The 2006 LIMSI Statistical Machine Translation System for TC-STAR

Daniel Déchelotte, Holger Schwenk and Jean-Luc Gauvain

LIMSI-CNRS

BP 133, 91403 Orsay, France

{dechelot,schwenk,gauvain}@limsi.fr

Abstract

This paper presents the LIMSI statistical machine translation system developed for 2006 TC-STAR evaluation campaign. We describe an A*-decoder that generates translation lattices using a word-based translation model. A lattice is a rich and compact representation of alternative translations that includes the probability scores of all the involved sub-models. These lattices are then used in subsequent processing steps, in particular to perform sentence splitting and joining, maximum BLEU training and to use improved statistical target language models.

1. Introduction

Statistical machine translation (SMT) has gained much attention within the research community during the last years. This is in part due to its robustness to erroneous sentences, for example those provided by an automatic speech recognition system, and to the good results that have been achieved in international evaluations. Most of the state-of-the-art systems use context in the translation model, either using phrases (Zens et al., 2002; Koehn et al., 2003) or bilingual n-gram tuplets (Marioño et al., 2005). In this work, we only use a word-based translation model in order to study several extensions that are not yet used in phrase-based systems, to the best of our knowledge.

In automatic speech recognition (ASR) it is today common practice to produce lattices as a compact representation of plausible hypotheses. Those lattices are then used for discriminant training, improved acoustic model adaptation, consensus network decoding or language model rescoring. In ASR, lattice processing generally achieves better results than using n-best lists. We argue that it should be promising to also apply the lattice processing framework to SMT. We have implemented an SMT decoder that generates as output a rich lattice where for each word all the individual feature function scores are available. In that way, we hope to perform different post-processing steps or the incorporation of additional knowledge sources without the need of several full decoding steps. This lattice framework is used for an efficient sentence splitting and joining algorithm, fast maximum BLEU training and lattice rescoring using improved target language models. In particular, we present a neural network language model (LM) that takes better advantage of the limited amount of training material.

The experimental results provided here were obtained in the framework of the second international evaluation organized by the European TC-STAR project. The main goal of this evaluation is to translate public European Parliament Plenary Sessions. Three different conditions are considered in the TC-STAR evaluation: translation of the official minutes edited by the European Parliament (*text*), translation of the transcriptions of the acoustic training data (*verbatim*) and translation of speech recognizer output (*ASR*). Here we consider the *verbatim* and *ASR* conditions, translating from Spanish to English only.

This paper is organized as follows. In the next section we first describe the baseline statistical machine translation system, giving details on the search algorithm and the lattice generation. Section 3. presents the used lattice operations and section 4. summarizes the experimental evaluation. The paper concludes with a discussion of future research directions.

2. SMT Decoder

The translation engine relies on the so-called IBM-4 word-based model (Brown et al., 1993). A brief description of this model is given below along with the decoding algorithm. The translation problem is to find a target sentence $\mathbf{e} = e_1 \dots e_J$ that is a valid translation of a source sentence $\mathbf{f} = f_1 \dots f_I$. Given the fact that the translation models are largely imperfect and are not guaranteed to produce a grammatical target sentence, the Bayes relation is classically used:

$$\underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e}) \qquad (1)$$

Using a generative terminology, the search problem is hence reversed and aims at finding what target sentence $\mathbf{e}$ is most likely to have produced the source sentence $\mathbf{f}$. The translation model $P(\mathbf{f}|\mathbf{e})$ relies on four model components:

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

The target language model $\Pr(\mathbf{e})$ and the four translation submodels are combined in a log-linear fashion (Och and Ney, 2002), the combination coefficients being optimized on the development set.

2.1. Search algorithm

An A* search is performed so as to find the translation predicted by the model. 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. The various hypotheses expansion operators are described below. An admissible heuristics is used to reduce the explored search space and is described at subsection 2.2. Details on hypotheses management and pruning are given at subsection 2.3. Some aspects of our decoder are inspired by work described in (Germann et al., 2001) and (Ortiz et al., 2003).

Operator "Append"

The "Append" operator appends exactly one target word and covers exactly one source word; it generates an edge as depicted at Figure 1. Partial costs include the translation cost (T) and the distortion cost (D) and the target language model cost (L). The fertility cost for the just inserted target word is not known yet, as it may have covered more than one source word (see the "Extend" operator below). However, the fertility cost for the previous target word e_{-1} is now known and is included (F).

