# Jane: Open Source Hierarchical Translation, Extended with Reordering and Lexicon Models

**David Vilar, Daniel Stein, Matthias Huck and Hermann Ney**
Lehrstuhl für Informatik 6
RWTH Aachen University
Aachen, Germany
{vilar,stein,huck,ney}@cs.rwth-aachen.de

## Abstract

We present Jane, RWTH's hierarchical phrase-based translation system, which has been open sourced for the scientific community. This system has been in development at RWTH for the last two years and has been successfully applied in different machine translation evaluations. It includes extensions to the hierarchical approach developed by RWTH as well as other research institutions. In this paper we give an overview of its main features.

We also introduce a novel reordering model for the hierarchical phrase-based approach which further enhances translation performance, and analyze the effect some recent extended lexicon models have on the performance of the system.

## 1 Introduction

We present a new open source toolkit for hierarchical phrase-based translation, as described in (Chiang, 2007). The hierarchical phrase model is an extension of the standard phrase model, where the phrases are allowed to have "gaps". In this way, long-distance dependencies and reorderings can be modelled in a consistent way. As in nearly all current statistical approaches to machine translation, this model is embedded in a log-linear model combination.

RWTH has been developing this tool during the last two years and it was used successfully in numerous machine translation evaluations. It is developed in C++ with special attention to clean code, extensibility and efficiency. The toolkit is available under an open source non-commercial license and downloadable from `http://www.hltpr.rwth-aachen.de/jane`.

In this paper we give an overview of the main features of the toolkit and introduce two new extensions to the hierarchical model. The first one is an additional reordering model inspired by the reordering widely used in phrase-based translation systems and the second one comprises two extended lexicon models which further improve translation performance.

## 2 Related Work

Jane implements many features presented in previous work developed both at RWTH and other groups. As we go over the features of the system we will provide the corresponding references.

Jane is not the first system of its kind, although it provides some unique features. There are other open source hierarchical decoders available. These include

- SAMT (Zollmann and Venugopal, 2006): The original version is not maintained any more and we had problems working on big corpora. A new version which requires Hadoop has just been released, however the documentation is still missing.

- Joshua (Li et al., 2009): A decoder written in Java by the John Hopkins University. This project is the most similar to our own, however both were developed independently and each one has some unique features. A brief comparison between these two systems is included in Section 5.1.

- Moses (Koehn et al., 2007): The de-facto standard phrase-based translation decoder has now been extended to support hierarchical translation. This is still in an experimental branch, however.

## 3 Features

In this section we will only give a brief overview of the features implemented in Jane. For detailed explanation of previously published algorithms and methods, we refer to the given literature.

### 3.1 Search Algorithms

The search for the best translation proceeds in two steps. First, a monolingual parsing of the input sentence is carried out using the CYK+ algorithm (Chappelier and Rajman, 1998), a generalization of the CYK algorithm which relaxes the requirement for the grammar to be in Chomsky normal form. From the CYK+ chart we extract a hypergraph representing the parsing space.

In a second step the translations are generated, computing the language model scores in an integrated fashion. Both the cube pruning and cube growing algorithms (Huang and Chiang, 2007) are implemented. For the latter case, the extensions concerning the language model heuristics similar to (Vilar and Ney, 2009) have also been included.

### 3.2 Language Models

Jane supports four formats for $n$-gram language models:

- The ARPA format for language models. We use the SRI toolkit (Stolcke, 2002) to support this format.

- The binary language model format supported by the SRI toolkit. This format allows for a more efficient language model storage, which reduces loading times. In order to reduce memory consumption, the language model can be reloaded for every sentence, filtering the $n$-grams that will be needed for scoring the possible translations. This format is specially useful for this case.

- Randomized LMs as described in (Talbot and Osborne, 2007), using the open source implementation made available by the authors of the paper. This approach uses a space efficient but approximate representation of the set of $n$-grams in the language model. In particular the probability for unseen $n$-grams may be overestimated.

- An in-house, exact representation format with on-demand loading of $n$-grams, using the internal prefix-tree implementation which is also used for phrase storage (see also Section 3.9).

