Continuous Space Language Models using Restricted Boltzmann Machines

Jan Niehues and Alex Waibel

International Center for Advanced Communication Technologies - InterACT
Institute for Anthropomatics
Karlsruhe Institute of Technology, Germany
firstname.lastname@kit.edu

Abstract

We present a novel approach for continuous space language models in statistical machine translation by using Restricted Boltzmann Machines (RBMs). The probability of an n-gram is calculated by the free energy of the RBM instead of a feed-forward neural net. Therefore, the calculation is much faster and can be integrated into the translation process instead of using the language model only in a re-ranking step.

Furthermore, it is straightforward to introduce additional word factors into the language model. We observed a faster convergence in training if we include automatically generated word classes as an additional word factor.

We evaluated the RBM-based language model on the German to English and English to French translation task of TED lectures. Instead of replacing the conventional n-gram-based language model, we trained the RBM-based language model on the more important but smaller in-domain data and combined them in a log-linear way. With this approach we could show improvements of about half a BLEU point on the translation task.

1. Introduction

Language models are very important in many tasks of natural language processing like, for example, machine translation or speech recognition. In most of these tasks, n-gram-based language models are successfully used. In this model the probability of a sentence is described as a product of the probabilities of the words given the previous words. For the conditional word probability a maximum likelihood estimation is used in combination with different smoothing techniques. Although this is often a very rough estimation, especially for rarely seen words, it can be trained very fast. This enables us to make use of huge corpora which are available for many language pairs.

But there are also several tasks where we need to build the best possible language model from a small corpus. When using a machine translation system, in many real-world scenarios we do not want to have a general purpose translation system, but a specific translation system performing well on one task, e.g. like translation of talks. For these cases, it has been shown that the translation quality can be improved significantly by adapting the system to the task. This has successfully been done by using an additional in-domain language model in the log-linear model used in statistical machine translation (SMT).

When adapting an MT system, we need to train a good language model on small amounts of in-domain data. Then the conventional n-gram-based language models often need to back-off to smaller contexts and the models do no longer perform as well. In contrast, continuous space language models (CSLMs) use always the same context size. Furthermore, the longer training time of CSLMs is no problem for small training corpora.

In contrast to most other continuous space language models, which use feed-forward neuronal nets, the probability in a Restricted Boltzmann Machine (RBM) can be calculated very efficiently. This enables us to use the language models during the decoding of the source sentence and not only in a re-scoring step.

The remaining paper is structured as follows: First we will review related work. Afterwards a brief overview of Restricted Boltzmann Machines will be given before we describe the RBM-based language model. In Section 5 we describe the results on different translation tasks. Afterwards, we will give a conclusion.

2. Related Work

A first approach to predict word categories using neural networks was presented in [1]. Later, [2] used neuronal networks for statistical language modelling. They described in detail an approach based on multi-layer perceptrons and could show that this reduces the perplexity on a test set compared to n-gram-based and class-based language models. In addition, they gave a short outlook to energy minimization networks.

An approach using multi-layer perceptrons has successfully been applied to speech recognition by [3], [4] and [5]. One main problem of continuous space language models is the size of the output vocabulary in large vocabulary continuous speech recognition. A first way to overcome this is to use a short list. Recently, [6] presented a structured output layer neural network which is able to handle large output vocabularies by using automatic word classes to group the output vocabulary.

A different approach also using Restricted Boltzmann Machines was presented in [7]. In contrast to our work, no approximation was performed and therefore, the calculation was more computation intensive. This approach and the beforementioned ones based on feed-forward networks were compared by Le et al. in [8].

Motivated by the improvements in speech recognition accuracy as well as in translation quality, authors tried to use the neural networks also for the translation model in a statistical machine translation system. In [9] as well as in [10] the authors modified the n-gram-based translation approach to use the neural networks to model the translation probabilities.

