a distributed representation of the knowledge.

The Google LM brought an improvement of 0.3 BLEU on the Dev data, all the improvements being again concentrated to the WEB part. Unfortunately, it turned out to be not useful on the Eval08 test data. The CSLM achieved an improvement of 1.1 BLEU on the test data, on top of this heavily optimized system. This final system achieved a very good ranking in the 2008 NIST MT evaluation. The official results of all participants have been published by NIST.[1]

[1]`http://www.nist.gov/speech/tests/mt/2008/`

5 Conclusion

Continuous space language models were initially developed in order to improve the generalization behavior when only a limited amount of in-domain language model training data is available (Schwenk and Gauvain, 2002). They were in particular used in the frame work of the DARPA EARS project to improve the modeling of conversational language. We have continuously improved the initial approach (Schwenk, 2004; Schwenk and Gauvain, 2005b; Schwenk, 2007). Today it is successfully used in all large vocabulary speech recognition systems developed at the LIMSI, consistently obtaining reductions of the word error rate of up to 1% absolute. We have developed very efficient algorithms to train the system on billions of words of text and to deploy it in real-time speech recognition system (Schwenk and Gauvain, 2005a). In this work, we have reported results obtained in LIMSI's Arabic and Mandarin speech recognition systems that were used in the DARPA GALE program. Finally, the same technique was successfully ported to a large phrase-based Arabic/English statistical machine translation system.

Acknowledgments

This work was partially supported under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022, by the European Commission under the projects CORE-TEX, TC-STAR and CHIL, and in part by the French Government under the project INSTAR (ANR JCJC06_143038) and by OSEO under the Quaero program.

References

Yoshua Bengio and Rejean Ducharme. 2001. A neural probabilistic language model. In *NIPS*, volume 13, pages 932–938.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *EMNLP*, pages 858–867.

Petr Fousek, Lori Lamel, and Jean-Luc Gauvain. 2008. Transcribing Broadcast Data Using MLP Features. In *Interspeech*, pages 1433–1436.

Holger Schwenk and Yannick Estève. 2008. Data selection and smoothing in an open-source system for the 2008 NIST machine translation evaluation. In *Interspeech*, pages 2727–2730.

Holger Schwenk and Jean-Luc Gauvain. 2002. Connectionist Language Modeling for Large Vocabulary Continuous Speech Recognition. In *Proceedings of ICASSP*, pages 765–768.

Holger Schwenk and Jean-Luc Gauvain. 2005a. Building continuous space language models for transcribing european languages. In *Interspeech*, pages 737–740.

Holger Schwenk and Jean-Luc Gauvain. 2005b. Training neural network language models on very large corpora. In *EMNLP*, pages 201–208.

Holger Schwenk. 2004. Efficient training of large neural networks for language modeling. In *IJCNN*, pages 3059–3062.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21:492–518.