Published in the Joint Conference ACL/Coling, pages 723–730, July 2006.

Continuous Space Language Models for Statistical Machine Translation *

Holger Schwenk and **Daniel Déchelotte** and **Jean-Luc Gauvain**
LIMSI-CNRS, BP 133
91403 Orsay cedex, FRANCE
{schwenk,dechelot,gauvain}@limsi.fr

Abstract

Statistical machine translation systems are based on one or more translation models and a language model of the target language. While many different translation models and phrase extraction algorithms have been proposed, a standard word n-gram back-off language model is used in most systems.

In this work, we propose to use a new statistical language model that is based on a continuous representation of the words in the vocabulary. A neural network is used to perform the projection and the probability estimation. We consider the translation of European Parliament Speeches. This task is part of an international evaluation organized by the TC-STAR project in 2006. The proposed method achieves consistent improvements in the BLEU score on the development and test data.

We also present algorithms to improve the estimation of the language model probabilities when splitting long sentences into shorter chunks.

1 Introduction

The goal of statistical machine translation (SMT) is to produce a target sentence $\mathbf{e}$ from a source sentence $\mathbf{f}$. Among all possible target sentences the one with maximal probability is chosen. The classical Bayes relation is used to introduce a target language model (Brown et al., 1993):

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} \Pr(\mathbf{e}|\mathbf{f}) = \arg\max_{\mathbf{e}} \Pr(\mathbf{f}|\mathbf{e}) \Pr(\mathbf{e})$$

where $\Pr(\mathbf{f}|\mathbf{e})$ is the translation model and $\Pr(\mathbf{e})$ is the target language model. This approach is usually referred to as the *noisy source-channel* approach in statistical machine translation.

Since the introduction of this basic model, many improvements have been made, but it seems that research is mainly focused on better translation and alignment models or phrase extraction algorithms as demonstrated by numerous publications on these topics. On the other hand, we are aware of only a small amount of papers investigating new approaches to language modeling for statistical machine translation. Traditionally, statistical machine translation systems use a simple 3-gram back-off language model (LM) during decoding to generate n-best lists. These n-best lists are then rescored using a log-linear combination of feature functions (Och and Ney, 2002):

$$\hat{\mathbf{e}} \approx \arg\max_{\mathbf{e}} \Pr(\mathbf{e})^{\lambda_1} \Pr(\mathbf{f}|\mathbf{e})^{\lambda_2} \qquad (1)$$

where the coefficients λ_i are optimized on a development set, usually maximizing the BLEU score. In addition to the standard feature functions, many others have been proposed, in particular several ones that aim at improving the modeling of the target language. In most SMT systems the use of a 4-gram back-off language model usually achieves improvements in the BLEU score in comparison to the 3-gram LM used during decoding. It seems however difficult to improve upon the 4-gram LM. Many different feature functions were explored in (Och et al., 2004). In that work, the incorporation of part-of-speech (POS) information gave only a small improvement compared to a 3-gram back-off LM. In another study, a factored LM using POS information achieved the same results as the 4-gram LM (Kirchhoff and Yang, 2005). Syntax-based LMs were investigated in (Charniak et al.,

This work was partially financed by the European Commission under the FP6 Integrated Project TC-STAR.

2003), and reranking of translation hypothesis using structural properties in (Hasan et al., 2006).

An interesting experiment was reported at the NIST 2005 MT evaluation workshop (Och, 2005): starting with a 5-gram LM trained on 75 million words of Broadcast News data, a gain of about 0.5 point BLEU was observed each time when the amount of LM training data was doubled, using at the end 237 billion words of texts. Most of this additional data was collected by Google on the Internet. We believe that this kind of approach is difficult to apply to other tasks than Broadcast News and other target languages than English. There are many areas where automatic machine translation could be deployed and for which considerably less *appropriate in-domain* training data is available. We could for instance mention automatic translation of medical records, translation systems for tourism related tasks or even any task for which Broadcast news and Web texts is of limited help.

In this work, we consider the translation of European Parliament Speeches from Spanish to English, in the framework of an international evaluation organized by the European TC-STAR project in February 2006. The training data consists of about 35M words of aligned texts that are also used to train the target LM. In our experiments, adding more than 580M words of Broadcast News data had no impact on the BLEU score, despite a notable decrease of the perplexity of the target LM. Therefore, we suggest to use more complex statistical LMs that are expected to take better advantage of the limited amount of appropriate training data. Promising candidates are random forest LMs (Xu and Jelinek, 2004), random cluster LMs (Emami and Jelinek, 2005) and the neural network LM (Bengio et al., 2003). In this paper, we investigate whether the latter approach can be used in a statistical machine translation system.