Restricted Boltzmann machines have already been successfully used for different tasks like user rating of movies [11] and images [12].

3. Restricted Boltzmann Machines

In this section we will give a brief overview on Restricted Boltzmann Machines (RBM). We will concentrate only on the points that are important for our RBM-based language model, which will be described in detail in the next section.

RBMs are a generative model that have already been used successfully in many machine learning applications. We use the following definition of RBMs as given in [13].

3.1. Layout

The RBM is a neural network consisting of two layers. One layer is the visible input layer, whose values are set to the current event. In the case of the RBM-based language model the n-gram will be represented by the states of the input layer. The second layer consists of the hidden units. In most cases those units are binary units, which can have two states. For the RBM-based language model we use "softmax" units instead of binary units for the input layer. The softmax units can have K different states instead of only two. They can be modeled as K different binary states with the restriction that exactly one binary unit is in state 1 while all others are in state 0.

In an RBM there are weighted connections between the two layers, but no connections within the layer. The layers are fully connected to each other.

3.2. Probability

The network defines a probability for a given set of states of the input and hidden units by using the energy function. Let v be the vector of all the states of the input units and h be the vector of states of the hidden units. Then the probability is defined as:

$$p(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{1}$$

using the energy function

$$E(v,h) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{2}$$

and the partition function

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{3}$$

In these formulas a_i is the bias of the visible units, while b_j is the bias of the hidden units. w_{ij} is the weight of the connection between the visible unit v_i and the hidden unit h_j.

If we want to assign the probability to a word sequence, we only have the input vector, but not have the hidden value. Therefore, we would like to have the probability of this word sequence with any given hidden value. Therefore, the probability of a visible vector is defined as:

$$p(v) = \frac{1}{Z} \sum_{h} e^{-E(v,h)} \tag{4}$$

The problem of this definition is that it is exponential in the number of hidden units. A better way to calculate this probability is to use the free energy of the visible vector $F(v)$:

$$e^{-F(v)} = \sum_{h} e^{-E(v,h)} \tag{5}$$

The free energy can by calculated as:

$$F(v) = -\sum_{i} v_i a_i - \sum_{j} log(1 + e^{x_j}) \tag{6}$$

In this definition x_j is defined as $b_j + \sum_i v_i w_{ij}$. Using this definition, we are still not able to calculate the probability $p(v)$ efficiently because of Z. However, we can calculate $e^{F(v)}$ efficiently, which is proportional to the probability $p(v)$, since Z is constant for all input vectors.

3.3. Training

In most cases RBMs are trained using Contrastive Divergence [14]. The aim during training is to increase the probability of the seen training example. In order to do this, we need to calculate the derivation of probability of the example given the weights:

$$\frac{\delta log p(v)}{\delta w_{ij}} = < v_i h_j >_{data} - < v_i h_j >_{model} \tag{7}$$

where $<>$ indicates the expectation of the value between the brackets given the distribution indicated after the brackets. The first term can be calculated easily, since there are no interconnections between the hidden units.

For the second term we use the expected value under a reconstructed distribution instead of the model distribution. This leads to a very rough approximation of the gradient, but in several experiments it was shown that it performs very well.

4. RBMs for Language modeling

After giving a general overview of RBMs we will now describe the RBM that is used for language modeling in detail. Furthermore, we will describe how we derive the sentence probability from the probabilities calculated by the RBM and how we integrate the RBM into the translation process.

4.1. Layout

The layout of the RBM used for language modeling is shown in Figure 1. The input layer of the n-gram language model consists of N blocks of input units for every word of the n-gram. Each of these blocks consists of a softmax unit, which can assume V different states representing the words of the vocabulary, where V is the vocabulary size. These softmax units are modeled by V binary units, where always exactly one unit has the value 1 and the other units have the value 0. The vocabulary consists of all the words of the text as well as the sentence end and beginning mark ($< s >$,$< /s >$) and the unknown word $< unk >$.

