Combining Techniques from different NN-based Language Models for Machine Translation

[†]**Jan Niehues** jan.niehues@kit.edu
[*]**Alexander Allauzen** allauzen@limsi.fr
[*] **François Yvon** yvon@limsi.fr
[†]**Alex Waibel** alex.waibel@kit.edu
[†]Karlsruhe Insitute of Technology, Karlsruhe, Germany
[*]Université Paris-Sud XI and LIMSI-CNRS, Orsay, France

Abstract

This paper presents two improvements of language models based on Restricted Boltzmann Machine (RBM) for large machine translation tasks. In contrast to other continuous space approach, RBM based models can easily be integrated into the decoder and are able to directly learn a hidden representation of the n-gram. Previous work on RBM-based language models do not use a shared word representation and therefore, they might suffer of a lack of generalization for larger contexts. Moreover, since the training step is very time consuming, they are only used for quite small copora. In this work we add a shared word representation for the RBM-based language model by factorizing the weight matrix. In addition, we propose an efficient and tailored sampling algorithm that allows us to drastically speed up the training process. Experiments are carried out on two German to English translation tasks and the results show that the training time could be reduced by a factor of 10 without any drop in performance. Furthermore, the RBM-based model can also be trained on large size corpora.

1 Introduction

Language models are very important in many natural language processing tasks like, for example, machine translation and 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 estimate, especially for rarely seen words, it can be trained very fast. This facilitates the use of huge corpora which are available for many language pairs.

Recently, language models based on neural network (NN) have successfully been used as an additional model in several tasks. In contrast to conventional n-gram language models, these models can cover longer contexts without a prohibitive increase of the number of parameters. Thereby, dependencies between words that do not occur in the direct neighborhood can be modeled. In state of the art language models contexts of up to 10 words are used.

Currently, most of these NN-based language models use feed-forward networks. These models can be trained on very large monolingual corpora. Furthermore, in most cases a shared word representation for all word positions is learned. Since the calculation of the language model probabilities is quite complex, often the language model can not be used during decoding, but only in a rescoring step. Vaswani et al. (2013) presented an approach to also use feed-forward language models during decoding.

Niehues and Waibel (2012a) proposed a language model based on Restricted Boltzmann Machine (RBM). Since this model uses a quite simple layout, the probability computation is very fast and the language model can be used during decoding. In contrast, the training time of these models depends on the vocabulary size and therefore, the training time can be quite long. Furthermore, this model does not make use of a shared word representation, which can hinder its generalization power with large context.

Motivated by techniques developed for other NN-based language model, we tackle in this work these two issues of RBM-based language models. First, we propose tailored sampling algorithms to speed up the training process. Secondly, we introduce a factored representation of the weight matrix to learn a shared word representation. Furthermore, we analyzed the advantage of the RBM-based language model to use the language model during decoding.

The remaining paper is structured as follows. First we review related work and then provide in section 3 a brief overview of RBM-based language models. Section 4 describes the tailored sampling strategies while we describe how the shared word representation is integrated into the RBM layout in section 5. Afterwards we describe and discuss experimental results measured in terms of translation quality in section 6.

2 Related Work

A first approach to predict word categories using neural networks was presented in Nakamura et al. (1990). Later, Bengio et al. (2003) introduced neural networks for statistical language modeling. The authors described in detail an approach based on multi-layer neural networks and reported a significative perplexity reduction compared to conventional and class-based language models. In addition, they gave a short outlook to energy minimization networks.

An approach using multi-layer neural networks has successfully been applied to speech recognition by Schwenk and Gauvain (2002), Schwenk (2007) and Mikolov et al. (2010). 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, Le et al. (2011) presented a structured output layer neural network which is able to handle large output vocabularies by using a clustering tree to represent the output vocabulary. 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 Schwenk et al. (2007) as well as in Le et al. (2012) the authors modified the n-gram-based translation approach to use the neural networks to model the translation probabilities. In Vaswani et al. (2013), noisy-contrastive estimation was used to train the neural network. Therefore, the probabilities do not need to be normalized and the language model can be used during decoding.