Several language models (also of mixed formats) can be used during search. Their scores are combined in the log-linear framework.

### 3.3 Syntactic Features

Soft syntactic features comparable to (Vilar et al., 2008) are implemented in the extraction step of the toolkit. In search, they are considered as additional feature functions of the translation rules.

The decoder is able to handle an arbitrary number of non-terminal symbols. The extraction has been extended so that the extraction of SAMT-rules is included (Zollmann and Venugopal, 2006) but this approach is not fully supported (there may be empty parses due to the extended number of non-terminals). We instead opted to support the generalization presented in (Venugopal et al., 2009), where the information about the new non-terminals is included as an additional feature in the log-linear model.

In addition, dependency information in the spirit of (Shen et al., 2008) is included. Jane features models for string-to-dependency language models and computes various scores based on the well-formedness of the resulting dependency tree.

Jane supports the Stanford parsing format,[1] but can be easily extended to other parsers.

### 3.4 Additional Reordering Models

In the standard formulation of the hierarchical phrase-based translation model two additional rules are added:

$$\begin{aligned} S &\rightarrow \langle S^{\sim 0} X^{\sim 1}, S^{\sim 0} X^{\sim 1} \rangle \\ S &\rightarrow \langle X^{\sim 0}, X^{\sim 0} \rangle \end{aligned} \tag{1}$$

This allows for a monotonic concatenation of phrases, very much in the way monotonic phrase-based translation is carried out.

It is a well-known fact that for phrase-based translation, the use of additional reordering models is a key component, essential for achieving good translation quality. In the hierarchical model, the reordering is already integrated in the translation formalism, but there are still cases where the required reorderings are not captured by the hierarchical phrases alone.

The flexibility of the grammar formalism allows us to add additional reordering models without the need to explicitely modify the code for supporting them. The most straightforward example would

[1] http://nlp.stanford.edu/software/lex-parser.shtml

be to include the ITG-Reorderings (Wu, 1997), by adding following rule

$$S \rightarrow \langle S^{\sim 0} S^{\sim 1}, S^{\sim 1} S^{\sim 0} \rangle \tag{2}$$

We can also model other reordering constraints. As an example, phrase-level IBM reordering constraints with a window length of 1 can be included substituting the rules in Equation (1) with following rules

$$\begin{aligned}
S &\rightarrow \langle M^{\sim 0}, M^{\sim 0} \rangle \\
S &\rightarrow \langle M^{\sim 0} S^{\sim 1}, M^{\sim 0} S^{\sim 1} \rangle \\
S &\rightarrow \langle B^{\sim 0} M^{\sim 1}, M^{\sim 1} B^{\sim 0} \rangle \\
M &\rightarrow \langle X^{\sim 0}, X^{\sim 0} \rangle \\
M &\rightarrow \langle M^{\sim 0} X^{\sim 1}, M^{\sim 0} X^{\sim 1} \rangle \\
B &\rightarrow \langle X^{\sim 0}, X^{\sim 0} \rangle \\
B &\rightarrow \langle B^{\sim 0} X^{\sim 1}, X^{\sim 1} B^{\sim 0} \rangle
\end{aligned} \tag{3}$$

In these rules we have added two additional non-terminals. The $M$ non-terminal denotes a *monotonic block* and the $B$ non-terminal a *back jump*. Actually both of them represent monotonic translations and the grammar could be simplified by using only one of them. Separating them allows for more flexibility, e.g. when restricting the jump width, where we only have to restrict the maximum span width of the non-terminal $B$. These rules can be generalized for other reordering constraints or window lengths.

Additionally distance-based costs can be computed for these reorderings. To the best of our knowledge, this is the first time such additional reorderings have been applied to the hierarchical phrase-based approach.

## 3.5 Extended Lexicon Models

We enriched Jane with the ability to score hypotheses with discriminative and trigger-based lexicon models that use global source sentence context and are capable of predicting context-specific target words. This approach has recently been shown to improve the translation results of conventional phrase-based systems. In this section, we briefly review the basic aspects of these extended lexicon models. They are similar to (Mauser et al., 2009), and we refer there for a more detailed exposition on the training procedures and results in conventional phrase-based decoding.