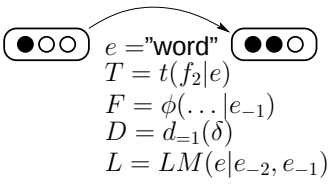

Figure 1: The "Append" operator

Operator "InsertNZFertAndAppend"

The "InsertNZFertAndAppend" operator appends up to N zero-fertility target words (that cover no source words) followed by one target word and covers exactly one source word. This operator is required to insert non-content words, which are typically handled by phrases in phrase-based systems. For each target word e, a short list of up to 20 target words that are both likely to precede e and to be of fertility 0 is computed at training time. In practice, N is limited to 1. Figure 2 shows an example. In comparison to the "Append" operator, the edges holds two additional costs: the target language model and the null fertility costs for the inserted target word.

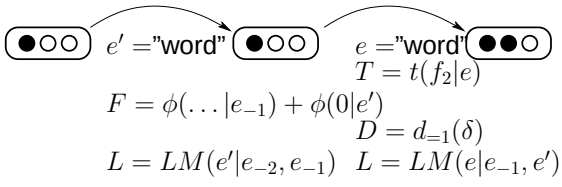

Figure 2: The "InsertNZFertAndAppend" operator

Operator "Extend"

The "Extend" operator covers one source word by extending the last target word, thus inserting no target word. Instead, the running fertility of e is incremented. Figure 3 shows in particular the different distortion cost.

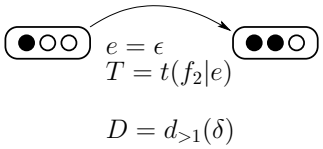

Figure 3: The "Extend" operator

Operator "CoverWithE0"

The "CoverWithE0" operator only applies when at least half of the source words are already translated. It aligns all uncovered source words to the null target word e_0, without producing any "real" target words. It is the only operator that produces a complete translation hypothesis and is therefore mandatory (even if all source positions are covered by regular target words) because it specifies the spontaneous insertion cost (S) and the final target language model cost. Figure 4 shows an example where the last source word is aligned to e_0. Only with this expansion operator, a spontaneous cost is computed as follows:

$$S = \binom{\phi_0}{J-\phi_0} p_0^{J-\phi_0} p_1^{\phi_0} \prod_{j \text{ uncovered}} t(f_j|e_0) \qquad (2)$$

where J is the source sentence length and ϕ_0 the number of yet uncovered source words.

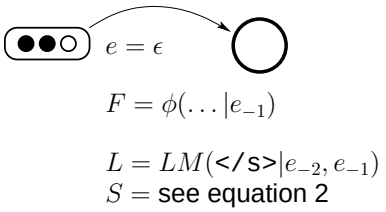

Figure 4: The "CoverWithE0" operator. It leads to the terminal node of the translation graph.

Lattice generation

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 explored search space. Figure 5 shows an example of such a search space, here heavily pruned for the sake of clarity.

2.2. Future cost heuristics

When considering which partial hypothesis should be expanded at each iteration, one possibility is to only take into account the various costs accumulated along the path, in the translation graph, that leads to each partial hypothesis. It is however well known that estimating the future costs can lead to faster decoding and even better results, by positively

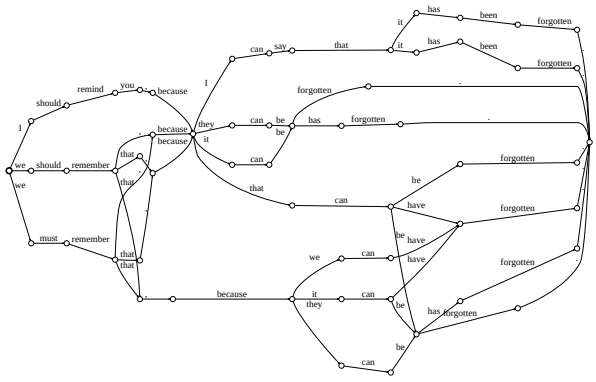

Figure 5: Example of a translation lattice. Source sentence: *"conviene recordarlo , porque puede que se haya olvidado ."* Reference 1: *"it is appropriate to remember this , because it may have been forgotten ."* Reference 2: *"it is good to remember this , because maybe we forgot it ."* The scores on each arc of the various statistical models are not shown for clarity.