The basic idea of the neural network LM, also called continuous space LM, is to project the word indices onto a continuous space and to use a probability estimator operating on this space. Since the resulting probability functions are smooth functions of the word representation, better generalization to unknown n-grams can be expected. A neural network can be used to simultaneously learn the projection of the words onto the continuous space and to estimate the n-gram probabilities. This is still a n-gram approach, but the LM posterior probabilities are "interpolated" for any possible context of length n-1 instead of backing-off to shorter contexts. This approach was successfully used in large vocabulary speech recognition (Schwenk and Gauvain, 2005), and we are interested here if similar ideas can be applied to statistical machine translation.

This paper is organized as follows. In the next section we first describe the baseline statistical machine translation system. Section 3 presents the architecture of the continuous space LM and section 4 summarizes the experimental evaluation. The paper concludes with a discussion of future research directions.

2 Statistical Translation Engine

A word-based translation engine is used based on the so-called IBM-4 model (Brown et al., 1993). A brief description of this model is given below along with the decoding algorithm.

The search algorithm aims at finding what target sentence $\mathbf{e}$ is most likely to have produced the observed source sentence $\mathbf{f}$. The translation model $\Pr(\mathbf{f}|\mathbf{e})$ is decomposed into four components:

1. a fertility model;
2. a lexical model of the form $t(f|e)$, which gives the probability that the target word e translates into the source word f;
3. a distortion model, that characterizes how words are reordered when translated;
4. and probabilities to model the insertion of source words that are not aligned to any target words.

An A* search was implemented to find the best translation as predicted by the model, when given enough time and memory, i.e., provided pruning did not eliminate it. The decoder manages partial hypotheses, each of which translates a subset of source words into a sequence of target words. Expanding a partial hypothesis consists of covering one extra source position (in random order) and, by doing so, appending one, several or possibly zero target words to its target word sequence. For details about the implemented algorithm, the reader is referred to (Déchelotte et al., 2006).

Decoding uses a 3-gram back-off target language model. Equivalent hypotheses are merged, and only the best scoring one is further expanded. The decoder generates a lattice representing the

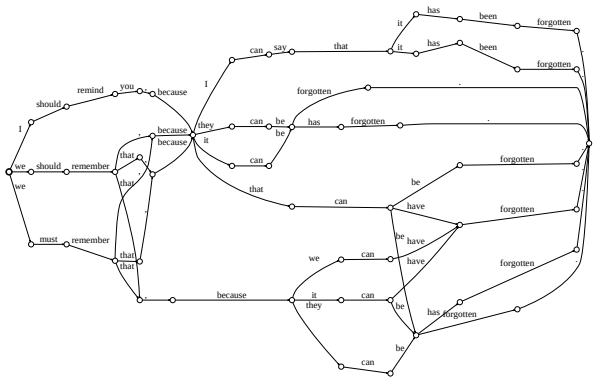

Figure 1: Example of a translation lattice. Source sentence: *"conviene recordarlo , porque puede que se haya olvidado ."*, Reference 1: *"it is appropriate to remember this , because it may have been forgotten ."* Reference 2: *"it is good to remember this , because maybe we forgot it ."*

explored search space. Figure 1 shows an example of such a search space, here heavily pruned for the sake of clarity.

2.1 Sentence Splitting

The execution complexity of our SMT decoder increases non-linearly with the length of the sentence to be translated. Therefore, the source text is split into smaller chunks, each one being translated separately. The chunks are then concatenated together. Several algorithms have been proposed in the literature that try to find the best splits, see for instance (Berger et al., 1996). In this work, we first split long sentences at punctuation marks, the remaining segments that still exceed the allowed length being split linearly. In a second pass, adjoining very short chunks are merged together.

During decoding, target LM probabilities of the type $\Pr(w_1|\texttt{<s>})$ and $\Pr(\texttt{</s>}|w_{n-1}w_n)$ will be requested at the beginning and at the end of the hypothesized target sentence respectively.[1] This is correct when a whole sentence is translated, but leads to wrong LM probabilities when processing smaller chunks. Therefore, we define a sentence break symbol, `<b>`, that is used at the beginning and at the end of a chunk. During decoding a 3-gram back-off LM is used that was trained on text where sentence break symbols have been added.

