PBML

The Prague Bulletin of Mathematical Linguistics

NUMBER 93 JANUARY 2010 137–146

Continuous-Space Language Models for Statistical Machine Translation

Holger Schwenk

LIUM, University of Le Mans, France

Abstract

This paper describes an open-source implementation of the so-called continuous space language model and its application to statistical machine translation. The underlying idea of this approach is to attack the data sparseness problem by performing the language model probability estimation in a continuous space. The projection of the words and the probability estimation are both performed by a multi-layer neural network. This paper describes the theoretical background of the approach, efficient algorithms to handle the computational complexity, and gives implementation details and reports experimental results on a variety of tasks.

1. Introduction

Language models play an important role in statistical machine translation (SMT), i.e. phrase-based systems like Moses (Koehn et al., 2007) or hierarchical phrase-based systems like Joshua (Li et al., 2009). The classical equation of SMT shows this explicitly. Let us assume that we translate a French sentence f into English e:

$$e^* = \arg\max_e P(e|f) = \arg\max_e P(f|e)P(e) \tag{1}$$

the term $P(e|f)$ was explicitly rewritten with the Bayes rule in order to decompose the overall task into two components:

1. $P(f|e)$ gives the probability that f is a translation of e without necessarily paying much attention to the fact whether the generated sentence is well-formed or not;
2. $P(e)$ expresses the probability that the produced sentence is grammatically and semantically correct without looking at the source sentence.

Corresponding author: holger.schwenk@lium.univ-lemans.fr
Cite as: Holger Schwenk. Continuous-Space Language Models for Statistical Machine Translation. The Prague Bulletin of Mathematical Linguistics No. 93, 2010, pp. 137–146. ISBN 978-80-904175-4-0.
doi: 10.2478/v10108-010-0014-6.

The language model (LM) is responsible for the second task. It is clear that the importance of the LM increases when translating into morphologically rich languages. Despite this importance there seems to be only a limited amount of research in improved language modeling for SMT, and most of the state-of-the-art systems use the well-known n-gram back-off LMs, introduced more than 20 years ago. There is a large body of research on smoothing techniques, i.e. how to obtain probabilities for n-grams that were not observed in the training data, see for instance (Chen and Goodman, 1999) for an extensive comparison. In practice, modified Kneser-Ney smoothing is used in most of the cases. Today, there is a clear tendency to train those models on ever larger amounts of data, up to hundreds of billions of words. Several authors propose variants of the back-off n-gram approach to tackle the enormous computational and storage complexity during training and decoding, for instance (Federico and Cettolo, 2007; Brants et al., 2007; Talbot and Osborne, 2007).

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. An alternative approach is based on a *continuous representation* of the words (Bengio et al., 2003; Schwenk, 2007). 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. This is still an n-gram approach, but an LM probability is available for any possible n-gram without the need to back-off to shorter contexts.

This continuous space LM was very successfully applied in large vocabulary continuous speech recognition (Schwenk, 2007) and references therein, and more recently in statistical machine translation (Schwenk et al., 2006a, 2007; Schwenk and Estève, 2008). In this paper we present an open-source implementation of this approach.

The paper is organized as follows. In the next section we first summarize the theoretical background of the CSLM. Section 3 presents implementation details of the toolkit and section 4 gives an overview on the performance of this approach.

2. Architecture of the continuous space language model

The architecture of the continuous space 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 $h_j{=}w_{j-n+1}, \ldots, w_{j-2}, w_{j-1}$ and the outputs are the posterior probabilities of *all* words of the vocabulary:

$$P(w_j = i|h_j) \qquad \forall i \in [1, N] \tag{2}$$

where N is the size of the vocabulary. The input uses the so-called 1-of-n coding, i.e. 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. The ith line of the $N \times P$ dimensional projection