influencing pruning. The heuristics used in the decoder is modeled after (Och et al., 2001).
The described heuristics is said to be admissible, meaning that it always overestimates the future probabilities (i.e., underestimates the costs of future expansions), in order to retain the optimality of the search. For each source word f at position i, an upper bound P_i to the probability of covering f is evaluated as follows. On the one hand, f may be covered by the null target word e_0 (which corresponds to no actual target word), with the probability $P_i^0 = t(f|e_0)$. On the other hand, if f is to be translated by a (fertile) target word e, the lexical, fertility and distortion submodels should be accounted for. An upper bound to the probability P_i^+ of covering f with an actual target word is:

$$P_i^+ = \max_{e,\phi} t(f|e) \sqrt[\phi]{f(\phi|e)} d_{\max} \quad (3)$$

where $d_{\max}$ is constant: it is the maximum value the distortion model can take. Finally, P_i is obtained as the minimum of P_i^0 and P_i^+:

$$P_i = \min\left(P_i^0, P_i^+\right) \quad (4)$$

2.3. Hypotheses management

Partial hypotheses are stored in several queues so as to easily compare hypotheses. For a source sentence of length J, 2^J queues are created, one per subset of source positions. It is consequently possible to keep those queues relatively small (usual size is 10, a size of 20 gives results marginally different from an infinite stack size) since hypotheses only "compete" with hypotheses that cover the exact same set of source words. Also, given the form of the future cost heuristics (see previous section), all hypotheses from a given queue share the heuristic future cost, which is computed once at queue creation time.
The histogram pruning due to the limited size of the queues is the only one that takes place during decoding. It allows decoding of a 10-word long sentence in two seconds on average, with decoding time almost doubling for each extra source word.

3. Lattice Processing

The lattice framework is used to perform the following operations: sentence splitting and joining, maximum BLEU training and the incorporation of improved statistical target language models. These operations are detailed in the following subsections.

3.1. Sentence splitting and joining

In the 2006 TC-STAR development corpus sentences proved to be of almost arbitrary length (the maximum length was 222 words in one segment). Our decoder is not able to cope with such long sentences and sentence splitting has to occur before the actual translation. Sentences are thus split into "chunks" of a maximum length of 16 words, as a compromise between acceptable performance degradation and tractable translation times. Splitting long sentences may lead to suboptimal translation for two reasons:

1. word-reordering is not possible across split points;
2. translation of each chunk does not take into account source and target texts of adjacent chunks.

To lessen the effect of the first point, sentences are preferably split at punctuation marks. However, if further splitting is necessary, uniform splitting is applied (so as to maximize the minimal length of the resulting chunks).

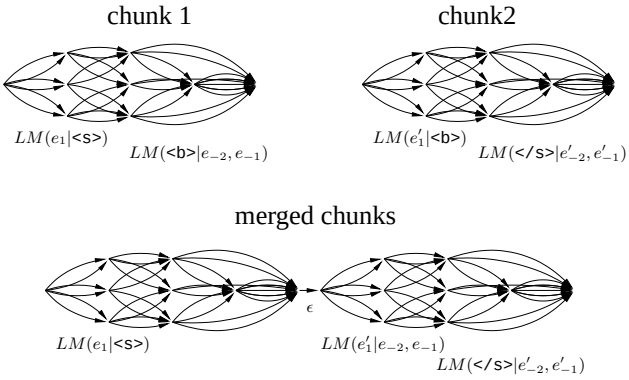

Figure 6: Separate decoding of two chunks, followed by translation graph fusion (with LM probabilities reestimation).

As for the second point, target LM probabilities of the type $\Pr(w_1|\texttt{<s>})$ and $\Pr(\texttt{</s>}|w_{n-1}w_n)$ are usually 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 propose to use a sentence break symbol, <b>, that is used at the beginning and at the end of a chunk (see Figure 6). The 3-gram

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

back-off LM used during decoding has been 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, suppressing the sentence break symbols. Finally the resulting lattice is rescored with a LM that has been trained *without* sentence breaks. In this way the best junction of the chunks is found. The results section provides comparative results of the different algorithms to split and join sentences.

3.2. Maximum BLEU training

As it is common practice, each of the translation submodels and the target language model are weighted in a log-linear fashion. In addition to a word-penalty feature, this amounts to six feature functions. The optimization criterion is the BLEU score on the development set.
First lattices are generated for all sentences of the development set using a default weighting of the individual feature functions. Then, a target sentence is extracted from each lattice using a separate tool, and the BLEU score is calculated. This tool is called in a loop in order to find the optimal parameter values that maximize the BLEU score, as depicted in Figure 7. The publicly available tool Condor (Berghen and Bersini, 2005) is used to perform a multivariate search on the parameters. When inappropriate default parameters for the initial decode were chosen, it may be necessary to repeat the whole procedure, i.e. lattice generation with better parameters, followed by Condor tuning.