A different approach also using Restricted Boltzmann Machines was presented in Mnih and Hinton (2007). However this approach exhibits the same complexity issue as feed-forward models. A head to head comparison between RBM and feed-forward language models can be found in Le et al. (2010). In Niehues and Waibel (2012a), another RBM-based language model was introduced. This approach differs from the one intrdoduced in Mnih and Hinton (2007) by a simpler layout that allows us for a fast probability computation. This yields the integration of the model during the decoding step feasible. However, the training complexity heavily depends on the vocabulary size and this model can be trained on a limited amount of training data. In Dahl et al. (2012), a sampling method was presented to efficiently train restricted Boltzmann machines on word observations. This approach enables us to train large scale language models.

3 RBM-based language models

In this section we will briefly review the continuous space language models using RBMs introduced in Niehues and Waibel (2012a). A restricted Boltzmann machine is a generative stochastic artificial neural network that can learn a probability distribution over a set of visible variables that represent the observed features and a set of hidden variables. Variables are represented by neural units and grouped in two layers, one for the visibible units and one for the hidden variables. For that purpose a RBM forms a bipartite graph as shown in Figure 1, where the connections between the visible and hidden layers have to be trained.

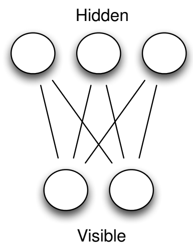


Figure 1: Restricted Boltzmann Machine.

The activation of the visible neurons will be determined by the input data. The standard input layer for neural network language models uses a 1-of-n coding to insert a word from the vocabulary into the network. This is a vector, where only the index of the word in the vocabulary is set to one and the rest to zero. The activation of the hidden units is usually binary and will be inferred from the visible units by using sampling techniques. Figure 2(a) is an example of the RBM-based language model.

To calculate the probability of a visible configuration v we will use the definition of the free energy in a restricted Boltzmann machine with binary stochastic hidden units, which is

$$F(v) = -\sum_i v_i a_i - \sum_j log(1 + \mathrm{e}^{x_j}) \tag{1}$$

where a_i is the bias of the ith visible neuron v_i and x_j is the activation of the jth hidden neuron. The activation x_j is defined as

$$x_j = b_j + \sum_i v_i w_{ij} \tag{2}$$

where b_j is the bias of the jth hidden neuron and w_{ij} is the weight between visible unit v_i and hidden unit x_j. Using these definitions, the probability of our visible configuration v is

$$p(v) = \frac{1}{Z} \mathrm{e}^{-F(v)} \tag{3}$$

with the partition function $Z = \sum_v \mathrm{e}^{-F(v)}$ being the normalization constant. Usually this normalization constant is not easy to compute, since it is a sum over an exponential amount of values. We know that the free energy will be proportional to the true probability of our visible vector. This is the reason for using the free energy as a feature in our log-linear model instead of the true probability. In order to use it as a feature inside the decoder we actually need to be able to compute the probability for a whole sentence. As shown in Niehues and Waibel (2012a) we can do this by summing over the free energy of all n-grams contained in the sentence.

3.1 Training

For training the restricted RBM-based language model (RBMLM) we used the Contrastive Divergence (CD) algorithm as proposed in Hinton (2002). In order to do this, we need to calculate the derivative of the probability of the training example given the weights

$$\frac{\delta \log p(v)}{\delta w_{ij}} = <v_i h_j>_{data} - <v_i h_j>_{model} \tag{4}$$

where $<v_i h_j>_{model}$ is the expected value of $v_i h_j$ given the distribution of the model. In other words we calculate the expectation of how often v_i and h_j are activated together, given the distribution of the data, minus the expectation of them being activated together given the distribution of the model, which will be calculated using Gibbs-Sampling techniques.