The hidden layer consists of H hidden units, where H is a free parameter, which will be set.

Using this setup, we need to train $N * V * H$ weights connecting the hidden and visible units as well as $N * V + H$ bias values.

4.1.1. Word Factors

For some tasks, it is interesting to not only use the surface form of the word, but consider different word factors. We can, for example, use also the part-of-speech (POS) tags of the words or we can use automatically generated word clusters. Such abstract word classes have the advantage, that they are seen more often and therefore, their weights can be trained more reliably. In this case, the additional word factor can be seen as a kind of smoothing.

The layout described before can be easily extended to also use different word factors. In that case, each of the N blocks consists of W sub-blocks, where W is the number of word factors that are used. These sub-blocks are then softmax units with different sizes depending on the vocabulary size of the factor. Like it is in the original layout, all the softmax units are then fully connected to all hidden units. The remaining layout of the framework stays the same.

4.2. Training

As it is done in most RBMs we train our model using contrastive divergence. In a first step, we collect all n-grams of the training corpus and shuffl them randomly. We then split the training examples into chunks of m examples to calculate the weight updates. This is done by calculating the difference between the products mentioned in Equation 7. The first term of the equation is straightforward to calculated. The second term is approximated using Gibbs sampling as suggested in [13]. Therefore, first the values of the hidden values are calculated given the input. Then the values of the visible units given the hidden values is calculated. And finally, a second forward calculation is used. In our experiments we only used one iterations of Gibbs sampling. In our experiments we use a value of 10 for m.

After calculating the updates, we average over all examples and then update the weights using a learning rate of 0.1. As described in [13], by averaging over the examples the size of the update is independent of m and therefore the learning rate does not need to be changed depending on the batch size. Unless stated otherwise, we perform this training for one iteration on the whole corpus.

4.3. Sentence Probabilty

Using the network described before we are able to calculate $e^{F(v)}$ efficiently, which is proportional to the probability of the n-gram $P(w_1 \dots w_N)$.

If we want to use the language model as part of a translation system, we are not interested in the probability of an n-gram, but the probability of a sentence $S =< s > w_1 \dots w_L < /s >$. In an n-gram-based language model this is done by defining the probability as a product of the word probabilities given its history $P(S) = \prod_{i=1}^{L+1} P(w_i|h_i)$, where we use $w_i =< s >$ for $i \leq 0$ and $w_i =< /s >$ for $i > L$. In an n-gram-based approach $P(w_i|h_i)$ is approximated by $P(w_i|w_{i-N+1} \dots w_{i-1})$.

In our approach we are able to calculate a score proportional to $P(w_1 \dots w_N)$ efficiently, but for the conditional probability we would need to sum over the whole vocabulary as shown in Equation 8, which would no longer be efficient.

$$P(w_i|w_{i-N+1} \dots w_{i-1}) = \frac{P(w_{i-N+1} \dots w_i)}{\sum_{w' \in V} P(w_{i-N+1} \dots w_{i-1} w')} \quad (8)$$