Note that the training for these models is not distributed together with Jane.

### 3.5.1 Discriminative Word Lexicon

The first of the two lexicon models is denoted as *discriminative word lexicon* (DWL) and acts as a statistical classifier that decides whether a word from the target vocabulary should be included in a translation hypothesis. For that purpose, it considers all the words from the source sentence, but does not take any position information into account, i.e. it operates on sets, not on sequences or even trees. The probability of a word being part of the target sentence, given a set of source words, are decomposed into binary features, one for each source vocabulary entry. These binary features are combined in a log-linear fashion with corresponding feature weights. The discriminative word lexicon is trained independently for each target word using the L-BFGS (Byrd et al., 1995) algorithm. For regularization, Gaussian priors are utilized.

DWL model probabilities are computed as

$$p(\mathbf{e}|\mathbf{f}) = \prod_{e \in \mathbf{V_E}} p(e^-|\mathbf{f}) \cdot \prod_{e \in \mathbf{e}} \frac{p(e^+|\mathbf{f})}{p(e^-|\mathbf{f})} \tag{4}$$

with $\mathbf{V_E}$ being the target vocabulary, $\mathbf{e}$ the set of target words in a sentence, and $\mathbf{f}$ the set of source words, respectively. Here, the event $e^+$ is used when the target word $e$ is included in the target sentence and $e^-$ if not. As the left part of the product in Equation (4) is constant given a source sentence, it can be dropped, which enables us to score partial hypotheses during search.

### 3.5.2 Triplet Lexicon

The second lexicon model we employ in Jane, the *triplet lexicon model*, is in many aspects related to IBM model 1 (Brown et al., 1993), but extends it with an additional word in the conditioning part of the lexical probabilities. This introduces a means for an improved representation of long-range dependencies in the data. Like IBM model 1, the triplets are trained iteratively with the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). Jane implements the so-called inverse triplet model $p(e|f, f')$.

The triplet lexicon model score $t(\cdot)$ of the application of a rule $X \rightarrow \langle \alpha, \beta \rangle$ where $(\alpha, \beta)$ is a bilingual phrase pair that may contain symbols from the non-terminal set is computed as

$$t(\alpha, \beta, f_0^J) = \tag{5}$$

$$-\sum_e \log \left( \frac{2}{J \cdot (J+1)} \sum_j \sum_{j'>j} p(e|f_j, f_{j'}) \right)$$

with $e$ ranging over all terminal symbols in the target part $\beta$ of the rule. The second sum selects all words from the source sentence $f_0^J$ (including the empty word that is denoted as $f_0$ here). The third sum incorporates the rest of the source sentence right of the first triggering word. The order of the triggers is not relevant because per definition $p(e|f, f') = p(e|f', f)$, i.e. the model is symmetric. Non-terminals in $\beta$ have to be skipped when the rule is scored.

In Jane, we also implemented scoring for a variant of the triplet lexicon model called the *path-constrained* (or *path-aligned*) triplet model. The characteristic of path-constrained triplets is that the first trigger $f$ is restricted to the aligned target word $e$. The second trigger $f'$ is allowed to move along the whole remaining source sentence. For the training of the model, we use word alignment information obtained by GIZA++ (Och and Ney, 2003). To be able to apply the model in search, Jane has to be run with a phrase table that contains word alignment for each phrase, too, with the exception of phrases which are composed purely of non-terminals. Jane's phrase extraction can optionally supply this information from the training data.

(Hasan et al., 2008) and (Hasan and Ney, 2009) employ similar techniques and provide some more discussion on the path-aligned variant of the model and other possible restrictions.

### 3.6 Forced Alignments

Jane has also preliminary support for forced alignments between a given source and target sentence. Given a sentence in the source language and its translation in the target language, we find the best way the source sentence can be translated into the given target sentence, using the available inventory of phrases. This is needed for more advanced training approaches like the ones presented in (Blunsom et al., 2008) or (Cmejrek et al., 2009). As reported in these papers, due to the restrictions in the phrase extraction process, not all sentences in the training corpus can be aligned in this way.