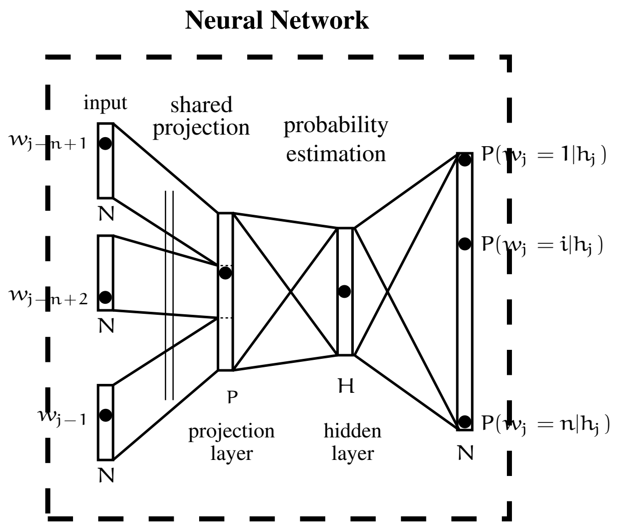

Figure 1. Architecture of the continuous space LM. h_j *denotes the context* $w_{j-n+1}, \ldots, w_{j-1}$. P *is the size of one projection and* H,N *is the size of the hidden and output layer respectively. When short-lists are used the size of the output layer is much smaller then the size of the vocabulary.*

matrix corresponds to the continuous representation of the ith word. Let us denote c_l 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:

$$d_j = \tanh\left(\sum_l m_{jl}\, c_l + b_j\right) \qquad j \in [1, H] \tag{3}$$

$$o_i = \sum_j v_{ij}\, d_j + k_i \qquad i \in [1, N] \tag{4}$$

$$p_i = e^{o_i} \Big/ \sum_{r=1}^{N} e^{o_r} \tag{5}$$

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:

$$E = \sum_{i=1}^{N} t_i \log p_i + \beta \left(\sum_{jl} m_{jl}^2 + \sum_{ij} v_{ij}^2 \right) \tag{6}$$

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 over-fitting the training data (weight decay). This is a well known technique when training neural networks by the back-propagation algorithm The parameter β has to be determined experimentally – in our experiments a value of 3^{-5} was used. Training is done using a resampling algorithm to weight multiple corpora.

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

The complexity to calculate one probability with this basic version of the continuous space LM is quite high due to the large output layer. To speed up the processing several improvements are implemented:

1. *Short-lists*: the neural network is only used to predict the LM probabilities of a subset of the whole vocabulary.
2. *Bunch mode*: several examples are propagated at once through the network. Instead of performing matrix/vector operations we have now matrix/matrix operations which can be heavily optimized on current CPU architectures.
3. *Grouping*: LM probabilities are not requested in a arbitrary order, but requests with the same context h_j are grouped together. By these means, only one forward pass through the neural network is needed.

The idea behind short-lists is to use the neural network to only predict the s most frequent words, s being much smaller than the size of the vocabulary. All words in the vocabulary are still considered at the input of the neural network. The LM probabilities of words in the short-list ($\hat{P}_N$) 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:

$$\hat{P}(w_t|h_t) = \begin{cases} \hat{P}_N(w_t|h_t) P_S(h_t) & \text{if } w_t \in \text{short-list} \\ \hat{P}_B(w_t|h_t) & \text{else} \end{cases} \tag{7}$$

$$P_S(h_t) = \sum_{w \in \text{short-list}(h_t)} \hat{P}_B(w|h_t) \tag{8}$$

It can be considered that the neural network redistributes the probability mass of all the words in the short-list.[1] This probability mass is precalculated and stored with the models. A back-off technique is used if the probability mass for an input context is not directly available. In practice the short-list have a size of 8192 or 12288. This allows to cover about 90% of the LM requests.

It would be relatively straightforward to integrate the CSLM into the Moses decoder. This would however lead to very slow decoding time since the complexity to calculate an arbitrary n-gram probability is much higher for the CSLM than for a standard back-off LM (which in principle just does a table look-up). Several speed-up techniques are possible, but they are not yet implemented. It would be in particular necessary to group requests with the same context.

Instead we use the Moses decoder to create n-best lists which are then processed by a separate tool. This tool supports recalculation of the LM score using a CSLM or a higher-order back-off LM. Several CSLMs can be interpolated on the fly (it is not possible to merge multiple CSLMs into an bigger one like this is done for back-off n-gram models). This tool collects all the LM requests of an n-best lists and groups together requests with the same context h_j. By these means the number of forward passes through the neural network are drastically reduced. In addition, bunch mode is used.