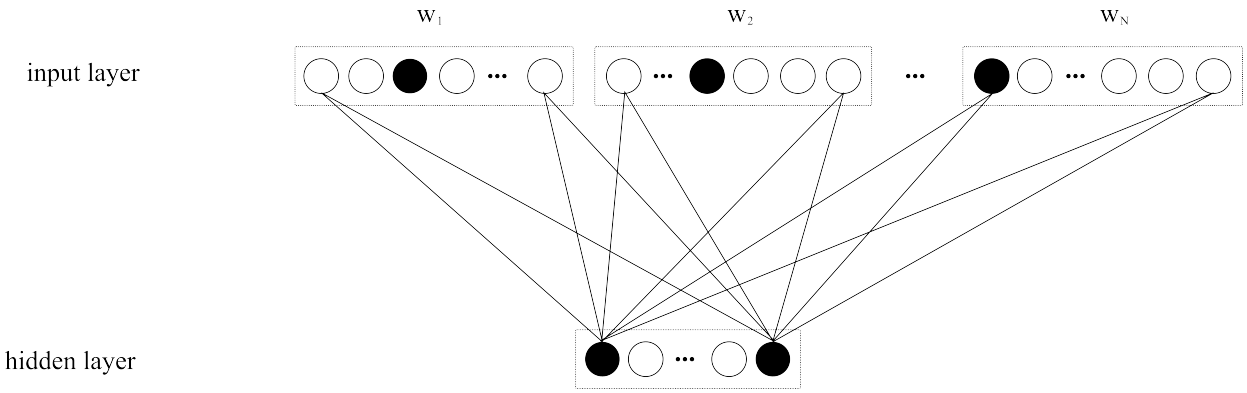
One technique often used for n-gram-based language models is to interpolate the probabilities of different history lengths. If we use the geometric mean of all n-gram probabilities up to the length N in our model we get the following definition for the conditional probability:

$$P'_{GM}(w_i|h_i) = \sqrt[N]{\prod_{j=1}^{N} P_j(w_i|w_{i-j+1} \dots w_{i-1})} \quad (9)$$

$$P_{GM}(w_i|h_i) = \frac{1}{Z_{h_i}} P'_{GM}(w_i|h_i) \quad (10)$$

where $Z_{h_i} = \sum_{w'} P'_{GM}(w'|h_i)$. Using this definition we can express the sentence probability $P_{RBM}(S)$ of our RBM-

Figure 1: RBM for Language model



based language model as:

$$
\begin{aligned}
P_{RBM}(S) &= \prod_{i=1}^{L+1} P_{GM}(w_i|h_i) \qquad (11)\\
&= \prod_{i=1}^{L+1} \frac{1}{Z_{h_i}} * \sqrt[N]{P'_{RBM}(S)}\\
P'_{RBM}(S) &= \prod_{i=1}^{L+1}\prod_{j=1}^{N} P(w_i|w_{i-j+1}\dots w_{i-1})\\
&\overset{(\dagger)}{=} \prod_{i=1}^{L} P(w_{i-N+1}\dots w_i)\\
&* \prod_{j=2}^{N} \frac{P(w_{L-j+2}\dots w_L < /s >)}{P(< s >)} * P(< /s >)\\
&= \frac{1}{Z_S}\prod_{j=1}^{L+N-1} \frac{1}{Z_M} e^{F(w_{j-N+1}\dots w_j)}
\end{aligned}
$$

In (†) we used the fact that $P(w_i|w_{i-j+1}\dots w_{i-1}) = P(w_{i-j+1}\dots w_{i-1}w_i)/P(w_{i-j+1}\dots w_{i-1})$. Then, except for the beginning and the end, all n-gram probabilties for $n < N$ cancel out. In the last line, Z_M is the partition function of the RBM and $Z_S = P(< /s >)/P(< s >)^{N-1}$.

To use the probability in the log-linear model we get:

$$
\begin{aligned}
log(P_{RBM}(S)) = &- \frac{1}{N}(log(Z_S) + (N-1) * log(Z_M))\\
&- \frac{L}{N} * log(Z_M) \qquad (12)\\
&- \sum_{i=1}^{L+1} log(Z_{h_i})\\
&+ \frac{1}{N}\sum_{j \in L+N-1} F(w_{j-N+1}\dots w_j)
\end{aligned}
$$

Here the first term is constant for all sentences, so we do not need to consider it in the log-linear model. Furthermore, the second term only depends on the length of the sentence. This is already modeled by the word count model in most phrase-based translation system. We cannot calculate the third term efficiently. If we ignore this term, it means that we approximate all n-gram probabilities by the unigram probabilities in this term, because in this case Z_{h_i} is zero. By using this approximation, we can use the last term as a good feature to describe the language model probability in our log-linear model. As described before, this part can be calculated efficiently.