### 3.7 Optimization Methods

Two method based on $n$-best for minimum error rate training (MERT) of the parameters of the log-linear model are included in Jane. The first one is the procedure described in (Och, 2003), which has become a standard in the machine translation community. We use an in-house implementation of the method.

The second one is the MIRA algorithm, first applied for machine translation in (Chiang et al., 2009). This algorithm is more adequate when the number of parameters to optimize is large.

If the Numerical Recipes library (Press et al., 2002) is available, an additional general purpose optimization tool is also compiled. Using this tool a single-best optimization procedure based on the downhill simplex method (Nelder and Mead, 1965) is included. This method, however, can be considered deprecated in favour of the above mentioned methods.

### 3.8 Parallelized operation

If the Sun Grid Engine[2] is available, all operations of Jane can be parallelized. For the extraction process, the corpus is split into chunks (the granularity being user-controlled) which are distributed in the computer cluster. Count collection, marginal computation and count normalization all happens in an automatic and parallel manner.

For the translation process a batch job is started on a number of computers. A server distributes the sentences to translate to the computers that have been made available to the translation job.

The optimization process also benefits from the parallelized optimization. Additionally, for the minimum error rate training methods, random restarts may be performed on different computers in a parallel fashion.

The same client-server infrastructure used for parallel translation may also be reused for interactive systems. Although no code in this direction is provided, one would only need to implement a corresponding frontend which communicates with the translation server (which may be located on another machine).

### 3.9 Extensibility

One of the goals when implementing the toolkit was to make it easy to extend it with new features. For this, an abstract class was created which we called *secondary model*. New models need only to derive from this class and implement the abstract methods for data reading and costs computation. This allows for an encapsulation of the computations, which can be activated and deactivated on demand. The models described in Sections 3.3

[2] http://www.sun.com/software/sge/

through 3.5 are implemented in this way. We thus try to achieve loose coupling in the implementation.

In addition a flexible prefix tree implementation with on-demand loading capabilities is included as part of the code. This class has been used for implementing on-demand loading of phrases in the spirit of (Zens and Ney, 2007) and the on-demand $n$-gram format described in Section 3.2, in addition to some intermediate steps in the phrase extraction process. The code may also be reused in other, independent projects.

### 3.10 Code

The main core of Jane has been implemented in C++. Our guideline was to write code that was correct, maintainable and efficient. We tried to achieve correctness by means of unit tests integrated in the source as well as regression tests. We also defined a set of coding guidelines, which we try to enforce in order to have readable and maintainable code. Examples include using descriptive variable names, appending an underscore to private members of classes or having each class name start with an uppercase letter while variable names start with lowercase letters.

The code is documented at great length using the doxygen system,[3] and the filling up of the missing parts is an ongoing effort. Every tool comes with an extensive help functionality, and the main tools also have their own man pages.

As for efficiency we always try to speed up the code and reduce memory consumption by implementing better algorithms. We try to avoid "dark magic programming methods" and hard to follow optimizations are only applied in critical parts of the code. We try to document every such occurrence.

## 4 Experimental Results

In this section we will present some experimental results obtained using Jane. We will pay special attention to the performance of the new reordering and lexicon models presented in this paper. We will present results on three different large-scale tasks and language pairs.

Additionally RWTH participated in this year's WMT evaluation, where Jane was one of the submitted systems. We refer to the system description for supplementary experimental results.

[3] `http://www.doxygen.org`

| | dev | | test | |
|---|---|---|---|---|
| System | BLEU | TER | BLEU | TER |
| Jane baseline | 24.2 | 59.5 | 25.4 | 57.4 |
| + reordering | 25.2 | 58.2 | 26.5 | 56.1 |

Table 1: Results for Europarl German-English data. BLEU and TER results are in percentage.

### 4.1 Europarl Data

The first task is the Europarl as defined in the Quaero project. The main part of the corpus in this task consists of the Europarl corpus as used in the WMT evaluation (Callison-Burch et al., 2009), with some additional data collected in the scope of the project.

