Joshua: An Open Source Toolkit for Parsing-based Machine Translation

Zhifei Li, Chris Callison-Burch, Chris Dyer,[†] **Juri Ganitkevitch,**[+] **Sanjeev Khudanpur, Lane Schwartz,**[⋆] **Wren N. G. Thornton, Jonathan Weese** and **Omar F. Zaidan**
Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD
† Computational Linguistics and Information Processing Lab, University of Maryland, College Park, MD
+ Human Language Technology and Pattern Recognition Group, RWTH Aachen University, Germany
⋆ Natural Language Processing Lab, University of Minnesota, Minneapolis, MN

Abstract

We describe **Joshua**, an open source toolkit for statistical machine translation. Joshua implements all of the algorithms required for synchronous context free grammars (SCFGs): chart-parsing, n-gram language model integration, beam- and cube-pruning, and k-best extraction. The toolkit also implements suffix-array grammar extraction and minimum error rate training. It uses parallel and distributed computing techniques for scalability. We demonstrate that the toolkit achieves state of the art translation performance on the WMT09 French-English translation task.

1 Introduction

Large scale parsing-based statistical machine translation (e.g., Chiang (2007), Quirk et al. (2005), Galley et al. (2006), and Liu et al. (2006)) has made remarkable progress in the last few years. However, most of the systems mentioned above employ tailor-made, dedicated software that is not open source. This results in a high barrier to entry for other researchers, and makes experiments difficult to duplicate and compare. In this paper, we describe **Joshua**, a general-purpose open source toolkit for parsing-based machine translation, serving the same role as Moses (Koehn et al., 2007) does for regular phrase-based machine translation.

Our toolkit is written in Java and implements all the essential algorithms described in Chiang (2007): chart-parsing, n-gram language model integration, beam- and cube-pruning, and k-best extraction. The toolkit also implements suffix-array grammar extraction (Lopez, 2007) and minimum error rate training (Och, 2003). Additionally, parallel and distributed computing techniques are exploited to make it scalable (Li and Khudanpur, 2008b). We have also made great effort to ensure that our toolkit is easy to use and to extend.

The toolkit has been used to translate roughly a million sentences in a parallel corpus for large-scale discriminative training experiments (Li and Khudanpur, 2008a). We hope the release of the toolkit will greatly contribute the progress of the syntax-based machine translation research.[1]

2 Joshua Toolkit

When designing our toolkit, we applied general principles of software engineering to achieve three major goals: *Extensibility*, *end-to-end coherence*, and *scalability*.

Extensibility: The Joshua code is organized into separate *packages* for each major aspect of functionality. In this way it is clear which files contribute to a given functionality and researchers can focus on a single package without worrying about the rest of the system. Moreover, to minimize the problems of unintended interactions and unseen dependencies, which is common hinderance to extensibility in large projects, all extensible components are defined by Java *interfaces*. Where there is a clear point of departure for research, a basic implementation of each interface is provided as an *abstract class* to minimize the work necessary for new extensions.

End-to-end Cohesion: There are many components to a machine translation pipeline. One of the great difficulties with current MT pipelines is that these diverse components are often designed by separate groups and have different file format and interaction requirements. This leads to a large investment in scripts to convert formats and connect the different components, and often leads to untenable and non-portable projects as well as hinder-

[1] The toolkit can be downloaded at `http://www.sourceforge.net/projects/joshua`, and the instructions in using the toolkit are at `http://cs.jhu.edu/~ccb/joshua`.

ing repeatability of experiments. To combat these issues, the Joshua toolkit integrates most critical components of the machine translation pipeline. Moreover, each component can be treated as a stand-alone tool and does not rely on the rest of the toolkit we provide.

Scalability: Our third design goal was to ensure that the decoder is scalable to large models and data sets. The parsing and pruning algorithms are carefully implemented with dynamic programming strategies, and efficient data structures are used to minimize overhead. Other techniques contributing to scalability includes suffix-array grammar extraction, parallel and distributed decoding, and bloom filter language models.