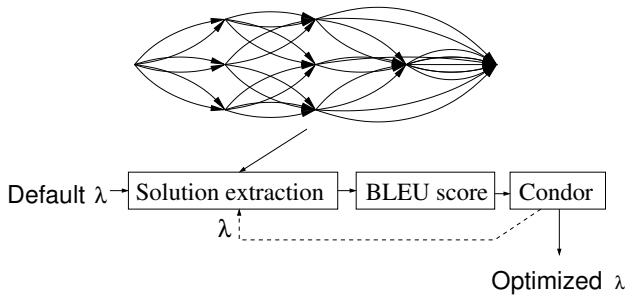

Figure 7: Setup to tune free parameters on development set.

This lattice procedure is comparable to the usual n-best list processing, but we believe that the lattice frame work is more efficient. First, the evaluation of many different hypothesis is faster since lattices have usually less redundancy than n-best lists. Second, at most two decode and parameter optimization cycles were needed, while usually several ones may be necessary when using n-best list rescoring. Finally, it seems easier to use different error functions in a lattice processing frame work, for instance minimum Bayes-risk decoding (Kumar and Byrne, 2004).

3.3. Improved Target Language Models

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. In addition to the standard feature functions many others were 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.
In the TC-STAR evaluation only about 35M words of *in-domain* data are available to train the target LM. We suggest to use more complex statistical LMs that are expected to take better advantage of the limited amount of appropriate training data, in particular a neural network LM (Bengio et al., 2003). 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. The neural network LM is used to rescore the 3-gram translation lattices. More details on this approach can be found in (Schwenk et al., 2006).

4. Experimental Results

In the 2006 TC-STAR SLT evaluation, the training material consists of the minutes edited by the European Parliament in several languages, also known as the EPPS 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. 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).
reference translations are provided. Table 1 gives some statistics about the data. Several normalization and preprocessing steps were performed onto the EPPS FTE parallel texts in order to match the style of the verbatim or ASR condition, in particular spelling out of numbers.

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

Table 1: Statistics of the parallel texts (EPPS Final Text Edition) used to train the SMT system.

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.[2] An EM procedure was used to find the interpola-

[2] Interpolation gives usually better results than training a LM on the pooled data.

tion 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 extract the global solution (concatenating 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).

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 (Verbatim Dev data).

In our current implementation, the selection of the sentence splits is based on punctuation marks in the source text. Our procedure is however compatible with other methods, as long as the sentence split algorithm can be applied on the LM training data so as to train the LM used during decoding on text with sentence split markers. This will be investigated in a future version of our statistical machine translation system.

4.2. Using the neural network language model

The 4-gram neural network 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). For each experiment, the parameters of the log-linear combination were optimized on the development data. Although the neural network LM could be used alone, better results are obtained when interpolating it with the 4-gram back-off LM. It even turned out advantageous to train several neural network LMs with different context sizes and to interpolate them altogether. For the sake of simplicity we will still call this interpolation the neural network LM. More details on the architecture and the training procedure are given in (Schwenk et al., 2006). Table 3 summarizes the perplexities on the development data of the different LMs and the impact on the translation quality when they are used to rescore the lattices.

Using a 4-gram back-off LM gives an improvement of 1 point BLEU compared to a 3-gram back-off LM. The neural network LM achieves an additional improvement of 1 point BLEU, on top of the 4-gram back-off LM. Small improvements of the word error rate (mWER) and the position independent word error rate (mPER) were also observed. 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 our lattices, we built a 4-gram back-off LM on the development data. Lattice rescoring achieved then a BLEU score of 59.10.

	Back-off LM		Neural LM
	3-gram	4-gram	4-gram
Perplexity	85.5	79.6	65.0
BLEU	42.35	43.36	**44.42**
mWER	45.9%	45.1%	44.4%
mPER	31.8%	31.3%	30.8%

Table 3: Result comparison when using different LMs to rescore the translation lattices (Verbatim Dev data). mWER=word error rate, mPER=position independent WER, both using multiple references.

4.3. Translation of speech input

The translation was performed on a ROVER system combination of all the ASR systems that had participated in the TC-STAR evaluation. The word error rate is 7.1% and 6.9% on the development and test data, respectively. When translating speech input, several specificities must be addressed. First, the standard MT quality measures like BLEU or the NIST score suppose that for each reference sentence an hypothesis is generated. This can not be guaranteed when translating speech input since the reference segmentation is not known. In the 2006 TC-STAR evaluation, automatic segmentation of the speech signal was used and the scoring of the MT system was performed by finding automatically the best alignment of the produced hypothesis with the reference translations (Matusov et al., 2006).

