

# Joshua 2.0: A Toolkit for Parsing-Based Machine Translation with Syntax, Semirings, Discriminative Training and Other Goodies

Zhifei Li, Chris Callison-Burch, Chris Dyer,<sup>†</sup> Juri Ganitkevitch, Ann Irvine, Sanjeev Khudanpur, Lane Schwartz,<sup>\*</sup> Wren N. G. Thornton, Ziyuan Wang, Jonathan Weese and Omar F. Zaidan

Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD

<sup>†</sup> Computational Linguistics and Information Processing Lab, University of Maryland, College Park, MD

<sup>\*</sup> Natural Language Processing Lab, University of Minnesota, Minneapolis, MN

## Abstract

We describe the progress we have made in the past year on **Joshua** (Li et al., 2009a), an open source toolkit for parsing based machine translation. The new functionality includes: support for translation grammars with a rich set of syntactic nonterminals, the ability for external modules to posit constraints on how spans in the input sentence should be translated, lattice parsing for dealing with input uncertainty, a semiring framework that provides a unified way of doing various dynamic programming calculations, variational decoding for approximating the intractable MAP decoding, hypergraph-based discriminative training for better feature engineering, a parallelized MERT module, document-level and tail-based MERT, visualization of the derivation trees, and a cleaner pipeline for MT experiments.

## 1 Introduction

**Joshua** is an open-source toolkit for parsing-based machine translation that is written in Java. The initial release of **Joshua** (Li et al., 2009a) was a re-implementation of the Hiero system (Chiang, 2007) and all its associated algorithms, including: chart parsing,  $n$ -gram language model integration, beam and cube pruning, and  $k$ -best extraction. The **Joshua** 1.0 release also included re-implementations of suffix array grammar extraction (Lopez, 2007; Schwartz and Callison-Burch, 2010) and minimum error rate training (Och, 2003; Zaidan, 2009). Additionally, it included parallel and distributed computing techniques for scalability (Li and Khudanpur, 2008).

This paper describes the additions to the toolkit over the past year, which together form the 2.0 release. The software has been heavily used by the

authors and several other groups in their daily research, and has been substantially refined since the first release. The most important new functions in the toolkit are:

- Support for any style of synchronous context free grammar (SCFG) including syntax augmentation machine translation (SAMT) grammars (Zollmann and Venugopal, 2006)
- Support for external modules to posit translations for spans in the input sentence that constrain decoding (Irvine et al., 2010)
- Lattice parsing for dealing with input uncertainty, including ambiguous output from speech recognizers or Chinese word segmenters (Dyer et al., 2008)
- A semiring architecture over hypergraphs that allows many inference operations to be implemented easily and elegantly (Li and Eisner, 2009)
- Improvements to decoding through variational decoding and other approximate methods that overcome intractable MAP decoding (Li et al., 2009b)
- Hypergraph-based discriminative training for better feature engineering (Li and Khudanpur, 2009b)
- A parallelization of MERT's computations, and supporting document-level and tail-based optimization (Zaidan, 2010)
- Visualization of the derivation trees and hypergraphs (Weese and Callison-Burch, 2010)
- A convenient framework for designing and running reproducible machine translation experiments (Schwartz, under review)

The sections below give short descriptions for each of these new functions.

## 2 Support for Syntax-based Translation

The initial release of **Joshua** supported only Hiero-style SCFGs, which use a single nonterminal symbol  $X$ . This release includes support for arbitrary SCFGs, including ones that use a rich set of linguistic nonterminal symbols. In particular we have added support for Zollmann and Venugopal (2006)’s syntax-augmented machine translation. SAMT grammar extraction is identical to Hiero grammar extraction, except that one side of the parallel is parsed, and syntactic labels replace the  $X$  nonterminals in Hiero-style rules. Instead of extracting this Hiero rule from the bitext

$$[X] \Rightarrow [X, 1] \text{ sans } [X, 2] \mid [X, 1] \text{ without } [X, 2]$$

the nonterminals can be labeled according to which constituents cover the nonterminal span on the parsed side of the bitext. This constrains what types of phrases the decoder can use when producing a translation.

$$[VP] \Rightarrow [VBN] \text{ sans } [NP] \mid [VBN] \text{ without } [NP]$$
$$[NP] \Rightarrow [NP] \text{ sans } [NP] \mid [NP] \text{ without } [NP]$$

Unlike GHKM (Galley et al., 2004), SAMT has the same coverage as Hiero, because it allows non-constituent phrases to get syntactic labels using CCG-style slash notation. Experimentally, we have found that the derivations created using syntactically motivated grammars exhibit more coherent syntactic structure than Hiero and typically result in better reordering, especially for languages with word orders that diverge from English, like Urdu (Baker et al., 2009).