The integration to the decoding process is very similar to the one used in n-gram-based language models. If we extend one translation hypothesis by a word, we have to add the additional n-gram probability to the current feature value as it is also done in the standard approach. We also have to save the context of $N-1$ words to calculate the probability. The only difference is that we add at the end of the sentence not only one n-gram ending with $< /s >$, but all the ones containing $< /s >$.

5. Evalutation

We evaluated the RBM-based language model on different tasks. We will first give a brief description of our SMT system. Then we will describe in detail our experiments on the German to English translation task. Afterwards, we will describe some more experiments on the English to French translation task.

5.1. System description

The translation system was trained on the European Parliament corpus, News Commentary corpus, the BTEC corpus and TED talks[1] . The data was preprocessed and compound splitting was applied for German. Afterwards the discriminative word alignment approach as described in [15] was applied to generate the alignments between source and target words. The phrase table was built using the scripts from the

[1] http://www.ted.com

Moses package [16]. A 4-gram language model was trained on the target side of the parallel data using the SRILM toolkit [17]. In addition we used a bilingual language model as described in [18].

Reordering was performed as a preprocessing step using POS information generated by the TreeTagger [19]. We used the reordering approach described in [20] and the extensions presented in [21] to cover long-range reorderings, which are typical when translating between German and English.

An in-house phrase-based decoder was used to generate the translation hypotheses and the optimization was performed using MERT[22].

We optimized the weights of the log-linear model on a separate set of TED talks and also used TED talks for testing. The development set consist of 1.7K segments containing 16K words. As test set we used 3.5K segments containing 31K words.

5.2. German to English

The results for translating German TED lectures into English are shown in Table 1. The baseline system uses a 4-gram language model trained on the target side of all parallel data. If we add a 4-gram RBM-based language model trained only on the TED data for 1 iteration using 32 hidden units we can improve the translation quality on the test data by 0.8 BLEU points (RBMLM H32 1Iter). We can gain additional 0.6 BLEU points by carrying out 10 instead of only 1 iteration of contrastive divergence.

If we use a factored language model trained on the surface word forms and the automatic clusters generated by the MKCLS algorithm [23] (FRBMLM H32 1Iter), we can get an improvement of 1.1 BLEU points already after the first iteration. We grouped the words into 50 word classes by the MKCLS algorithm.

If we add an n-gram-based language model trained only on the in-domain data (Baseline+NGRAM), we can improve by 1 BLEU point over the baseline system. So the factored RBM-based language model as well as the one trained for 10 iteration can outperform the second n-gram-based language model.

We can get further improvements by combining the n-gram-based in-domain language model and the RBM-based language model. In this case we use 3 different language models in our system. As shown in the lower part of Table 1, additional improvements of 0.3 to 0.4 BLEUs points can be achieved compared to the system not using any RBM-based language model. Furthermore, it is no longer as important to perform 10 iteration of training. The difference between one and 10 training iterations is quite small. The factored version of the language model still performs slightly better than the language model trained only on words.

Table 1: *Experiments on German to English*

Iterations	BLEU Score	
	Dev	Test
Baseline	26.31	23.02
+ RBMLM H32 1Iter	27.39	23.82
+ RBMLM H32 10Iter	27.61	24.47
+ FRBMLM H32 1Iter	27.54	24.15
Baseline+NGRAM	27.45	24.06
+ RBMLM H32 1Iter	27.64	24.33
+ RBMLM H32 10Iter	27.95	24.38
+ FRBMLM H32 1Iter	27.80	24.40

5.3. Network layout

We carried out more experiments on this task to analyse the influence of the network layout on the translation quality. Therefore, we used a smaller system only using the n-gram-based or RBM-based in-domain language model trained on the target side of the TED corpus. The results of these experiments are summarised in Table 2. The first system uses an n-gram-based language model trained on the TED corpus. The other systems use all an RBM-based language model trained for one iteration on the same corpus.