3. Implementation details

The tool is written in C++ and all the routines are contained in the library `libcslm`. The current version is closely interfaced with the SRILM toolkit (Stolcke, 2002). We use in particular classes to load the vocabulary or back-off LMs for the short-list. Therefore, the SRILM toolkit must be installed. It is planed to also support IRSTLM and randomized language models in future versions. In addition, BLAS libraries[2] and the numerical optimization tool CONDOR are needed (see below). The CSLM toolkit provides the following executables:

- `cslm_train`: main tool to train CSLMs.
- `nbest_tool`: tool to process n-best lists (rescoring with a CSLM, recalculation of global scores, solution extraction, . . .)
- `text2bin`: convert text files to a binary representation for faster loading

The tool kit can be downloaded from the following web-page: `http://liumtools.univ-lemans.fr`. Detailed documentation on the available options of the executables and examples are provided with the software.

The neural network routines of the library are generic and can be used for other applications of neural networks. Arbitrary multi-layer networks can be created, either

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

[2]Basic Linear Algebra Subprograms, a commonly used library for matrix operations.

by composing them sequentially, i.e. the output of one layer is fed to the input of the following layer; or in parallel, i.e. the outputs of several layers are concatenated and build up a larger layer (this is used to compose the projection layer of the CSLM). Currently, only fully connected layers are implemented, i.e. each neuron in the input layer is connected with each neuron in the output layer. Training is performed with the standard back-propagation algorithm. Weight decay regularization is available (see equation 6). The neural network functions are implemented with the goal to achieve very fast training and recognition. For this high performance BLAS libraries are needed which take advantage of specifics of the processor to achieve fast matrix operations, e.g. SSE and multi-threading on Intel processors. We use the libraries MKL which are available from Intel for a very small fee,[3] but there are also freely available libraries, like ATLAS.[4] Speed comparison between different BLAS libraries were not performed.

The program `nbest_tool` is used to process Moses-style n-best list. Its main function is to calculate LM probabilities with the CSLM for all the hypotheses. In addition, it can be used to recalculate the global scores for a given set of parameters λ, sort the scores and extract the best hypotheses. This is used to reoptimize the system (see figure 2). It is planed to extend this tool with other operations on n-best lists, in particular MBR decoding.

The CSLM toolkit provides the program `text2bin` to convert the training texts into a binary representation. Basically, each word is replaced with its integer index in the word-list. It is more efficient to process this representation when using the resampling algorithm.

4. Experimental evaluation

The CSLM was initially applied to large vocabulary continuous speech recognition systems. It achieved significant reductions in the word error rate of up to 1% absolute for a large variety of languages and domains (Schwenk, 2007). Based on this success, the application to SMT was investigated, first in the framework of a system to translate texts from the European parliament proceedings (Schwenk et al., 2006b). The good results were then confirmed on a variety tasks, ranging from the resource-poor IWSLT evaluations (Schwenk et al., 2006a, 2007) to large NIST systems (Schwenk and Estève, 2008). These experiments are summarized in the following.

In all cases a two-pass approach was applied: first Moses was run using a 4-gram back-off LM and distinct 1000-best lists were created. Those 1000-best lists were then rescored with the program `nbest_tool`. The LM probabilities calculated by the CSLM can be used to replace those of the standard back-off LM or added as an additional feature function. In both cases we performed a minimum error training of the coeffi-