Below we give a short description about the main functions implemented in our Joshua toolkit.

2.1 Training Corpus Sub-sampling

Rather than inducing a grammar from the full parallel training data, we made use of a method proposed by Kishore Papineni (personal communication) to select the subset of the training data consisting of sentences useful for inducing a grammar to translate a particular test set. This method works as follows: for the development and test sets that will be translated, every n-gram (up to length 10) is gathered into a map $\mathcal{W}$ and associated with an initial count of zero. Proceeding in order through the training data, for each sentence pair whose source-to-target length ratio is within one standard deviation of the average, if any n-gram found in the *source sentence* is also found in $\mathcal{W}$ with a count of less than k, the sentence is selected. When a sentence is selected, the count of every n-gram in $\mathcal{W}$ that is found in the source sentence is incremented by the number of its occurrences in the source sentence. For our submission, we used $k = 20$, which resulted in 1.5 million (out of 23 million) sentence pairs being selected for use as training data. There were 30,037,600 English words and 30,083,927 French words in the subsampled training corpus.

2.2 Suffix-array Grammar Extraction

Hierarchical phrase-based translation requires a translation grammar extracted from a parallel corpus, where grammar rules include associated feature values. In real translation tasks, the grammars extracted from large training corpora are often far too large to fit into available memory.

In such tasks, feature calculation is also very expensive in terms of time required; huge sets of extracted rules must be sorted in two directions for relative frequency calculation of such features as the translation probability $p(f|e)$ and reverse translation probability $p(e|f)$ (Koehn et al., 2003). Since the extraction steps must be re-run if any change is made to the input training data, the time required can be a major hindrance to researchers, especially those investigating the effects of tokenization or word segmentation.

To alleviate these issues, we extract only a subset of all available rules. Specifically, we follow Callison-Burch et al. (2005; Lopez (2007) and use a source language suffix array to extract only those rules which will actually be used in translating a particular set of test sentences. This results in a vastly smaller rule set than techniques which extract all rules from the training set.

The current code requires suffix array rule extraction to be run as a pre-processing step to extract the rules needed to translate a particular test set. However, we are currently extending the decoder to directly access the suffix array. This will allow the decoder at runtime to efficiently extract exactly those rules needed to translate a particular sentence, without the need for a rule extraction pre-processing step.

2.3 Decoding Algorithms[2]

Grammar formalism: Our decoder assumes a probabilistic synchronous context-free grammar (SCFG). Currently, it only handles SCFGs of the kind extracted by Heiro (Chiang, 2007), but is easily extensible to more general SCFGs (e.g., (Galley et al., 2006)) and closely related formalisms like synchronous tree substitution grammars (Eisner, 2003).

Chart parsing: Given a source sentence to decode, the decoder generates a one-best or k-best translations using a CKY algorithm. Specifically, the decoding algorithm maintains a *chart*, which contains an array of *cells*. Each cell in turn maintains a list of proven *items*. The parsing process starts with the axioms, and proceeds by applying the inference rules repeatedly to prove new items until proving a goal item. Whenever the parser proves a new item, it adds the item to the appropriate chart cell. The item also maintains back-

[2]More details on the decoding algorithms are provided in (Li et al., 2009a).

pointers to antecedent items, which are used for k-best extraction.

Pruning: Severe pruning is needed in order to make the decoding computationally feasible for SCFGs with large target-language vocabularies. In our decoder, we incorporate two pruning techniques: beam and cube pruning (Chiang, 2007).

Hypergraphs and k-best extraction: For each source-language sentence, the chart-parsing algorithm produces a *hypergraph*, which represents an exponential set of likely derivation hypotheses. Using the k-best extraction algorithm (Huang and Chiang, 2005), we extract the k most likely derivations from the hypergraph.