We tried the reordering approach presented in Section 3.4 on the German-English language pair. The results are shown in Table 1. As can be seen from these results, the additional reorderings obtain nearly 1% improvement both in BLEU and TER scores. Regrettably for this corpus the extended lexicon models did not bring any improvements.

Table 2 shows the results for the French-English language pair of the Europarl task. On this task the extended lexicon models yield an improvement over the baseline system of 0.9% in BLEU and 0.9% in TER on the test set.

### 4.2 NIST Arabic-English

We also show results on the Arabic-English NIST'08 task, using the NIST'06 set as development set. It has been reported in other work that the hierarchical system is not competitive with a phrase-based system for this language pair (Birch et al., 2009). We report the figures of our state-of-the-art phrase-based system as comparison (denoted as PBT).

As can be seen from Table 3, the baseline Jane system is in fact 0.6% worse in BLEU and 1.0% worse in TER than the baseline PBT system. When we include the extended lexicon models we see that the difference in performance is reduced. For Jane the extended lexicon models give an improvement of up to 1.9% in BLEU and 1.7% in TER, respectively, bringing the system on par with the PBT system extended with the same lexicon models, and obtaining an even slightly better BLEU score.

| | dev | | test | |
|---|---|---|---|---|
| | BLEU | TER | BLEU | TER |
| Baseline | 30.0 | 52.6 | 31.1 | 50.0 |
| DWL | 30.4 | 52.2 | 31.4 | 49.6 |
| Triplets | 30.4 | 52.0 | 31.7 | 49.4 |
| path-constrained Triplets | 30.3 | 52.1 | 31.6 | 49.3 |
| DWL + Triplets | 30.7 | 52.0 | 32.0 | 49.1 |
| DWL + path-constrained Triplets | 30.8 | 51.7 | 31.6 | 49.3 |

Table 2: Results for the French-English task. BLEU and TER results are in percentage.

| | dev (MT'06) | | | | test (MT'08) | | | |
|---|---|---|---|---|---|---|---|---|
| | Jane | | PBT | | Jane | | PBT | |
| | BLEU | TER | BLEU | TER | BLEU | TER | BLEU | TER |
| Baseline | 43.2 | 50.8 | 44.1 | 49.4 | 44.1 | 50.1 | 44.7 | 49.1 |
| DWL | 45.3 | 48.7 | 45.1 | 48.4 | 45.6 | 48.4 | 45.6 | 48.4 |
| Triplets | 44.4 | 49.1 | 44.6 | 49.2 | 45.3 | 48.8 | 44.9 | 49.0 |
| path-constrained Triplets | 44.3 | 49.4 | 44.7 | 49.1 | 44.9 | 49.3 | 45.3 | 48.7 |
| DWL + Triplets | 45.0 | 48.9 | 45.1 | 48.5 | 45.3 | 48.6 | 45.5 | 48.5 |
| DWL + path-constrained Triplets | 45.2 | 48.8 | 45.1 | 48.6 | 46.0 | 48.5 | 45.8 | 48.3 |

Table 3: Results for the Arabic-English task. BLEU and TER results are in percentage.

## 5 Discussion

We feel that the hierarchical phrase-based translation approach still shares some shortcomings concerning lexical selection with conventional phrase-based translation. Bilingual lexical context beyond the phrase boundaries is barely taken into account by the base model. In particular, if only one generic non-terminal is used, the selection of a sub-phrase that fills the gap of a hierarchical phrase is not affected by the words composing the phrase it is embedded in – except for the language model score. This shortcoming is one of the issues syntactically motivated models try to address.

The extended lexicon models analyzed in this work also try to address this issue. One can consider that they complement the efforts that are being made on a deep structural level within the hierarchical approach. Though they are trained on surface forms only, without any syntactic information, they still operate at a scope that exceeds the capability of common feature sets of standard hierarchical phrase-based SMT systems.

