Demonstration of 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 translation via 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 also provide a demonstration outline for illustrating the toolkit's features to potential users, whether they be newcomers to the field or power users interested in extending the toolkit.

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 Java-based 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.

*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.

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 a common hindrance 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 formats 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 hindering 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 con-

System	BLEU-4
google	31.14
lium	26.89
dcu	26.86
joshua	**26.52**
uka	25.96
limsi	25.51
uedin	25.44
rwth	24.89
cmu-statxfer	23.65

Table 1: Uncased BLEU scores for the top primary systems on the WMT-09 French-English Task.

tributing to scalability includes suffix-array grammar extraction, parallel and distributed decoding, and bloom filter language models.

The Joshua system offers state-of-the-art translation quality, having been ranked 4th out of 16 systems in the French-English task of the 2009 WMT evaluation, both in automated (see Table 1) and human evaluation (Callison-Burch et al., 2009).

2.1 Joshua Toolkit Features

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

- **Training Corpus Sub-sampling:** Rather than inducing a grammar from the full parallel training data, we select a subset of the training data consisting of sentences useful for inducing a grammar to translate a particular test set. To accomplish this, we make use of the method proposed by Kishore Papineni (personal communication), outlined in further detail in (Li et al., 2009a). For the 2009 WMT evaluation, the method achieved a 90% reduction in training corpus size while maintaining state-of-the-art performance.

- **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. Similarly, the grammars' size makes feature extraction time-consuming, which 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) and 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.

  Joshua incorporates direct access to the suffix array into the decoder, allowing for rule extraction to be performed for each input sentence individually. Running rule extraction as a stand-alone pre-processing step is also supported.

- **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 (as in 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 the CKY algorithm. In addition to standard CKY, the system also maintains backpointers, 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), it is possible to extract the k most likely derivations from the hypergraph.

- **Parallel and distributed decoding:** We also implement *parallel decoding* and a *dis-*

tributed 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 (2008).

- **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.[1] 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).

- **Minimum Error Rate Training:** Joshua's MERT module optimizes parameter weights so as to maximize performance on a development set as measured 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 MERT iteration 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).[2]

- **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).

[1] 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.

[2] 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`.)

3 Demonstration Outline

The purpose of the demonstration is 4-fold: 1) to give newcomers to the field of statistical machine translation an idea of the state-of-the-art; 2) to show actual, live, end-to-end operation of the system, highlighting its main components, targeting potential users; 3) to illustrate, through visual aids, the underlying algorithms, for those interested in the technical details; and 4) to explain how those components can be extended, for potential *power users* who want to be familiar with the code itself.

The first component of the demonstration will be an interactive user interface, where arbitrary user input in a source language is entered into a web form and then translated into a target language by the system. This component specifically targets newcomers to SMT, and demonstrates the current state of the art in the field. We will have trained multiple systems (for multiple language pairs), hosted on a remote server, which will be queried with the sample source sentences.

Potential users of the system would be interested in seeing an actual operation of the system, in a similar fashion to what they would observe on their own machines when using the toolkit. For this purpose, we will demonstrate three main modules of the toolkit: the rule extraction module, the MERT module, and the decoding module. Each module will have a separate terminal window executing it, hence demonstrating both the module's expected output as well as its speed of operation.

In addition to demonstrating the functionality of each module, we will also provide accompanying visual aids that illustrate the underlying algorithms and the technical operational details. We will provide visualization of the search graph and

the 1-best derivation, which would illustrate the functionality of the decoder, as well as alternative translations for phrases of the source sentence, and where they were learned in the parallel corpus, illustrating the functionality of the grammar rule extraction. For the MERT module, we will provide figures that illustrate Och's efficient line search method.

Finally, power users of the system would be interested in extending the toolkit to accommodate new functionality. To that end, we will specifically illustrate several potential extensions to the toolkit by highlighting portions of the code that the user would be expected to modify and/or augment when implementing such extensions. For instance, we will point out how a new feature function can be added to the decoder, or how a new metric can be incorporated to the MERT module.

4 Demonstration Requirements

The different components of the demonstration will be spread across at most 3 machines: one for the live "instant translation" user interface, one for demonstrating the different components of the system and algorithmic visualizations, and one designated for technical discussion of the code. We will provide the machines ourselves and ensure the proper software is installed and configured. However, we are requesting that large LCD monitors be made available, if possible, since that would allow more space to demonstrate the different components with clarity than our laptop displays would provide. We will also require Internet connectivity for the live demonstration, in order to gain access to remote servers where trained models will be hosted.

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, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece, March. Association for Computational Linguistics.

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, 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*.

Zhifei Li and Sanjeev Khudanpur. 2008. 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, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009a. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, March. Association for Computational Linguistics.

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*.

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 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.