Second, case and punctuation are traditionally not taken into account when measuring ASR accuracy, due to the difficulty to come up with a gold standard for these matters. Translation systems, however, are trained on data with case information and rich punctuations, and applying those systems on raw ASR output leads to a mismatch between training and testing conditions. The ROVER input provided to translation systems contains some case information, as well

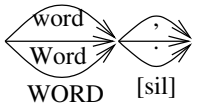

Figure 8: Lattice fragment generated for each word in ROVER input.

as some minimal punctuation (some final points, when an end of sentence is detected). In order to minimize the mismatch between the ROVER input and the training texts, the input was preprocessed in the following way: Each word was replaced by a fragment of word lattice of the form of Figure 8. Segment boundaries were then inserted at pauses greater than 80 ms, and optional ones at other pauses. Last, the resulting lattice was rescored using a regular 4-gram LM, with parameters adapted so as to reproduce the number of punctuation signs in the development data.
Finally, close coupling of speech transcription and translation can be envisaged using n-best lists or lattices. This was not used in this evaluation given the low ASR word error rate. We simply translated the 1-best ASR hypothesis. The results for the ASR condition are given in Table 4.

	Back-off LM		Neural LM
	3-gram	4-gram	4-gram
BLEU	32.70	33.74	**34.35**

Table 4: Result on the Dev data for the ASR condition.

For this condition, the improvement brought be the neural network LM is smaller (0.6 points BLEU). This might be explained by the fact that there are more grammatical errors in the target sentences (due to the error in the source sentences produced by the ASR module), which complicates the task of the target language model.

4.4. Evaluation results

Table 5 summarizes the results obtained on the evaluation data using the same coefficients of the log-linear feature function combination than for the development set.

	Back-off LM		Neural LM
	3-gram	4-gram	4-gram
Verbatim, BLEU	39.77	40.62	**41.45**
mWER	48.2%	47.4%	46.7%
mPER	33.6%	33.1%	32.8%
ASR, BLEU	31.50	31.40	31.86

Table 5: Performance on the evaluation data.

As usually observed in SMT systems, the improvements obtained on the evaluation data are smaller than those obtained on the Dev data. Using the 4-gram increases the BLEU score by 0.85 points (+1.0 on Dev), and the neural network LM achieves an improvement of 0.83 points BLEU (+1 on Dev). The results on the evaluation data for the ASR condition are somewhat surprising: the 4-gram achieves worse results than the 3-gram due to a large brevity penalty (0.922 for the 4-gram and 0.955 for the 3-gram).

5. Conclusion

We have described the LIMSI statistical machine translation system developed for the 2006 TC-STAR evaluation. Our decoder creates rich lattices that are used to perform sentence splitting and joining, maximum BLEU training and to use a neural network target language model. These techniques achieved improvements in the BLEU score of several points. Although our system uses only word-based translation models, it achieved performances that are close to those of phrase-based systems.
Future work will concentrate on different techniques to use context dependent translation models and on the incorporation of higher-level knowledge sources.

6. 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.

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.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast decoding and optimal decoding for machine translation. In *ACL*, pages 228–235.

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

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *HLT*.

S. Kumar and W. Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *HLT*, pages 169–176.

J. Marioño, R. Blanchs, J. Crego, A. de Gispert, P. Lambert, M. R. Costa-jussà, and J. Fonollosa. 2005. Bilingual n-gram statistical machine translation. In *MT-Summit X*.

E. Matusov, N. Ueffing, and H. Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *EACL*.

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.

Franz Jospeh Och, N. Ueffing, and Hermann Ney. 2001. An efficient A* search algorithm for statistical machine translation". In *ACL*, pages 55–62.

D. Ortiz, I. García-Varea, and F. Casacuberta. 2003. An empirical comparison of stack-based decoding algorithms for statistical machine translation. In *Iberian Conference on Pattern Recognition and Image Analysis, IbPRIA*, pages 654–663.

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

Holger Schwenk, Daniel Déchelotte, and Jean-Luc Gauvain. 2006. Continuous space language models for statistical machine translation. In *Coling/ACL*.

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

R. Zens, Franz Josef Och, and Hermann Ney. 2002. Phrased-based statistical machine translation. In *German Conference on Artificial Intelligence (KI 2002)*, pages 18–32.