We need to calculate the hidden activation given the input data $p(h_i|v)$ for the first term. In this case only n of the $V \times n$ visible units have non-zero values, where n is the context size and V the vocabulary size. Therefore, this can be done very fast. In each step of the Gibbs sampling, first the activation probability of the visible layer given the hidden layer $p(v_i^s|h)$ is calculated. Afterwards, the activation of the hidden variable given the sampled visible layer $p(h_i^s|v^s)$ needs to be calculated. In this case, not only n visible units have to be considered, but all $V \times n$ ones. Consequently, the calculation is very slow especially for large vocabularies.

Usually many steps of Gibbs-Sampling are necessary to get an unbiased sample from the distribution, but in the Contrastive Divergence algorithm only one step of sampling is performed.

4 Sampling

In contrast to the time used for the calculation of the free energy, the time used for training the language model heavily depends on the vocabulary size. Therefore, the training of large scale language models is too time consuming. One solution to reduce the training time is to rely on a different sampling strategy.

4.1 Metropolis-Hasting sampling for word observations

A similar language model has been presented in Mnih and Hinton (2007). However, the training time issue due to the vocabulary size is similar. In Dahl et al. (2012), an approach based on Metropolis Hasting was presented to solve this problem. In contrast to considering all input units v_i during Gibbs sampling, they first sample a list of words for every word position. Then only the visible units corresponding to these words are used during the calculation of $p(v_i^s|h)$ and $p(h_i^s|v^s)$.

While this approach uses probabilistic binary values for the input neurons, Hinton (2010) mention that it is advantageous to use the probabilities themselves. Therefore, we present a different approach for sampling which uses the probabilities themselves. Furthermore, we tried to include other information into the sampling process like the confusability of words in the translation model.

4.2 A tailored sampling approach

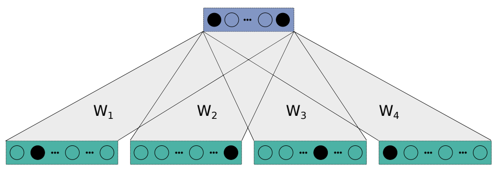
In order to minimize the effort for this calculation, we developed the following approximation. Instead of calculating all probabilities, we have a set of candidate words C_j for every word position. Then only the probabilities $p(v_i^s|h)$ for $v_i \in C_0 \dots C_n$ need to be calculated. Since the probabilities are calculated given the hidden activation, they should lead to similar n-grams. Therefore, most probabilities will be close to zero and some of them can be neglected. With this approximation, we no longer calculate all probabilities, but only the ones for the candidate words $C_0 \dots C_n$. Therefore, the time needed for the calculation no longer depends on $n * v$, but on $\sum_{i=1}^{n} |C_i|$.

In this work, we investigate different possibilities to calculate the set of candidate words C_i. In a first approach, we sample the words using the unigram probabilities (*unigram*). We pre-calculate a set of candidates for every word and use the same set of words during training. In this case, we sample until we have c distinct words.

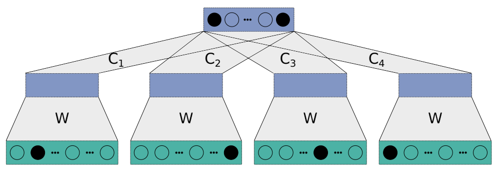
In a second attempt, we re-sample for every training example (*unigram.online*) using the same probability distribution. Since this has to be done during training, we only sample c times. Therefore, the number of words in the set can be lower than c.

Since we are using the language model in a translation system, it is more important that the language model can distinguish between words that are translations of the same source word than other words. Therefore, we use information from the translation model to select our samples (*translation*). We use the lexical translation probabilities $p(e|f)$ and $p(f|e)$ to calculate the translation confusability probability $p_c(e'|e) = \sum_{f \in F} p(e'|f) * p(f|e)$. We then use this probability to sample the set $C(e)$. Since the set of words, where the probability $p_c(e'|e)$ is non-zero, is often smaller than c, we fall-back to unigram probabilities, if we sample e itself. This is also done in a preprocessing step until we sampled c distinct words.