## 3 Specifying Constraints on Translation

Integrating output from specialized modules (like transliterators, morphological analyzers, and modality translators) into the MT pipeline can improve translation performance, particularly for low-resource languages. We have implemented an XML interface that allows external modules to propose alternate translation rules (constraints) for a particular word span to the decoder (Irvine et al., 2010). Processing that is separate from the MT engine can suggest translations for some set of source side words and phrases. The XML format allows for both hard constraints, which must be used, and soft constraints, which compete with standard extracted translation rules, as well as specifying associated feature weights. In addition to specifying translations, the XML format allows constraints on the lefthand side of SCFG rules, which allows constraints like forcing a par-

ticular span to be translated as an NP. We modified **Joshua**’s chart-based decoder to support these constraints.

## 4 Semiring Parsing

In **Joshua**, we use a hypergraph (or packed forest) to compactly represent the exponentially many derivation trees generated by the decoder for an input sentence. Given a hypergraph, we may perform many atomic inference operations, such as finding one-best or  $k$ -best translations, or computing expectations over the hypergraph. For each such operation, we could implement a dedicated dynamic programming algorithm. However, a more general framework to specify these algorithms is semiring-weighted parsing (Goodman, 1999). We have implemented the inside algorithm, the outside algorithm, and the inside-outside speedup described by Li and Eisner (2009), plus the first-order expectation semiring (Eisner, 2002) and its second-order version (Li and Eisner, 2009). All of these use our newly implemented semiring framework.

The first- and second-order expectation semirings can also be used to compute many interesting quantities over hypergraphs. These quantities include expected translation length, feature expectation, entropy, cross-entropy, Kullback-Leibler divergence, Bayes risk, variance of hypothesis length, gradient of entropy and Bayes risk, covariance and Hessian matrix, and so on.

## 5 Word Lattice Input

We generalized the bottom-up parsing algorithm that generates the translation hypergraph so that it supports translation of word lattices instead of just sentences. Our implementation’s runtime and memory overhead is proportional to the size of the lattice, rather than the number of paths in the lattice (Dyer et al., 2008). Accepting lattice-based input allows the decoder to explore a distribution over input sentences, allowing it to select the best translation from among all of them. This is especially useful when **Joshua** is used to translate the output of statistical preprocessing components, such as speech recognizers or Chinese word segmenters, which can encode their alternative analyses as confusion networks or lattices.

## 6 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) that have the same yield. In principle, the goodness of a string is measured by the total probability of its many derivations. However, finding the best string during decoding is then NP-hard. The first version of **Joshua** implemented the Viterbi approximation, which measures the goodness of a translation using only its most probable derivation.

The Viterbi approximation is efficient, but it ignores most of the derivations in the hypergraph. We implemented variational decoding (Li et al., 2009b), which works as follows. First, given a foreign string (or lattice), the MT system produces a hypergraph, which encodes a probability distribution  $p$  over possible output strings and their derivations. Second, a distribution  $q$  is selected that approximates  $p$  as well as possible but comes from a family of distributions  $\mathcal{Q}$  in which inference is tractable. Third, the best string according to  $q$  (instead of  $p$ ) is found. In our implementation, the  $q$  distribution is parameterized by an  $n$ -gram model, under which the second and third steps can be performed efficiently and exactly via dynamic programming. In this way, variational decoding considers all derivations in the hypergraph but still allows tractable decoding.

## 7 Hypergraph-based Discriminative Training

Discriminative training with a large number of features has potential to improve the MT performance. We have implemented the hypergraph-based minimum risk training (Li and Eisner, 2009), which minimizes the *expected loss* of the reference translations. The minimum-risk objective can be optimized by a gradient-based method, where the risk and its gradient can be computed using a second-order expectation semiring. For optimization, we use both L-BFGS<sup>1</sup> and Rprop<sup>2</sup>.

We have also implemented the average Perceptron algorithm and forest-reranking (Li and Khudanpur, 2009b). Since the reference translation may not be in the hypergraph due to pruning or inherent deficiency of the translation grammar, we

need to use an *oracle translation* (i.e., the translation in the hypergraph that is most similar to the reference translation) as a surrogate for training. We implemented the *oracle extraction* algorithm described by Li and Khudanpur (2009a) for this purpose.

Given the current infrastructure, other training methods (e.g., maximum conditional likelihood or MIRA as used by Chiang et al. (2009)) can also be easily supported with minimum coding. We plan to implement a large number of feature functions in **Joshua** so that exhaustive feature engineering is possible for MT.

## 8 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 (Och, 2003).

We have parallelized our MERT module in two ways: parallelizing the computation of metric scores, and parallelizing the search over parameters. The computation of metric scores is a computational concern when tuning to a metric that is slow to compute, such as translation edit rate (Snover et al., 2006). Since scoring a candidate is independent from scoring any other candidate, we parallelize this computation using a multi-threaded solution<sup>3</sup>. Similarly, we parallelize the optimization of the intermediate initial weight vectors, also using a multi-threaded solution.

