Parsing-based Machine Translation using an Open Source Toolkit: Joshua for Tamil Language

B.P.Sreeja[1], G.Saratha Devi[2]
[1]Assistant Professor, IT, Karpagam College of Engineering,
[2]Assistant Professor, IT, Karpagam College of Engineering,
Coimbatore, Tamilnadu, India

***Abstract*- Joshua, an open source toolkit for statistical machine translation. It 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. In this paper, it is demonstrated that the toolkit achieves state of the art translation performance on the Tamil -English translation task.**

***Index Terms* – Corpus, SCFGs.**

I. 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, it is described that 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 ma-chine 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]

II. JOSHUA TOOLKIT

When designing the toolkit, the general principles of software engineering is applied 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 hindrance to extensibility in large projects, all extensible components are defined by Java interfaces. Where there is a clear point of departure for researches, 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 in-vestment 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 en-sure 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 a short description about the main functions implemented in the Joshua toolkit.

2.1 Training Corpus Sub-sampling

Rather than inducing a grammar from the full parallel training data, made use of a method pro-posed 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 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 W with a count of less than k, the sentence is selected. When a sentence is selected, the count of every n-gram in 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 Tamil words in the sub sampled 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 je) and reverse translation probability p(ejf) (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.

2.3 Decoding Algorithms

Grammar formalism: The 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 de-code, the decoder generates 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 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), extract the k most likely derivations from the hypergraph.

Parallel and distributed decoding: parallel decoding and a distributed language model is implemented 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, implement three local n-gram language models. Specifically, A straightforward implementation of the n-gram scoring function in Java is provided. 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] 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. A Bloom Filter LM in Joshua, following Talbot and Osborne (2007) is also implemented

III. TRANSLATION TASK RESULTS

3.1 Training and Development Data

A very large Tamil-English training corpus (Callison-Burch, 2009) is assembled by conducting a web crawl that targeted bilingual web sites from the Canadian government, the European Union, and various international organizations like the Amnesty International and the Olympic Commit-tee. 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 Tamil URLs onto English URLs, and considered matching documents to be translations of each other. This yielded 2 million Tamil documents paired with their English equivalents. The sentences and paragraphs in these documents are divided,performed sentence-aligned them using software that IBM Model 1 probabilities into account (Moore, 2002), Filtered and reduplicated the resulting parallel corpus. After discarding 630 thousand sentence pairs which had more than 100 words, the final corpus had 21.9 million sentence pairs with 587,867,024 English words and 714,137,609 Tamil words.

The corpus to the other participants is distributed to use in addition to the Tamil-English parallel corpus (Koehn, 2005), which consists of approximately 1.4 million sentence pairs with 39 million English words and 44 million Tamil words. The translation model was trained on these corpora using the sub sampling descried in Section 2.1

The module is also available as a standalone applica-tion, 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 Tamil words. This is in domain data that was gathered from the same news sources as the test set.

3.2 Translation Scores

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

Baseline: In this system, use the GIZA++ toolkit, a suffix array archi-tecture the SRILM toolkit and minimum error rate training to obtain word alignments, a translation model, language models, and the optimal weights for combining these models, respectively.

Minimum Bayes Risk Rescoring: In this sys-tem, we re-ranked the n-best output of our base-line system using Minimum Bayes Risk rescore the top 300 translations to minimize expected loss under the Bleu metric.

Deterministic Annealing: In this system, in-stead of using the regular MERT the training objective is to minimize the one best error, to use the deterministic annealing training procedure described, the 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 in-tractable. Therefore, most systems use a simple

Viterbi approximation that measures the goodness

System	BLEU-4
Joshua Baseline	28.81
Minimum Bayes Risk Rescoring	30.11
Deterministic Annealing	23.01
Variational Decoding	25.43

Table 1: The uncased BLEU scores on Tamil-English Task.

The test set consists of 2525 segments, each with one reference translation 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 this system, both deterministic annealing (for training) and variational decoding (for decoding) is used.

IV. CONCLUSION

In this paper, a scalable toolkit for parsing-based machine translation is described. It is written in Java and implements all the essential algorithms de-scribed. Chart-parsing, n-gram language model integration, beam- and cube-pruning, and k-best extraction is also described. The toolkit also implements suffix-array grammar extraction and minimum error rate training .Additionally, parallel and distributed computing techniques are exploited to make it scalable. The decoder achieves state of the art translation performance.

REFERENCES

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

[2] 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 (WMT0*

[3] Chris Callison-Burch. 2009. *A 10^9 word parallel cor-pus. In preparation.*

[4] David Chiang. 2007. *Hierarchical phrase-based trans-lation. Computational Linguistics*, 33(2):201–228.

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

[6] 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 Pro-ceedings of the ACL/Coling*.

[7] Liang Huang and David Chiang. 2005. Better k-best parsing. *In Proceedings of the International Work-shop on Parsing Technologies.*

[8] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. *Statistical phrase-based translation. In Pro-ceedings of HLT/NAACL.*

[9] 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.*

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

[11] Shankar Kumar and William Byrne. 2004. *Minimum bayes-risk decoding for statistical machine transla-tion. In Proceedings of HLT/NAACL.*

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

[13] 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.

[14] 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.

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

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

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

[18] Franz Josef Och and Hermann Ney. 2003. *A sys-tematic comparison of various statistical alignment models.* Computational Linguistics, 29(1):19–51.

[19] Franz Josef Och. 2003. *Minimum error rate training for statistical machine translation. In Proceedings of ACL.*

[20] Chris Quirk, Arul Menezes, and Colin Cherry. 2005. *Dependency treelet translation: Syntactically in-formed phrasal smt.* In Proceedings of ACL.

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

[22] Andreas Stolcke. 2002. *SRILM - an extensible lan-guage modeling toolkit.* In Proceedings of the Inter-national Conference on Spoken Language Process-ing, Denver, Colorado, September.

[23] David Talbot and Miles Osborne. 2007. *Randomised language modelling for statistical machine transla-tion. In Proceedings of ACL.*

[24] 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.