As the experiments in Section 4 show, the effect of these extended lexicon models is more important for the hierarchical phrase-based approach than for the phrase-based approach. In our opinion this is probably mainly due to the higher flexibility of the hierarchical system, both because of its intrinsic nature and because of the higher number of phrases extracted by the system. The scoring of the phrases is still carried out by simple relative frequencies, which seem to be insufficient. The additional lexicon models seem to help in this respect.

### 5.1 Short Comparison with Joshua

As mentioned in Section 2, Joshua is the most similar decoder to our own. It was developed in parallel at the Johns Hopkins University and it is

| System | words/sec |
|---|---|
| Joshua | 11.6 |
| Jane cube prune | 15.9 |
| Jane cube grow | 60.3 |

Table 4: Speed comparison Jane vs. Joshua. We measure the translated words per second.

currently used by a number of groups around the world.

Jane was started separately and independently. In their basic working mode, both systems implement parsing using a synchronous grammar and include language model information. Each of the projects then progressed independently, most of the features described in Section 3 being only available in Jane.

Efficiency is one of the points where we think Jane outperforms Joshua. One of the reasons can well be the fact that it is written in C++ while Joshua is written in Java. In order to compare running times we converted a grammar extracted by Jane to Joshua's format and adapted the parameters accordingly. To the best of our knowledge we configured both decoders to perform the same task (cube pruning, 300-best generation, same pruning parameters). Except for some minor differences[4] the results were equal.

We tried this setup on the IWSLT'08 Arabic to English translation task. The speed results (measured in translated words per second) can be seen in Table 4. Jane operating with cube prune is nearly 50% faster than Joshua, at the same level of translation performance. If we switch to cube grow, the speed difference is even bigger, with a speedup of nearly 4 times. However this usually comes with a penalty in BLEU score (normally under 0.5% BLEU in our experience). This increased speed can be specially interesting for applications like interactive machine translation or online translation services, where the response time is critical and sometimes even more important than a small (and often hardly noticeable) loss in translation quality.

Another important point concerning efficiency is the startup time. Thanks to the binary format described in Section 3.9, there is virtually no delay in the loading of the phrase table in Jane.[5] In fact Joshua's long phrase table loading times were the main reason the performance measures were done on a small corpus like IWSLT instead of one of the large tasks described in Section 4.

We want to make clear that we did not go into great depth in the workings of Joshua, just stayed at the basic level described in the manual. This tool is used also for large-scale evaluations and hence there certainly are settings for dealing with these big tasks. Therefore this comparison has to be taken with a grain of salt.

We also want to stress that we explicitly chose to leave translation results out of this comparison. Several different components have great impact on translation quality, including phrase extraction, minimum error training and additional parameter settings of the decoder. As we pointed out we do not have the expertise in Joshua to perform all these tasks in an optimal way, and for that reason we did not include such a comparison. However, both JHU and RWTH participated in this year's WMT evaluation, where the systems, applied by their respective authors, can be directly compared.

And in no way do we see Joshua and Jane as "competing" systems. Having different systems is always enriching, and particularly as system combination shows great improvements in translation quality, having several alternative systems can only be considered a positive situation.

## 6 Licensing

Jane is distributed under a custom open source license. This includes free usage for noncommercial purposes as long as any changes made to the original software are published under the terms of the same license. The exact formulation is available at the download page for Jane.

## 7 Conclusion

With Jane, we release a state-of-the-art hierarchical toolkit to the scientific community and hope to provide a good starting point for fellow researchers, allowing them to have a solid system even if the research field is new to them. It is available for download from `http://www.hltpr.rwth-aachen.de/jane`. The system in its current state is stable and efficient enough to handle even large-scale tasks such as

[4]E.g. the OOVs seem to be handled in a slightly different way, as the placement was sometimes different.

[5]There is, however, still some delay when loading the language model for some of the supported formats.

the WMT and NIST evaluations, while producing highly competitive results.

Moreover, we presented additional reordering and lexicon models that further enhance the performance of the system.

And in case you are wondering, Jane is **J**ust an **A**cronym, **N**othing **E**lse. The name comes from the character in the Ender's Game series (Card, 1986).