5 Shared Word Representation



(a) Baseline layout of RBM-based language model



(b) Shared Word Representation

Figure 2: Different layouts for RBM-based language models

In the RBM-based language model as well as in other neural network language models, the input words are encoded by an 1-out-of-n vector. In the RBM-based language model, this vector is then multiplied by the weight matrix W_i of the size $N * H$, where H is size of the hidden layer. An illustration is shown in Figure 2(a). Since many words occur quite rarely, some of the weights cannot not be estimated well.

In contrast, most other neural network language model use a shared word representation. In this case, the sparse word vector is first multiplied by a matrix W shared over all word positions, resulting in a word representation vector. These representations are then multiplied by a position dependent weight matrix to form the input for the hidden layer.

In Mnih and Hinton (2007), a factorization of the weight matrix into a word dependent and context dependent part for the bi-log-linear model is suggested. Thereby, it is possible to also use a shared representation for the words.

We adapted this approach to the RBM-based language model. This results in the layout presented in Figure 2(b). Instead of multiplying the input vector by W_i, it is now first multiplied by W and then by the context dependent part C_i. Therefore, the number of weights is no longer $n * V * H$, but $V * S + n * S * H$, where S is the size of the shared word representation.

6 Experimental results

We evaluated the sampling and the shared word representation on two tasks translating from German to English. First, we used the RBM-based language model in a speech translation

Sampling size	BLEU Score
No RBMLM	25.16
No Sampling	25.42
5	25.15
10	25.01
50	25.23
100	25.25
1000	25.46

Table 1: Results for different sample sizes

system. Afterwards, the system was used to translate German news data into English.

6.1 System description

The speech 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 Niehues and Vogel (2008) was applied to generate the alignments between source and target words. The phrase table was built using the scripts from the Moses package (Koehn et al., 2007). A 4-gram language model was trained on the target side of the parallel data using the SRILM toolkit (Stolcke, 2002). In addition we used a bilingual language model as described in Niehues et al. (2011).

Reordering was performed as a preprocessing step using POS information generated by the TreeTagger (Schmid, 1994). We used the reordering approach described in Rottmann and Vogel (2007) and the extensions presented in Niehues and Kolss (2009) 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 Minimum Error Rate Training (MERT) (Venugopal et al., 2005).

We tested the language models on two different sets. First, 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. In addition, we optimized and tested the systems on a set of computer science lectures collected at a university.

The language models were tested on three different conditions. First, we used the baseline system, then we used a system, which has been adapted to the TED task by using an additional n-gram language model trained on the TED data. Finally, we also tested the language model in a translation system using phrase table adaptation as described in Niehues and Waibel (2012b). In summary, the language models were tested on 6 different sets and the reported results are the average BLEU scores on all test sets. The RBM-based language model was trained only on the TED data. If not stated differently, we used 32 hidden units and a context length of 4 words.

The system used for the news translation task, was trained on all the data that was available in the WMT 2014. The system was trained similarly. A detailed description of the system can be found in Herrmann et al. (2014). The system was optimized on newstest2012 and tested on newstest2013.

6.2 Sampling

In a first set of experiments we analyzed the correlation of translation performance and the sample size. Therefore, we used the unigram sampling approach and the results are summarized

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

Model	Sampling	Size	1-Layer	Shared
No RBMLM			25.16	
SOUL			25.56	
RBMLM	No Sampling		25.42	25.36
RBMLM	unigram	100	25.25	25.01
	unigram.online	100	25.37	25.49
	translation	100	25.36	25.26
	unigram	1000	25.46	25.29
	unigram.online	1000	25.33	25.30
	translation	1000	25.37	25.4

Table 2: Results for 4-gram RBM-based language models on TED

in Table 1

We are not able to train useful language models with a sample size of five or ten candidates. In this case, the translation performance can not be improved over the baseline using no RBM-based language model. If we increase the sample size to 50 or 100 words, the language model slightly improves the translation quality, but the performance is still worse than the system using no sampling. When using a sample size of 1000 candidates, the same performance as with the language model using no sampling can be achieved.

After having an initial impression of appropriate samples sizes, we evaluated the performance of the different sampling approaches on the TED translation task. The results are summarized in the left column titled as ”1-layer” of Table 2.