Parallel and distributed decoding: We also implement *parallel decoding* and a *distributed language model* by exploiting multi-core and multi-processor architectures and distributed computing techniques. More details on these two features are provided by Li and Khudanpur (2008b).

2.4 Language Models

In addition to the distributed LM mentioned above, we implement three local n-gram language models. Specifically, we first provide a straightforward implementation of the n-gram scoring function in Java. This Java implementation is able to read the standard ARPA backoff n-gram models, and thus the decoder can be used independently from the SRILM toolkit.[3] We also provide a native code bridge that allows the decoder to use the SRILM toolkit to read and score n-grams. This native implementation is more scalable than the basic Java LM implementation. We have also implemented a Bloom Filter LM in Joshua, following Talbot and Osborne (2007).

2.5 Minimum Error Rate Training

Johsua's MERT module optimizes parameter weights so as to maximize performance on a development set as measuered by an automatic evaluation metric, such as Bleu. The optimization consists of a series of line-optimizations along the dimensions corresponding to the parameters. The search across a dimension uses the efficient method of Och (2003). Each iteration of our MERT implementation consists of multiple weight updates, each reflecting a greedy selection of the dimension giving the most gain. Each iteration also optimizes several random "intermediate initial" points in addition to the one surviving from the previous iteration, as an approximation to performing multiple random restarts. More details on the MERT method and the implementation can be found in Zaidan (2009).[4]

3 WMT-09 Translation Task Results

3.1 Training and Development Data

We assembled a very large French-English training corpus (Callison-Burch, 2009) by conducting a web crawl that targted bilingual web sites from the Canadian government, the European Union, and various international organizations like the Amnesty International and the Olympic Committee. The crawl gathered approximately 40 million files, consisting of over 1TB of data. We converted pdf, doc, html, asp, php, etc. files into text, and preserved the directory structure of the web crawl. We wrote set of simple heuristics to transform French URLs onto English URLs, and considered matching documents to be translations of each other. This yielded 2 million French documents paired with their English equivalents. We split the sentences and paragraphs in these documents, performed sentence-aligned them using software that IBM Model 1 probabilities into account (Moore, 2002). We filtered and de-duplcated the resulting parallel corpus. After discarding 630 thousand sentence pairs which had more than 100 words, our final corpus had 21.9 million sentence pairs with 587,867,024 English words and 714,137,609 French words.

We distributed the corpus to the other WMT09 participants to use in addition to the Europarl v4 French-English parallel corpus (Koehn, 2005), which consists of approximately 1.4 million sentence pairs with 39 million English words and 44 million French words. Our translation model was trained on these corpora using the subsampling descried in Section 2.1.

For language model training, we used the monolingual news and blog data that was assembled by the University of Edinburgh and distributed as part of WMT09. This data consisted

[3] This feature allows users to easily try the Joshua toolkit without installing the SRILM toolkit and compiling the native bridge code. However, users should note that the basic Java LM implementation is not as scalable as the native bridge code.

[4] The module is also available as a standalone application, *Z-MERT*, that can be used with other MT systems. (Software and documentation at: `http://cs.jhu.edu/~ozaidan/zmert`.)

of 21.2 million English sentences with half a billion words. We used SRILM to train a 5-gram language model using a vocabulary containing the 500,000 most frequent words in this corpus. Note that we did not use the English side of the parallel corpus as language model training data.

To tune the system parameters we used News Test Set from WMT08 (Callison-Burch et al., 2008), which consists of 2,051 sentence pairs with 43 thousand English words and 46 thousand French words. This is in-domain data that was gathered from the same news sources as the WMT09 test set.

3.2 Translation Scores

The translation scores for four different systems are reported in Table 1.[5]

Baseline: In this system, we use the GIZA++ toolkit (Och and Ney, 2003), a suffix-array architecture (Lopez, 2007), the SRILM toolkit (Stolcke, 2002), and minimum error rate training (Och, 2003) to obtain word-alignments, a translation model, language models, and the optimal weights for combining these models, respectively.