When comparing the BLEU scores on the development and test data, we see that we can improve the translation quality by increasing the number of hidden units to up to 32 hidden states. If we use less hidden states, the network is not able to store the probabilities of the n-grams properly. If we increase the number of hidden units further, the performance in translation quality decreases again. One reason for this might be that we have too many parameters to train given the size of the training data.

Table 2: *Experiments using different number of hidden units*

System	Hidden Units	BLEU Score	
		Dev	Test
NGRAM		27.09	23.80
RBMLM	8	25.65	23.16
	16	25.67	23.07
	32	26.40	23.41
	64	26.12	23.18

5.4. Training iterations

One critical point of the continuous space language model is the training time. While an n-gram-based language model can be trained very fast on a small corpus like the TED corpus without any parallelization, the training of the continuous space language model takes a lot longer. In our case the corpus consists of 942K words and the vocabulary size is 28K. We trained the RBM-based language model using 10 cores in parallel and it took 8 hours to train the language model for

one iteration.

Therefore, we analysed in detail the influence of the number of iterations on the translation performance. The experiments were again performed on the smaller system using no large n-gram-based language model mentioned before (No Large LM) and the system using a large n-gram language model trained on all data mentioned in the beginning (Large LM). They are summarized in Table 3. In the first line we show the performance of the system using a n-gram-based language model trained only on the TED corpus for comparison.

In these experiments, we see that the performance increases up to 10 iterations of the training data. Using 10 instead of one iteration, we can increase the translation quality by up to 0.5 BLEU points on the development data as well as on the test data. Using the large language model we could outperform the small n-gram-based language model by the RBM-based language model trained for 10 iterations. Performing more than 10 iterations does not lead to further improvements. The translation quality even decreases again. The reason for this might be that we are facing over-fitting after the 10th iteration. In the smaller setup, using the RBM language model cannot help to outperform the n-gram-based language model.

Table 3: *Experiments using different number of training iterations*

System	Iterations	No Large LM		Large LM	
		Dev	Test	Dev	Test
NGRAM		27.09	23.80	27.45	24.06
RBMLM	1	26.40	23.41	27.39	23.82
	5	26.72	23.38	27.40	23.98
	10	26.90	23.51	27.61	24.47
	15	26.57	23.47	27.63	24.22
	20	26.16	23.20	27.49	24.30

5.5. RBMLM for English-French

We also tested the RBM-based language model on the English to French translation task of TED lectures. We trained and tested the system on the data provided for the official IWSLT Evaluation Campaign 2012. The system is similar to the one used on the German to English tasks, but uses language model and phrase table adaptation to the target domain. The results for this task are shown in Table 4.

The difference between the Baseline system and the systems using RBM-based language models is smaller than in the last experiments, since the baseline system uses already several n-gram-based language models. On the development set both the RBM-based language model as well as the factored RBM-based language model using also automatic word classes could improve by 0.1 BLEU points. For the test set only the factored version can improve the translation quality by 0.1 BLEU points.

Table 4: *Experiments on English to French*

Iterations	BLEU Score	
	Dev	Test
Baseline	28.93	31.90
RBMLM	28.99	31.76
FRBMLM	29.02	32.03

6. Conclusions

In this work we presented a novel approach for continuous space language models. We used a Restricted Boltzmann Machine instead of a feed-forward neuronal net. Since this network is less complex, we were able to integrate it directly into the decoding process. Using this approach, the run-time for the calculation of the probability no longer depends on the vocabulary size, but only on the number of hidden units.

The layout of the network allows an easy integration of different word factors. We were able to improve the quality of the language model by using automatically determined word classes as an additional word factor.

As shown in the experiments, this type of language model works especially well for quite small corpora as they are typically used in the domain adaptation scenario. Therefore, the longer training time of a continuous space language model does not matter as much as for language models trained on huge amounts of data.