Another feature is the module's awareness of document information, and the capability to perform optimization of *document-based* variants of the automatic metric (Zaidan, 2010). For example, in document-based Bleu, a Bleu score is calculated *for each document*, and the tuned score is the average of those document scores. The MERT module can furthermore be instructed to target a specific subset of those documents, namely the *tail* subset, where only the subset of documents with the lowest document Bleu scores are considered.<sup>4</sup>

More details on the MERT method and the implementation can be found in Zaidan (2009).<sup>5</sup>

<sup>3</sup>Based on sample code by Kenneth Heafield.

<sup>4</sup>This feature is of interest to GALE teams, for instance, since GALE's evaluation criteria place a lot of focus on translation quality of tail documents.

<sup>5</sup>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>.)

<sup>1</sup><http://en.wikipedia.org/wiki/L-BFGS>

<sup>2</sup><http://en.wikipedia.org/wiki/Rprop>

## 9 Visualization

We created tools for visualizing two of the main data structures used in **Joshua** (Weese and Callison-Burch, 2010). The first visualizer displays hypergraphs. The user can choose from a set of input sentences, then call the decoder to build the hypergraph. The second visualizer displays derivation trees. Setting a flag in the configuration file causes the decoder to output parse trees instead of strings, where each nonterminal is annotated with its source-side span. The visualizer can read in multiple n-best lists in this format, then display the resulting derivation trees side-by-side. We have found that visually inspecting these derivation trees is useful for debugging grammars.

We would like to add visualization tools for more parts of the pipeline. For example, a chart visualizer would make it easier for researchers to tell where search errors were happening during decoding, and why. An alignment visualizer for aligned parallel corpora might help to determine how grammar extraction could be improved.

## 10 Pipeline for Running MT Experiments

Reproducing other researchers' machine translation experiments is difficult because the pipeline is too complex to fully detail in short conference papers. We have put together a workflow framework for designing and running reproducible machine translation experiments using **Joshua** (Schwartz, under review). Each step in the machine translation workflow (data preprocessing, grammar training, MERT, decoding, etc) is modeled by a Make script that defines how to run the tools used in that step, and an auxiliary configuration file that defines the exact parameters to be used in that step for a particular experimental setup. Workflows configured using this framework allow a complete experiment to be run – from downloading data and software through scoring the final translated results – by executing a single Makefile.

This framework encourages researchers to supplement research publications with links to the complete set of scripts and configurations that were actually used to run the experiment. The Johns Hopkins University submission for the WMT10 shared translation task was implemented in this framework, so it can be easily and exactly reproduced.

## Acknowledgements

Research funding was provided by the NSF under grant IIS-0713448, by the European Commission through the EuroMatrixPlus project, and by the DARPA GALE program under Contract No. HR0011-06-2-0001. The views and findings are the authors' alone.

## References

- Kathy Baker, Steven Bethard, Michael Bloodgood, Ralf Brown, Chris Callison-Burch, Glen Copper-smith, Bonnie Dorr, Wes Filardo, Kendall Giles, Anni Irvine, Mike Kayser, Lori Levin, Justin Martineau, Jim Mayfield, Scott Miller, Aaron Phillips, Andrew Philpot, Christine Piatko, Lane Schwartz, and David Zajic. 2009. Semantically informed machine translation (SIMT). SCALE summer workshop final report, Human Language Technology Center Of Excellence.
- David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *NAACL*, pages 218–226.
- David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.
- Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio, June. Association for Computational Linguistics.
- Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *ACL*.
- Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *HLT-NAACL*.
- Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.
- Ann Irvine, Mike Kayser, Zhifei Li, Wren Thornton, and Chris Callison-Burch. 2010. Integrating output from specialized modules in machine translation: Transliteration in joshua. *The Prague Bulletin of Mathematical Linguistics*, 93:107–116.
- Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *EMNLP*, Singapore.
- Zhifei Li and Sanjeev Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *ACL SSST*, pages 10–18.
- Zhifei Li and Sanjeev Khudanpur. 2009a. Efficient extraction of oracle-best translations from hypergraphs. In *Proceedings of NAACL*.

- Zhifei Li and Sanjeev Khudanpur. 2009b. Forest reranking for machine translation with the perceptron algorithm. In *GALE book chapter on "MT From Text"*.
- 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 *WMT09*.
- Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. 2009b. Variational decoding for statistical machine translation. In *ACL*.
- Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *EMNLP-CoNLL*.
- Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*.
- Lane Schwartz and Chris Callison-Burch. 2010. Hierarchical phrase-based grammar extraction in joshua. *The Prague Bulletin of Mathematical Linguistics*, 93:157–166.
- Lane Schwartz. under review. Reproducible results in parsing-based machine translation: The JHU shared task submission. In *WMT10*.
- Matthew Snover, Bonnie J. Dorr, and Richard Schwartz. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.
- Jonathan Weese and Chris Callison-Burch. 2010. Visualizing data structures in parsing-based machine translation. *The Prague Bulletin of Mathematical Linguistics*, 93:127–136.
- 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.
- Omar F. Zaidan. 2010. Document- and tail-based minimum error rate training of machine translation systems. In preparation.
- Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the NAACL-2006 Workshop on Statistical Machine Translation (WMT-06)*, New York, New York.