Minimum Bayes Risk Rescoring: In this system, we re-ranked the n-best output of our baseline system using Minimum Bayes Risk (Kumar and Byrne, 2004). We re-score the top 300 translations to minimize expected loss under the Bleu metric.

Deterministic Annealing: In this system, instead of using the regular MERT (Och, 2003) whose training objective is to minimize the one-best error, we use the deterministic annealing training procedure described in Smith and Eisner (2006), whose objective is to minimize the *expected* error (together with the entropy regularization technique).

Variational Decoding: Statistical models in machine translation exhibit *spurious ambiguity*. That is, the probability of an output string is split among many distinct derivations (e.g., trees or segmentations). In principle, the goodness of a string is measured by the total probability of its many derivations. However, finding the best string (e.g., during decoding) is then computationally intractable. Therefore, most systems use a simple Viterbi approximation that measures the goodness of a string using only its most probable derivation. Instead, we develop a variational approximation, which considers all the derivations but still allows tractable decoding. More details will be provided in Li et al. (2009b). In this system, we have used both deterministic annealing (for training) and variational decoding (for decoding).

[5]Note that the implementation of the novel techniques used to produce the non-baseline results is not part of the current Joshua release, though we plan to incorporate it in the next release.

System	BLEU-4
Joshua Baseline	25.92
Minimum Bayes Risk Rescoring	26.16
Deterministic Annealing	25.98
Variational Decoding	**26.52**

Table 1: The uncased BLEU scores on WMT-09 French-English Task. The test set consists of 2525 segments, each with one reference translation.

4 Conclusions

We have described a scalable toolkit for parsing-based machine translation. It is written in Java and implements all the essential algorithms described in Chiang (2007) and Li and Khudanpur (2008b): chart-parsing, n-gram language model integration, beam- and cube-pruning, and k-best extraction. The toolkit also implements suffix-array grammar extraction (Callison-Burch et al., 2005; Lopez, 2007) and minimum error rate training (Och, 2003). Additionally, parallel and distributed computing techniques are exploited to make it scalable. The decoder achieves state of the art translation performance.

Acknowledgments

This research was supported in part by the Defense Advanced Research Projects Agency's GALE program under Contract No. HR0011-06-2-0001 and the National Science Foundation under grants No. 0713448 and 0840112. The views and findings are the authors' alone.

References

Chris Callison-Burch, Colin Bannard, and Josh Schroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of ACL*.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation (WMT08)*.

Chris Callison-Burch. 2009. A 10^9 word parallel corpus. In preparation.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of ACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the ACL/Coling*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the International Workshop on Parsing Technologies*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT/NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the ACL-2007 Demo and Poster Sessions*.

Philipp Koehn. 2005. A parallel corpus for statistical machine translation. In *Proceedings of MT-Summit*, Phuket, Thailand.

Shankar Kumar and William Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *Proceedings of HLT/NAACL*.

Zhifei Li and Sanjeev Khudanpur. 2008a. Large-scale discriminative n-gram language models for statistical machine translation. In *Proceedings of AMTA*.

Zhifei Li and Sanjeev Khudanpur. 2008b. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings Workshop on Syntax and Structure in Statistical Translation*.

Zhifei Li, Chris Callison-Burch, Sanjeev Khudanpur, and Wren Thornton. 2009a. Decoding in joshua: Open source, parsing-based machine translation. *The Prague Bulletin of Mathematical Linguistics*, 91:47–56.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009b. Variational decoding for statistical machine translation. In preparation.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment templates for statistical machine translation. In *Proceedings of the ACL/Coling*.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of EMNLP-CoLing*.

Robert C. Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Proceedings of AMTA*.

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 *Proceedings of ACL*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of ACL*.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of the ACL/Coling*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, Denver, Colorado, September.

David Talbot and Miles Osborne. 2007. Randomised language modelling for statistical machine translation. In *Proceedings of ACL*.

Omar F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.