By integrating this language model into our statistical machine translation system, we could improve the translation quality by up to 0.4 BLEU points compared to a baseline system using already an in-domain n-gram-based language model.

7. Acknowledgements

This work was partly achieved as part of the Quaero Programme, funded by OSEO, French State agency for innovation. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287658.

8. References

[1] M. Nakamura, K. Maruyama, T. Kawabata, and K. Shikano, "Neural network approach to word category prediction for english texts," in *Proceedings of the 13th conference on Computational linguistics - Volume 3*, ser. COLING '90, 1990.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, Mar. 2003.

[3] H. Schwenk and J.-L. Gauvain, "Connectionist language modeling for large vocabulary continuous speech

recognition," in *In International Conference on Acoustics, Speech and Signal Processing*, 2002.

[4] H. Schwenk, "Continuous space language models," *Comput. Speech Lang.*, vol. 21, no. 3, Jul. 2007.

[5] T. Mikolov, M. Karafit, L. Burget, J. ernock, and S. Khudanpur, "Recurrent neural network based language model," in *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH2010)*, vol. 2010, no. 9. International Speech Communication Association, 2010.

[6] H. S. Le, I. Oparin, A. Allauzen, J.-L. Gauvain, and F. Yvon, "Structured output layer neural network language model," in *ICASSP*. IEEE, 2011.

[7] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proceedings of the 24th International Conference on Machine Learning*, 2007.

[8] H. S. Le, A. Allauzen, G. Wisniewski, and F. Yvon, "Training continuous space language models: Some practical issues," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, 2010.

[9] H. Schwenk, M. R. Costa-jussa, and J. A. R. Fonollosa, "Smooth bilingual n-gram translation," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, June 2007.

[10] H.-S. Le, A. Allauzen, and F. Yvon, "Continuous Space Translation Models with Neural Networks," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Montréal, Canada: Association for Computational Linguistics, Jun. 2012.

[11] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning*, ser. ICML '07. New York, NY, USA: ACM, 2007.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, Jul. 2006.

[13] G. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines," Tech. Rep., 2010.

[14] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002.

[15] J. Niehues and S. Vogel, "Discriminative Word Alignment via Alignment Matrix Modeling." in *Proc. of Third ACL Workshop on Statistical Machine Translation*, Columbus, USA, 2008.

[16] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open Source Toolkit for Statistical Machine Translation," in *ACL 2007, Demonstration Session*, Prague, Czech Republic, 2007.

[17] A. Stolcke, "SRILM – An Extensible Language Modeling Toolkit." in *Proc. of ICSLP*, Denver, Colorado, USA, 2002.

[18] J. Niehues, T. Herrmann, S. Vogel, and A. Waibel, "Wider Context by Using Bilingual Language Models in Machine Translation," in *Sixth Workshop on Statistical Machine Translation (WMT 2011)*, Edinburgh, UK, 2011.

[19] H. Schmid, "Probabilistic Part-of-Speech Tagging Using Decision Trees," in *International Conference on New Methods in Language Processing*, Manchester, UK, 1994.

[20] K. Rottmann and S. Vogel, "Word Reordering in Statistical Machine Translation with a POS-Based Distortion Model," in *TMI*, Skövde, Sweden, 2007.

[21] J. Niehues and M. Kolss, "A POS-Based Model for Long-Range Reorderings in SMT," in *Fourth Workshop on Statistical Machine Translation (WMT 2009)*, Athens, Greece, 2009.

[22] A. Venugopal, A. Zollman, and A. Waibel, "Training and Evaluation Error Minimization Rules for Statistical Machine Translation," in *Workshop on Data-drive Machine Translation and Beyond (WPT-05)*, Ann Arbor, MI, 2005.

[23] F. J. Och, "An Efficient Method for Determining Bilingual Word Classes." in *EACL'99*, 1999.