Each chunk is translated and a lattice is generated. The individual lattices are then joined, omitting the sentence break symbols. Finally, the resulting lattice is rescored with a LM that was trained on text *without* sentence breaks. In that way we find the best junction of the chunks. Section 4.1 provides comparative results of the different algorithms to split and join sentences.

2.2 Parameter Tuning

It is nowadays common practice to optimize the coefficients of the log-linear combination of feature functions by maximizing the BLEU score on the development data (Och and Ney, 2002). This is usually done by first creating n-best lists that are then reranked using an iterative optimization algorithm.

In this work, a slightly different procedure was used that operates directly on the translation lattices. We believe that this is more efficient than reranking n-best lists since it guarantees that always all possible hypotheses are considered. The decoder first generates large lattices using the current set of parameters. These lattices are then processed by a separate tool that extracts the best path, given the coefficients of six feature functions (translations, distortion, fertility, spontaneous insertion, target language model probability, and a sentence length penalty). Then, the BLEU score of the extracted solution is calculated. This tool is called in a loop by the public numerical optimization tool Condor (Berghen and Bersini, 2005). The solution vector was usually found after about 100 iterations. In our experiments, only two cycles of lattice generation and parameter optimization were necessary (with a very small difference in the BLEU score).

In all our experiments, the 4-gram back-off and the neural network LM are used to calculate language model probabilities that replace those of the default 3-gram LM. An alternative would be to define each LM as a feature function and to combine them under the log-linear model framework, using maximum BLEU training. We believe that this would not make a notable difference in our experiments since we do interpolate the individual LMs, the coefficients being optimized to minimize perplexity on the development data. However, this raises the interesting question whether the two criteria lead to equivalent performance. The result section provides some experimental evidence on this topic.

[1] The symbols `<s>` and `</s>` denote the begin and end of sentence marker respectively.

3 Continuous Space Language Models

The architecture of the neural network LM is shown in Figure 2. 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 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:

$$d_j = \tanh\left(\sum_l m_{jl}\, c_l + b_j\right) \tag{3}$$

$$o_i = \sum_j v_{ij}\, d_j + k_i \tag{4}$$

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

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 overfitting the training data (weight decay). The parameter β has to be determined experimentally. Training is done using a resampling algorithm (Schwenk and Gauvain, 2005).

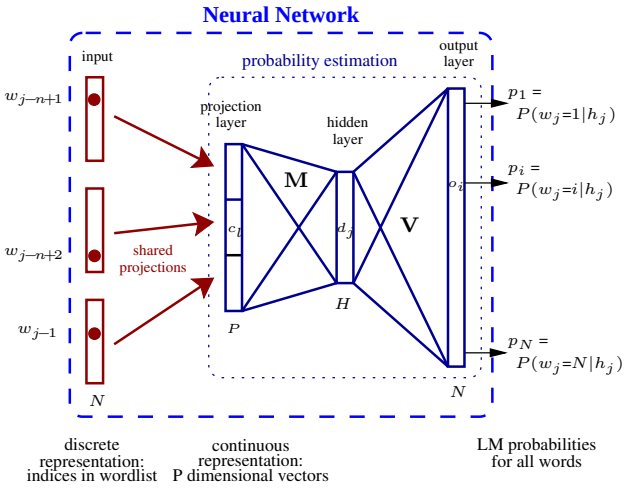

Figure 2: Architecture of the continuous space LM. h_j denotes the context $w_{j-n+1}, \ldots, w_{j-1}$. P is the size of one projection and H,N is the size of the hidden and output layer respectively. When short-lists are used the size of the output layer is much smaller then the size of the vocabulary.

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

The complexity to calculate one probability with this basic version of the neural network LM is quite high due to the large output layer. To speed up the processing several improvements were used (Schwenk, 2004):

1. *Lattice rescoring*: the statistical machine translation decoder generates a lattice using a 3-gram back-off LM. The neural network LM is then used to rescore the lattice.

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

3. *Efficient implementation*: collection of all LM probability requests with the same context h_t in one lattice, propagation of several examples at once through the neural network and utilization of libraries with CPU optimized matrix-operations.

The idea behind short-lists is to use the neural

network only to 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:

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

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

It can be considered that the neural network redistributes the probability mass of all the words in the short-list. This probability mass is precalculated and stored in the data structures of the back-off LM. A back-off technique is used if the probability mass for a input context is not directly available.