The first three lines describe different baseline systems. In the first system, we do not use any neural network language models. In the second system, we used a SOUL language model. This is a feed-forward neural network using a structured output layer (Le et al., 2011).This improved the performance by 0.4 BLEU points on average in all the systems. Finally, as a third baseline system, we trained an RBM-based language model on the TED data. In this case, we can improve the performance by 0.3 BLEU points. Hence, we get similar improvements by using the RBM-based language model and the SOUL language model.

After comparing the baseline systems, we tested the different sampling methods. First, we tested all three sampling techniques using a sample size of 100. The three techniques perform very similar leading to improvements of 0.1 to 0.2 BLEU points. Hence, the performance drops at most by only 0.2 BLEU points. The unigram approach for sampling performs slightly worse in this configuration compared to the other two approaches. When accepting this drop in performance, the training time of the language model drops by from 12 hours to 0.1 to 0.2 hours. Therefore, it is possible to train the model also on corpora that are larger than the quite small TED corpus. Furthermore, the training time no longer depends on the vocabulary size. While the vocabulary size quite small for the TED corpus with 28K words, for many other tasks, it is significantly larger.

Afterwards, we also tested the techniques using a sample size of 1000. In this case, the training time is still faster, reducing the training time to roughly 0.8 hours. In this case all three systems perform very similar to the one using no sampling. The performance drops by at most 0.1 BLEU points. So it is possible to reach approximately the same performance than with the baseline RBM-based language model, but using only one tenth of the training time. Again, all methods to sample the words perform quite similarly.

In conclusion, it is possible to achieve nearly the same performance using sampling than the baseline training method by using only one tenth of the training time. Furthermore, even if we reduce the training time even more by a factor of 10, we have only a slight drop in

SharedWord	Hidden	BLEU
128	32	25.30
256	32	25.38
512	32	25.40
1024	32	25.43
512	64	25.34
512	128	25.33

Table 3: Results for different shared word sizes

performance.

6.3 Shared Word Representation

After we analyzed the influence of the different sampling techniques on the translation quality, we concentrated on evaluating the shared word representation. The results are shown in the right column of Table 2. In these experiments, a shared word representation consists of 512 neurons.

If we use no sampling, the shared representation performs similar, but slightly worse than the 1-layer typology. For the unigram sampling approach, the shared representation performs always worse than the 1-layer typology. For the other two sampling techniques, we see a similar performance for both network layouts.

In the experiments in Table 3, we analyzed the effect of the size of the shared word representation and of the hidden layer. In all the experiments we used translation sampling approach with a sample size of 1000.

We can improve the translation quality by increasing the size of the shared word representation. But starting from a size of 256 the improvements in translation quality are very small. Since the training time depends of the size of the layer, we used a size of 512 for the remaining experiments. If we compare the system using different number of neurons in the hidden layer, the translation quality cannot be improved by using a larger hidden layer.

In summary, we are able to reach a similar performance by using the shared word representation as by the 1-layer layout. But it is not possible to outperform the performance of the 1-layer typology.

6.4 N-Gram Length

After we performed a detailed analysis for 4-gram RBM-based language models, we continue with analyses the performance of the 8-gram language models. The results for these experiments are summarized in the two left most columns in Table 4.

In a first experiment, we used the SOUL language model using 4-gram and 8-gram context for comparison. We can see that it is possible to improve the performance of the SOUL LM by an additional 0.1 BLEU points leading to an improvement of 0.5 BLEU points over the baseline system.

We compared the 4-gram and 8-gram RBM-based language models on 6 different configurations. First, we compared the 1-layer layout and the layout using shared word representation. Furthermore, we tested both typologies using 3 different sampling techniques. We used the unigram.online sampling technique using sample sizes of 100 and 1000 words and the sampling using translation probabilities with 1000 samples.

If we first look at the language models using a 1-layer layout, no improvements can be achieved by using a 8-gram language model compared to the 4-gram language model. Only small changes of less than 0.1 BLEU points can be seen.