[3]Math Kernel Library, `http://software.intel.com/en-us/intel-mkl/`

[4]`http://math-atlas.sourceforge.net/`

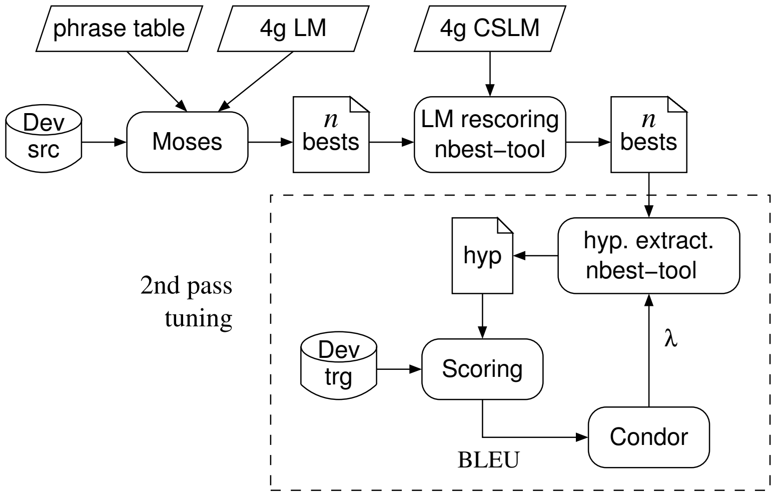

Figure 2. Two pass decoding architecture. First distinct n-best lists are created with Moses using a 4-gram back-off LM. These n-best lists are rescored with the CSLM. The resulting n-best lists are then used to retune the coefficients of the feature functions (2nd pass tuning). For this the freely numerical optimization tool Condor *is used. It iteratively provides a set of parameters λ which are used to recalculate the global score and to extract the current best hypotheses. This is done with the program* nbest-tool. *Those hypotheses are evaluated against the reference translations. The BLEU score is used by the* Condor *tool to propose a new set of parameters λ, until convergence.*

cients of the feature functions, using the freely available numerical optimization tool CONDOR (Berghen and Bersini, 2005). CONDOR performs a Powell-style numerical optimization. The two-pass architecture is summarized in figure 2. Examples of tuning with CONDOR and supporting scripts are provided with the CSLM toolkit.

The goal of the IWSLT BTEC[5] task is to translate typical expressions from the travel domain, mainly between Asian languages and English. There are about 200 to 400 thousand words of bitexts available to train the translation models, in function of the source language. The LM is trained on the English side of those texts. Given the particular domain of this task, adding texts from other sources, e.g. newspaper texts from LDC's Gigaword corpora, has only a small impact. Therefore, it should be promising to apply the CSLM to this task since it is expected to take better advantage of the limited amount of in-domain LM training data. This could be confirmed by the ex-

[5]Basic Travel Expression Corpus

perimental results using standard phrase-based as well as n-gram based SMT systems (Schwenk et al., 2006a). Those results are summarized in Table 1.

	Phrase-based		N-gram-based	
	Ref.	CSLM	Ref.	CSLM
Zh/En	19.74	21.01	20.34	21.16
Ja/En	15.11	15.73	16.14	16.35
Ar/En	23.72	24.86	23.83	23.70
It/En	35.55	37.41	35.95	37.65

Table 1. BLEU scores on the IWSLT 2006 test data, 4-gram back-off versus CSLM for phrase-based and n-gram-based translation systems.

It is also interesting to use the CSLM in conjunction with an n-gram based SMT system to improve the *translation model*. This approach basically uses a language model on bilingual tuples (Mariño et al., 2006), resulting in the same data sparseness problems as for the model on the target language. Practically, we just need to apply the CSLM to change the corresponding feature function in the n-best list. Table 2 presents those results for the Italian/English language pair (Schwenk et al., 2007). The continuous space language model achieved a gain of 1 point BLEU while the continuous space translation model only brought a little more than 0.2 BLEU points. However, both models together achieved a gain of 1.5 points BLEU.

	Back-off TM+LM	neural TM	neural LM	**neural TM+LM**
It/En	36.97	37.21	38.04	**38.50**

Table 2. BLEU scores on IWSLT 2006 test data for the combination of a neural translation model (TM) and a neural language model (LM).