It was not envisaged to use the neural network LM directly during decoding. First, this would probably lead to slow translation times due to the higher complexity of the proposed LM. Second, it is quite difficult to incorporate n-gram language models into decoding, for n>3. Finally, we believe that the lattice framework can give the same performances than direct decoding, under the condition that the alternative hypotheses in the lattices are rich enough. Estimates of the lattice oracle BLEU score are given in the result section.

4 Experimental Evaluation

The experimental results provided here were obtained in the framework of an international evaluation organized by the European TC-STAR project[2] in February 2006. This project is envisaged as a long-term effort to advance research in all core technologies for speech-to-speech translation.

The main goal of this evaluation is to translate public European Parliament Plenary Sessions (EPPS). The training material consists of the minutes edited by the European Parliament in several languages, also known as the Final Text Editions (Gollan et al., 2005). These texts were aligned at the sentence level and they are used to train the statistical translation models (see Table 1 for some statistics). In addition, about 100h of Parliament plenary sessions were recorded and transcribed. This data is mainly used to train the speech recognizers, but the transcriptions were also used for the target LM of the translation system (about 740k words).

[2]http://www.tc-star.org/

	Spanish	*English*
Sentence Pairs	1.2M	
Total # Words	37.7M	33.8M
Vocabulary size	129k	74k

Table 1: Statistics of the parallel texts used to train the statistical machine translation system.

Three different conditions are considered in the TC-STAR evaluation: translation of the Final Text Edition (*text*), translation of the transcriptions of the acoustic development data (*verbatim*) and translation of speech recognizer output (*ASR*). Here we only consider the *verbatim* condition, translating from Spanish to English. For this task, the development data consists of 792 sentences (25k words) and the evaluation data of 1597 sentences (61k words). Parts of the test data origins from the Spanish parliament which results in a (small) mismatch between the development and test data. Two reference translations are provided. The scoring is case sensitive and includes punctuation symbols.

The translation model was trained on 1.2M sentences of parallel text using the Giza++ tool. All back-off LMs were built using modified Kneser-Ney smoothing and the SRI LM-toolkit (Stolcke, 2002). Separate LMs were first trained on the English EPPS texts (33.8M words) and the transcriptions of the acoustic training material (740k words) respectively. These two LMs were then interpolated together. Interpolation usually results in lower perplexities than training directly one LM on the pooled data, in particular if the corpora come from different sources. An EM procedure was used to find the interpolation coefficients that minimize the perplexity on the development data. The optimal coefficients are 0.78 for the Final Text edition and 0.22 for the transcriptions.

4.1 Performance of the sentence splitting algorithm

In this section, we first analyze the performance of the sentence split algorithm. Table 2 compares the results for different ways to translate the individual chunks (using a standard 3-gram LM versus an LM trained on texts with sentence breaks inserted), and to extracted the global solution (con-

LM used during decoding	*Concatenate 1-best*	*Lattice join*
Without sentence breaks	40.20	41.63
With sentence breaks	41.45	42.35

Table 2: BLEU scores for different ways to translate sentence chunks and to extract the global solution (see text for details).

catenating the 1-best solutions versus joining the lattices followed by LM rescoring). It can be clearly seen that joining the lattices and recalculating the LM probabilities gives better results than just concatenating the 1-best solutions of the individual chunks (first line: BLEU score of 41.63 compared to 40.20). Using a LM trained on texts with sentence breaks during decoding gives an additional improvement of about 0.7 points BLEU (42.35 compared to 41.63).

In our current implementation, the selection of the sentence splits is based on punctuation marks in the source text, but our procedure is compatible with other methods. We just need to apply the sentence splitting algorithm on the training data used to build the LM during decoding.

4.2 Using the continuous space language model

The continuous space language model was trained on exactly the same data than the back-off reference language model, using the resampling algorithm described in (Schwenk and Gauvain, 2005). In this work, we use only 4-gram LMs, but the complexity of the neural network LM increases only slightly with the order of the LM. For each experiment, the parameters of the log-linear combination were optimized on the development data.

Perplexity on the development data set is a popular and easy to calculate measure to evaluate the quality of a language model. However, it is not clear if perplexity is a good criterion to predict the improvements when the language model will be used in a SMT system. For information, and comparison with the back-off LM, Figure 3 shows the perplexities for different configurations of the continuous space LM. The perplexity clearly decreases with increasing size of the short-list and a value of 8192 was used. In this case, 99% of the requested LM probabilities are calculated by the neural network when rescoring a lattice.