Layout	Sampling	Sample size	Rescoring 4-gram	Rescoring 8-gram	Decoder 4-gram	Decoder 8-gram
	No RBMLM		25.16			
	SOUL		25.56	25.66		
1-Layer	unigram.online	100	25.37	25.30	25.55	25.36
1-Layer	unigram.online	1000	25.33	25.41	25.27	25.46
1-Layer	translation	1000	25.37	25.38	25.64	25.24
Shared	unigram.online	100	25.39	25.56	25.51	25.33
Shared	unigram.online	1000	25.30	25.48	25.41	25.55
Shared	translation	1000	25.4	25.48	25.38	25.23

Table 4: Results for 8-gram LMs

In the lower part of the table the results for the RBMs using shared word representation are shown. In this cases the performance can be improved when going from an 4-gram context to an 8-gram context. This improvement is consistent for all sampling techniques. Similar to the SOUL language model, the improvements are between 0.1 and 0.2 BLEU points.

In conclusion, we can only improve the performance by increasing the context from four to eight words, when using the shared word representation. When comparing the performance of the RBM-based and the SOUL language models, the SOUL language model performs roughly 0.1 BLEU points better than the RBM-based language model.

6.5 Language Model Integration

In contrast to other neural network language models like the SOUL model, the RBM-based language model has a quite simple structure. Therefore, it is possible to calculate the probabilities very fast and the language model can be directly integrated into the decoder in contrast to using it only during rescoring. Therefore, we investigated the effect of using the language model during decoding. The results are summarized in the right columns of Table 4.

For the language models using a context of 4-words, the performance can be improved in 4 out of 6 cases by 0.1 to 0.3 BLEU points. In the cases where the performance is worse, the difference is very small, so the performance stays mainly the same. Hence, it is possible to improve the performance by directly integrating the language model into the decoder.

If we look at the 8-gram language model, the integration no longer leads to these improvements. The performance is better when using the language model during decoding, but in all cases by less than 0.1 BLEU points. Furthermore, in three cases the performance drops, in some cases even by 0.2 BLEU points.

In summary, it was possible to improve the performance when using a 4-gram language model during decoding compared to using it only during restoring. In contrast, when using 8-gram language models, this is no longer the case.

6.6 Large-Scale Language Models

After we did a detailed analysis of the different language models on the quite small data of the TED corpus, we also used the language model on a larger scale. Therefore, we trained the language models on the target side of the parallel data of the German-English translation task of the WMT. The results for these experiments are shown in Table 5.

As shown in the table, the SOUL model could improve the translation quality by around 0.4 BLEU points. In this case, we used KBMira optimization instead of MERT, since for this model it performed significant better. In contrast to the TED translation task, the 4-gram and 8-gram language models as well as the 1-layer or Shared Word Representation layout perform similar.

Method	BLEU
Baseline	27.02
SOUL	27.42
1-Layer 4-gram	27.20
1-Layer 8-gram	27.22
Shared 4-gram	27.19
Shared 8-gram	27.15

Table 5: Results for NN language model on WMT task

All this language model could improve the translation quality by 0.1 to 0.2 BLEU points. So it is possible to improve the translation quality using the RBM-based language model, but it not yet performing as well as the SOUL language model.

7 Conclusion

This paper describes the integration of two techniques motivated by other NN-based language models into RBM-based language models. In a first step, we suggested different sampling techniques in order to speed up the training process. Using these techniques, the training time no longer depends on the vocabulary size. Thereby, the training time could be reduced by a factor of 10 without loss in performance. Furthermore, when reducing the training time from 12 hours to 0.2 hours, the performance drops only slightly. With this technique it is possible to train RBM-based language models even on large data sets like the WMT data.

Furthermore, we used a shared word representation by factorizing the weight matrix in an RBM-based language model. On shorter context sizes of 4 words this leads to the same performance as the default layout of an RBM. But for longer context sizes of 8 words, small improvements could be seen by using the shared word representation.

By combining these two contributions, RBM language models can achieved state of the art performance and can be efficiently training on large datasets. In addition, we showed that RBM-based language models can yield similar performance to the well-known SOUL language models. Moreover, one advantage of RBM-based language models is their easy integration into the decoder. We showed that, in our experiments, the performance could only be improved when using shorter contexts of 4 words.

