PBML

The Prague Bulletin of Mathematical Linguistics
NUMBER 97 APRIL 2012 83–98

Kriya – An end-to-end Hierarchical Phrase-based MT System

Baskaran Sankaran, Majid Razmara, Anoop Sarkar

Simon Fraser University

Abstract

This paper describes *Kriya* – a new statistical machine translation (SMT) system that uses hierarchical phrases, which were first introduced in the Hiero machine translation system (Chiang, 2007). Kriya supports both a grammar extraction module for synchronous context-free grammars (SCFGs) and a CKY-based decoder. There are several re-implementations of Hiero in the machine translation community, but Kriya offers the following novel contributions: (a) Grammar extraction in Kriya supports extraction of the full set of Hiero-style SCFG rules but also supports the extraction of several types of compact rule sets which leads to faster decoding for different language pairs without compromising the BLEU scores. Kriya currently supports extraction of compact SCFGs such as grammars with one non-terminal and grammar pruning based on certain rule patterns, and (b) The Kriya decoder offers some unique improvements in the implementation of cube-pruning, such as increasing diversity in the target language n-best output and novel methods for language model (LM) integration. The Kriya decoder can take advantage of parallelization using a networked cluster. Kriya supports both KENLM and SRILM for language model queries. This paper also provides several experimental results which demonstrate that the translation quality of Kriya compares favourably to the Moses (Koehn et al., 2007) phrase-based system in several language pairs while showing a substantial improvement for Chinese-English similar to Chiang (2007). We also quantify the model sizes for phrase-based and Hiero-style systems and also present experiments comparing variants of Hiero models.

1. Introduction

Hierarchical Phrase-based Machine Translation (Chiang, 2005, 2007) is a prominent approach for Statistical Machine Translation (SMT). It is usually comparable to or better than conventional phrase-based systems for several language pairs.

Corresponding author: baskaran@cs.sfu.ca
Cite as: Baskaran Sankaran, Majid Razmara, Anoop Sarkar. Kriya – An end-to-end Hierarchical Phrase-based MT System. The Prague Bulletin of Mathematical Linguistics No. 97, 2012, pp. 83–98.
doi: 10.2478/v10108-012-0004-y.

In this paper, we present Kriya which implements a hierarchical phrase-based machine translation system which includes a grammar extraction module and a decoder. The name Kriya is the Sanskrit word for *verb* to signify that syntactic parsing techniques can be useful for machine translation.

Kriya is similar to some of the existing hierarchical phrase-based systems, but has some distinguishing features. For example, Kriya uses a unique approach for computing the language model (LM) heuristic in cube-pruning (Chiang, 2007) and it can also optionally support better diversity in the cube-pruning step. Kriya supports extraction of different types of more compact grammars as an alternative to full grammars typically extracted using the synchronous CFG (SCFG) extraction heuristics described in the original Hiero paper. The full grammar is typically associated with issues such as over-generation and search errors (de Gispert et al., 2010) and the use of compact grammars can achieve BLEU scores comparable to full grammar. Kriya also supports shallow-n decoding that leads to faster decoding while maintaining the same BLEU scores as the full decoding.

The rest of the paper is structured as follows. First we review some of the existing Machine Translation systems focusing on Hiero-style systems (Section 2) highlighting specific features. We then give a brief definition of synchronous context-free grammar (SCFG) in Section 3 to set the stage. In Section 4 we describe both grammar extractor and decoder modules interspersed with the features in Kriya. We finally present some experiments (Section 5) comparing Kriya with the well-known phrase-based system Moses which is used to benchmark Kriya's performance for several language pairs.

2. Related Works

`Moses`[1] (Koehn et al., 2007) is an open source toolkit that supports three types of state-of-the-art statistical machine translation systems: phrase-based, hierarchical phrase-based and syntax-based SMT. The toolkit is written in C++ and supports SRILM (Stolcke, 2002), KENLM (Heafield, 2011), randLM (Talbot and Osborne, 2007) and irstLM (Federico et al., 2008) for language model queries. To speed up training, tuning and test steps, Moses supports Oracle Grid Engine[2] (formerly Sun Grid Engine) and Amazon EC2 cloud and implements several memory/speed optimization algorithms. Chart decoding is done by the CKY+ algorithm which enables it to process arbitrary context free grammars with no limitations on the number of terminals or non-terminals in a rule. It also implements Chiang (2007)'s cube-pruning algorithm. Advanced methods such as Factored Models (Koehn and Hoang, 2007), Minimum Bayes Risk (MBR) decoding, Lattice MBR, Consensus Decoding and multiple translation table decoding (to name a few) have been implemented in Moses.

[1] `http://www.statmt.org/moses`

[2] `http://www.oracle.com/us/sun`

Joshua[3] (Li et al., 2009, 2010; Weese et al., 2011) developed at the Center for Language and Speech Processing at the Johns Hopkins University, is an open source machine translation toolkit written in Java that implements most critical algorithms required for hierarchical decoding such as chart-parsing, n-gram language model integration, beam and cube-pruning and k-best extraction. An advantage of this toolkit is that each component in the machine translation pipeline can be run with other components or separately such as Z-MERT (Zaidan, 2009) which is a stand-alone implementation of Och (2002)'s algorithm written in Java. The toolkit implements training corpus sub-sampling by which the most representative subset of the training corpus is used to extract rules from resulting in a faster training phase. In addition, Minimum Bayes Risk, Deterministic Annealing and Variational Decoding algorithms are implemented in this toolkit.