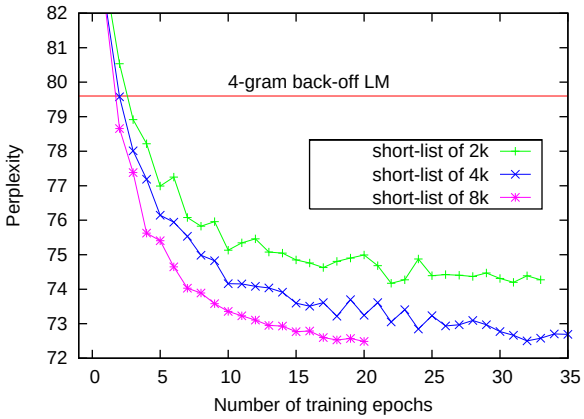

Figure 3: Perplexity of different configurations of the continuous space LM.

Although the neural network LM could be used alone, better results are obtained when interpolating it with the 4-gram back-off LM. It has even turned out that it was advantageous to train several neural network LMs with different context sizes[3] and to interpolate them altogether. In that way, a perplexity decrease from 79.6 to 65.0 was obtained. For the sake of simplicity we will still call this interpolation the neural network LM.

	Back-off LM		Neural LM
	3-gram	4-gram	4-gram
Perplexity	85.5	79.6	65.0
Dev data:			
BLEU	42.35	43.36	**44.42**
WER	45.9%	45.1%	44.4%
PER	31.8%	31.3%	30.8%
Eval data:			
BLEU	39.77	40.62	**41.45**
WER	48.2%	47.4%	46.7%
PER	33.6%	33.1%	32.8%

Table 3: Result comparison for the different LMs. BLEU uses 2 reference translations. WER=word error rate, PER=position independent WER.

Table 3 summarizes the results on the development and evaluation data. The coefficients of the feature functions are always those optimized on the development data. The joined translation lattices were rescored with a 4-gram back-off and the neural network LM. Using a 4-gram back-off LM gives an improvement of 1 point BLEU

[3]The values are in the range 150. . .400. The other parameters are: H=500, β=0.00003 and the initial learning rate was 0.005 with an exponential decay. The networks were trained for 20 epochs through the training data.

Spanish:	*es el Ã°nico premio Sajarov que no ha podido recibir su premio despuÃ©s de mÃ¡s de tres mil quinientos dÃas de cautiverio .*
Backoff LM:	*it is only the Sakharov Prize has not been able to receive the prize after three thousand , five days of detention .*
CSLM :	*it is the only Sakharov Prize has not been able to receive the prize after three thousand five days of detention .*
Reference 1:	*she is the only Sakharov laureate who has not been able to receive her prize after more than three thousand five hundred days in captivity .*
Reference 2:	*she is the only Sacharov prizewinner who couldn't yet pick up her prize after more than three thousand five hundred days of imprisonment .*

Figure 4: Example translation using the back-off and the continuous space language model (CSLM).

on the Dev data (+0.8 on Test set) compared to the 3-gram back-off LM. The neural network LM achieves an additional improvement of 1 point BLEU (+0.8 on Test data), on top of the 4-gram back-off LM. Small improvements of the word error rate (WER) and the position independent word error rate (PER) were also observed.

As usually observed in SMT, the improvements on the test data are smaller than those on the development data which was used to tune the parameters. As a rule of thumb, the gain on the test data is often half as large as on the Dev-data. The 4-gram back-off and neural network LM show both a good generalization behavior.

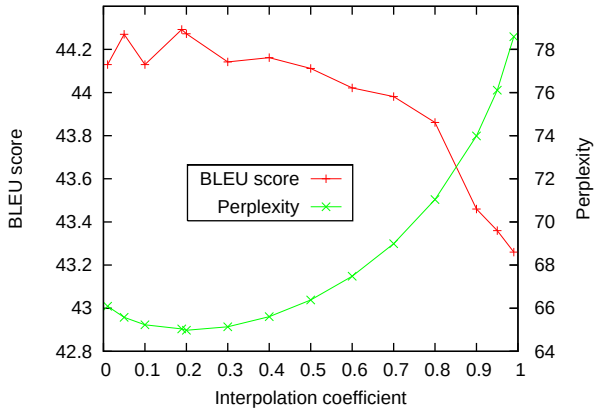

Figure 5: BLEU score and perplexity in function of the interpolation coefficient of the back-off 4-gram LM.

Figure 5 shows the perplexity and the BLEU score for different interpolation coefficients of the 4-gram back-off LM. For a value of 1.0 the back-off LM is used alone, while only the neural network LMs are used for a value of 0.0. Using an EM procedure to minimize perplexity of the interpolated model gives a value of 0.189. This value also seems to correspond to the best BLEU score.