Finally, the method was applied to a task for which very large amounts of LM training data are available, namely the NIST evaluation tasks. The goal of this evaluations is to translate newspaper and web texts from Chinese and Arabic into English. In addition to the English side of about 200 million words of bitexts, the LDC collection of newspaper texts can be considered to be close to in-domain. Those texts are known as the Gigaword corpus and they amount to more than 3 billion words. It is of course impossible to train a neural network on so many examples and this would be even a bad idea: large collections like the AFP texts would dominate the smaller but important text sources. It is well known that it is better to build separate back-off LMs on each text source, to determine a coefficient for each one and to interpolate them together. Loosely inspired by this technique, a resampling method was used to train the neural network. At each epoch of back-propagation training, we resample randomly a subset of each corpus. The resampling coefficient is chosen to take all of the small and important corpora, and smaller subsets of the other *"background corpora"*. More details of this procedure are given in (Schwenk, 2007) and references therein.

The Arabic English system was optimized on the NIST'06 test set and tested on the NIST'08 data. The reference systems achieves a BLEU score of 47.02 using a very large back-off LM that was trained by keeping all the 4-grams that appear in more than 3.3 million words of texts. Applying the CSLM, again by rescoring distinct 1000-best lists, improves the BLEU score to 47.90. The final system achieved a very good ranking in the 2009 NIST evaluation.[6]

5. Conclusion

This paper described an open-source implementation of a continuous space language model for SMT. The integration of the CSLM into the translation process is performed by a two-pass approach: first n-best lists are generated which are then rescored using a provided tool. This approach can be used with different types of machine translation systems, as long as they are able to produce n-best lists that contain the scores of all the feature functions. During the last years, continuous space LMs were successfully applied to a variety of SMT systems. Improvements of the BLEU scores of more than 1 point BLEU were observed and, in general, this also lead to better human judgments.

The CSLM toolkit is meant to be a starting point for ongoing research in the field of estimating probabilities in the continuous domain. Future versions may include factored representations of the words, support for lattices instead of n-best lists and an application of the approach to the translation model.

6. Acknowledgments

This work has been partially funded by the French Government under the project INSTAR (ANR JCJC06 143038) and the European project FP7 EUROMATRIXPLUS.

[6]http://www.itl.nist.gov/iad/mig/tests/mt/2009/ResultsRelease/currentArabic.html

Bibliography

Bengio, Yoshua, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(2):1137–1155, 2003.

Berghen, Frank Vanden and Hugues Bersini. CONDOR, a new parallel, constrained extension of powell's UOBYQA algorithm: Experimental results and comparison with the DFO algorithm. *Journal of Computational and Applied Mathematics*, 181:157–175, 2005.

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

Chen, Stanley F. and Joshua T. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

Federico, Marcello and Maura Cettolo. Efficient handling of n-gram language models for statistical machine translation. In *Second ACL Workshop on Statistical Machine Translation*, pages 88–95, 2007.

Koehn et al., Philipp. Moses: Open source toolkit for statistical machine translation. In *ACL, demonstration session*, 2007.

Li, Zhifei, Chris Callison-Burch, Sanjeev Khudanpur, and Wren Thornton. Joshua: Open source, parsing-based machine translation. *The Prague Bulletin of Mathematical Linguistics*, (91), 2009.

Mariño, J.B., R.E. Banchs, J.M. Crego, A. de Gispert, P. Lambert, J.A.R. Fonollosa, and M. R. Costa-jussà. Bilingual n-gram statistical machine translation. *Computational Linguistics*, 32(4):527–549, December 2006.

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

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

Schwenk, Holger, Marta R. Costa-jussà, and José A. R. Fonollosa. Continuous space language models for the IWSLT 2006 task. In *International Workshop on Spoken Language Translation*, pages 166–173, November 2006a.

Schwenk, Holger, Daniel Déchelotte, and Jean-Luc Gauvain. Continuous space language models for statistical machine translation. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 723–730, 2006b.

Schwenk, Holger, Marta R. Costa-jussà, and José A. R. Fonollosa. Smooth bilingual n-gram translation. In *Empirical Methods in Natural Language Processing*, pages 430–438, 2007.

Stolcke, Andreas. SRILM - an extensible language modeling toolkit. In *International Conference on Speech and Language Processing*, pages II: 901–904, 2002.

Talbot, David and Miles Osborne. Randomised language modelling for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 512–519, 2007. URL `http://www.aclweb.org/anthology/P07-1065`.