In future work, we will analyze how we could also leverage longer context during decoding. Furthermore, we will investigate the use of deeper neural networks.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287658. This work was partially financed by the DGA RAPID project RAPMAT (`http://www.limsi.fr/tlp/rapmat.html`).

References

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

Dahl, G., Adams, R., and Larochelle, H. (2012). Training restricted boltzmann machines on word observations. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, volume abs/1202.5695, Edinburgh, UK.

Herrmann, T., Mediani, M., Cho, E., Ha, T.-L., Niehues, J., Slawik, I., Zhang, Y., and Waibel, A. (2014). The karlsruhe institute of technology translation systems for the wmt 2014. In *Proceedings of the Ninth Workshop on Statistical Machine Translation (WMT 2014)*, pages 130–135, Baltimore, Maryland, USA.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Hinton, G. (2010). A practical guide to training restricted boltzmann machines. Technical report.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, Prague, Czech Republic.

Le, H., Allauzen, A., Wisniewski, G., and Yvon, F. (2010). Training continuous space language models: Some practical issues. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, Cambridge, MA.

Le, H., Allauzen, A., and Yvon, F. (2012). 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 (NAACL-HLT 2012)*, Montreal, Canada. Association for Computational Linguistics.

Le, H., Oparin, I., Allauzen, A., Gauvain, J.-L., and Yvon, F. (2011). Structured output layer neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2011)*, Prague, Czech Republic. IEEE.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (Interspeech 2010)*, volume 2010, Makuhari, Japan. International Speech Communication Association.

Mnih, A. and Hinton, G. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, Corvallis, Oregon, USA.

Nakamura, M., Maruyama, K., Kawabata, T., and Shikano, K. (1990). Neural network approach to word category prediction for english texts. In *Proceedings of the 13rd International Conference on Computational Linguistics (COLING '90)*, Helsinki, Finland.

Niehues, J., Herrmann, T., Vogel, S., and Waibel, A. (2011). Wider context by using bilingual language models in machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT 2011)*, Edinburgh, UK.

Niehues, J. and Kolss, M. (2009). A pos-based model for long-range reorderings in smt. In *Proceedings of the Fourth Workshop on Statistical Machine Translation (WMT 2009)*, Athens, Greece.

Niehues, J. and Vogel, S. (2008). Discriminative word alignment via alignment matrix modeling. In *Proceedings of the Third Workshop on Statistical Machine Translation (WMT 2008)*, pages 18–25, Columbus, Ohio, USA.

Niehues, J. and Waibel, A. (2012a). Continuous space language models using restricted boltzmann machines. In *Proceedings of the Ninth International Workshop on Spoken Language Translation (IWSLT 2012)*, Hong Kong.

Niehues, J. and Waibel, A. (2012b). Detailed analysis of different strategies for phrase table adaptation in smt. In *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas (AMTA 2012)*, San Diego, California, USA.

Rottmann, K. and Vogel, S. (2007). Word reordering in statistical machine translation with a pos-based distortion model. In *Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-07)*, Skövde, Sweden.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK.

Schwenk, H. (2007). Continuous space language models. *Computer Speech and Language*, 21(3):492–518.

Schwenk, H. and Gauvain, J.-L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2002)*, Orlando, Florida, USA.

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

Stolcke, A. (2002). Srilm – an extensible language modeling toolkit. In *Proceedings of the International Conference of Spoken Language Processing (ICSLP 2002)*, Denver, Colorado, USA.

Vaswani, A., Zhao, Y., Fossum, V., and Chiang, D. (2013). Decoding with large-scale neural language models improves translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1387–1392, Seattle, USA.

Venugopal, A., Zollman, A., and Waibel, A. (2005). Training and evaluation error minimization rules for statistical machine translation. In *Proceedings of the Workshop on Data-drive Machine Translation and Beyond (WPT-05)*, Ann Arbor, Michigan, USA.