## Acknowledgments

Special thanks to the people who have contributed code to Jane: Markus Freitag, Stephan Peitz, Carmen Heger, Arne Mauser and Niklas Hoppe.

This work was partly realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation, and also partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DARPA.

## References

Alexandra Birch, Phil Blunsom, and Miles Osborne. 2009. A Quantitative Analysis of Reordering Phenomena. In *Proc. of the Workshop on Statistical Machine Translation*, pages 197–205, Athens, Greece, March.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A Discriminative Latent Variable Model for Statistical Machine Translation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 200–208, Columbus, Ohio, June.

Peter F. Brown, Stephan A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, June.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proc. of the Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece, March.

Orson Scott Card. 1986. *Speaker for the Dead*. Tor Books.

Jean-Cédric Chappelier and Martin Rajman. 1998. A Generalized CYK Algorithm for Parsing Stochastic CFG. In *Proc. of the First Workshop on Tabulation in Parsing and Deduction*, pages 133–137, Paris, France, April.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new Features for Statistical Machine Translation. In *Proc. of the Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 218–226, Boulder, Colorado, June.

David Chiang. 2007. Hierarchical Phrase-based Translation. *Computational Linguistics*, 33(2):201–228, June.

Martin Cmejrek, Bowen Zhou, and Bing Xiang. 2009. Enriching SCFG Rules Directly From Efficient Bilingual Chart Parsing. In *Proc. of the International Workshop on Spoken Language Translation (IWSLT)*, pages 136–143, Tokyo, Japan.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–22.

Saša Hasan and Hermann Ney. 2009. Comparison of Extended Lexicon Models in Search and Rescoring for SMT. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, volume short papers, pages 17–20, Boulder, CO, USA, June.

Saša Hasan, Juri Ganitkevitch, Hermann Ney, and Jesús Andrés-Ferrer. 2008. Triplet Lexicon Models for Statistical Machine Translation. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 372–381.

Liang Huang and David Chiang. 2007. Forest Rescoring: Faster Decoding with Integrated Language Models. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 144–151, Prague, Czech Republic, June.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 177–180, Prague, Czech Republic, June.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009. Joshua: An Open Source Toolkit for Parsing-Based Machine Translation. In *Proc. of the Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, March.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 210–218, Singapore, August.

John A. Nelder and Roger Mead. 1965. The Downhill Simplex Method. *Computer Journal*, 7:308.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och. 2003. Minimum Error Rate Training for Statistical Machine Translation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2002. *Numerical Recipes in C++*. Cambridge University Press, Cambridge, UK.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 577–585, Columbus, Ohio, June.

Andreas Stolcke. 2002. SRILM – an Extensible Language Modeling Toolkit. In *Proc. of the International Conference on Spoken Language Processing (ICSLP)*, volume 3, pages 901–904, Denver, Colorado, September.

David Talbot and Miles Osborne. 2007. Smoothed Bloom Filter Language Models: Tera-scale LMs on the Cheap. In *Proc. of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 468–476, Prague, Czech Republic, June.

Ashish Venugopal, Andreas Zollmann, N.A. Smith, and Stephan Vogel. 2009. Preference Grammars: Softening Syntactic Constraints to Improve Statistical Machine Translation. In *Proc. of the Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 236–244, Boulder, Colorado, June.

David Vilar and Hermann Ney. 2009. On LM Heuristics for the Cube Growing Algorithm. In *Proc. of the Annual Conference of the European Association for Machine Translation (EAMT)*, pages 242–249, Barcelona, Spain, May.

David Vilar, Daniel Stein, and Hermann Ney. 2008. Analysing Soft Syntax Features and Heuristics for Hierarchical Phrase Based Machine Translation. In *Proc. of the International Workshop on Spoken Language Translation (IWSLT)*, pages 190–197, Waikiki, Hawaii, October.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Richard Zens and Hermann Ney. 2007. Efficient Phrase-Table Representation for Machine Translation with Applications to Online MT and Speech Translation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 492–499, Rochester, New York, April.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax Augmented Machine Translation via Chart Parsing. In *Proc. of the Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 138–141, New York, June.