This is a surprising result, and has the advantage that we do not need to tune the interpolation coefficient in the framework of the log-linear feature function combination. The weights of the other feature functions were optimized separately for each experiment. We noticed a tendency to a slightly higher weight for the continuous space LM and a lower sentence length penalty.

In a contrastive experiment, the LM training data was substantially increased by adding 352M words of commercial Broadcast News data and 232M words of CNN news collected on the Internet. Although the perplexity of the 4-gram back-off LM decreased by 5 points to 74.1, we observed no change in the BLEU score. In order to estimate the oracle BLEU score of the lattices we build a 4-gram back-off LM on the development data. Lattice rescoring achieved a BLEU score of 59.10.

There are many discussions about the BLEU score being or not a meaningful measure to assess the quality of an automatic translation system. It would be interesting to verify if the continuous space LM has an impact when human judgments of the translation quality are used, in particular with respect to fluency. Unfortunately, this is not planed in the TC-STAR evaluation campaign, and we give instead an example translation (see Figure 4). In this case, two errors were corrected (insertion of the word "the" and deletion of the comma).

5 Conclusion and Future Work

Some SMT decoders have an execution complexity that increases rapidly with the length of the sentences to be translated, which are usually split

into smaller chunks and translated separately. This can lead to translation errors and bad modeling of the LM probabilities of the words at both ends of the chunks. We have presented a lattice joining and rescoring approach that obtained significant improvements in the BLEU score compared to simply concatenating the 1-best solutions of the individual chunks. The task considered is the translation of European Parliament Speeches in the framework of the TC-STAR project.

We have also presented a neural network LM that performs probability estimation in a continuous space. Since the resulting probability functions are smooth functions of the word representation, better generalization to unknown n-grams can be expected. This is particularly interesting for tasks where only limited amounts of appropriate LM training material are available, but the proposed LM can be also trained on several hundred millions words. The continuous space LM is used to rescore the translation lattices. We obtained an improvement of 0.8 points BLEU on the test data compared to a 4-gram back-off LM, which itself had already achieved the same improvement in comparison to a 3-gram LM.

The results reported in this paper have been obtained with a word based SMT system, but the continuous space LM can also be used with a phrase-based system. One could expect that the target language model plays a different role in a phrase-based system since the phrases induce some local coherency on the target sentence. This will be studied in the future. Another promising direction that we have not yet explored, is to build long-span LM, i.e. with n much greater than 4. The complexity of our approach increases only slightly with n. Long-span LM could possibly improve the word-ordering of the generated sentence if the translation lattices include the correct paths.

References

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

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

Frank Vanden Berghen and Hugues Bersini. 2005. 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.

P. Brown, S. Della Pietra, Vincent J. Della Pietra, and R. Mercer. 1993. The mathematics of statistical machine translation. *Computational Linguistics*, 19(2):263–311.

E. Charniak, K. Knight, and K. Yamada. 2003. Syntax-based language models for machine translation. In *Machine Translation Summit*.

Daniel Déchelotte, Holger Schwenk, and Jean-Luc Gauvain. 2006. The 2006 LIMSI statistical machine translation system for TC-STAR. In TC-STAR *Speech to Speech Translation Workshop, Barcelona*.

Ahmad Emami and Frederick Jelinek. 2005. Random clusterings for language modeling. In *ICASSP*, pages I:581–584.

C. Gollan, M. Bisani, S. Kanthak, R. Schlueter, and H. Ney. 2005. Cross domain automatic transcription on the TC-STAR EPPS corpus. In *ICASSP*.

Sasa Hasan, Olivier Bender, and Hermann Ney. 2006. Reranking translation hypothesis using structural properties. In *LREC*.

Katrin Kirchhoff and Mei Yang. 2005. Improved language modeling for statistical machine translation. In *ACL'05 workshop on Building and Using Parallel Text*, pages 125–128.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL*, pages 295–302, University of Pennsylvania.

F.-J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *NAACL*, pages 161–168.

Franz-Joseph Och. 2005. The Google statistical machine translation system for the 2005 Nist MT evaluation, Oral presentation at the 2005 Nist MT Evaluation workshop, June 20.

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

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

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *ICSLP*, pages II: 901–904.

Peng Xu and Frederick Jelinek. 2004. Random forest in language modeling. In *EMNLP*, pages 325–332.