cdec[4] (Dyer et al., 2010) is another translation toolkit written in C++ which allows training and decoding a number of statistical machine translation models, including word-based models, phrase-based models and hierarchical phrase-based models. cdec provides support for Hadoop (an implementation of a distributed filesystem and MapReduce) for parallelization. Input to this system can be a sentence, lattice or context-free forest, which is then transformed to a unified translation forest. Secondly, language model re-scoring, pruning, inference algorithms and k-best derivation extraction are uniformly applied to the generated translation forest. cdec supports a number of optimization algorithms, including Minimum Error Rate Training (MERT) (Och, 2003), LBFGS (Liu and Nocedal, 1989), RPROP (Riedmiller and Braun, 1993) and Stochastic Gradient Descent. Compared to Joshua, cdec uses a smaller memory footprint with the same running time (Dyer et al., 2010).

Jane[5] (Vilar et al., 2010; Stein et al., 2011), RWTH's hierarchical phrase-based translation system, is a more recent open source toolkit which offers similar features. It is written in C++ and includes tools for phrase extraction and translation. Most of the operations can be parallelized by supporting grid engine clusters. The implementation of Jane allows for augmenting the feature set with arbitrary number of additional features as described in Stein et al. (2011). It also offers two ways to support additional models: combination in log-linear fashion and a mechanism to get the model to score a derivation to be incorporated in the main model's score. For tuning, Jane supports three different optimization methods: Minimum Error Rate Training (MERT) (Och, 2003), Margin Infused Relaxed Algorithm (MIRA) (Crammer et al., 2006) and the Downhill Simplex method (Nelder and Mead, 1965). Stein (2011) shows that Jane is 50% faster than Joshua on identical settings.

[3] http://www.sourceforge.net/projects/joshua

[4] https://github.com/redpony/cdec

[5] http://www.hltpr.rwth-aachen.de/jane

3. Synchronous Context-Free Grammars

This section provides a formal definition of a synchronous context-free grammar (SCFG) as a precursor to the discussion of the implementation in Kriya.

Formally a grammar G in a hierarchical phrase-based model is a special case of probabilistic synchronous context-free grammar (PSCFG) that is defined as a 4-tuple: $G = (T, NT, R, R_g)$, where, T and NT are the sets of terminals and non-terminals in G. Hiero grammars typically use two non-terminals X and S where S is the start symbol. R is a set of production rules of the form:

$$X \rightarrow \langle \gamma,\ \alpha,\ \sim \rangle,\ \gamma, \alpha \in \{X \cup T^{+}\} \tag{1}$$

The $\sim$ in the hierarchical rule indicates the alignment indices for the non-terminals in the production rule such that the co-indexed non-terminal pairs are rewritten synchronously. These production rules are combined in the top by the *glue* rules R_g leading to the start symbol S:

$$S \rightarrow \langle X_1,\ X_1 \rangle \tag{2}$$

$$S \rightarrow \langle S_1 X_2,\ S_1 X_2 \rangle \tag{3}$$

where the non-terminal indices indicate synchronous rewriting of the source and target non-terminals having the same index.

4. Kriya

Our implementation of Kriya closely follows the original exposition in Chiang (2007) with extensions that provide several additional features. Broadly, Kriya consists of two independent modules: a *grammar extractor* and a *CKY-based decoder*. Traditionally, grammar extraction has been a bottleneck in Hiero-style translation, due to the massive size of Hiero SCFG grammars and also due to the increasing availability of parallel data. The grammar extractor in Kriya has been designed to efficiently learn translation model even for a very large data set and this is achieved by way of parallelization and optimization. Thus our approach does not resort to sub-sampling to choose a smaller representative training set. Alternately Kriya also supports extraction of several variants of more compact grammars, for example extracting a 1 non-terminal grammar or filtering the full grammar based on certain greedily selected rule patterns (Iglesias et al., 2009).

Kriya decoder currently supports SCFG models for string inputs and features cube-pruning (Chiang, 2007) for integrating the language model scores with the decoder. We introduce a novel approach for improving the heuristic language model scores for the left and right contexts in CKY-based decoder by taking into account the potential position of the target hypothesis fragment in the final candidate. Kriya also supports

shallow-n decoding (de Gispert et al., 2010) for fast decoding without impacting the translation quality for certain close language pairs.

Kriya has been written primarily in Python (versions 2.6 and 2.7). This allows us to test new ideas by quickly implementing them in short duration at the same time keeping the code-base small, manageable and easy to read. On the negative side, Kriya is bit slower mainly due to the well-known speed issues in Python, which we alleviate using several engineering optimizations. These optimizations have resulted in practically acceptable training and decoding speeds in Kriya as we later quantify in Section 5.

4.1. Kriya Grammar Extractor

The Hiero grammar extraction algorithm (Chiang, 2007) starts from the set of initial phrases that are identified by growing the word alignments into longer phrases. Given the initial phrases corresponding to a sentence pair, the heuristic algorithm first designates the smaller initial phrases (such as phrase pairs having non-decomposable alignments) as terminal rules, expanded from a non-terminal X. The algorithm then extracts hierarchical rules by substituting the smaller spans within the larger phrases by the non-terminal X if the phrase pair corresponding to the smaller span has already been identified as a rule. It extracts all possible rules from the initial phrases subject to a maximum of two X non-terminals in a rule such that they do not rewrite adjacent spans in the source side. The Hiero extraction assumes unit count for each initial phrase and distributes this uniformly to all the rules extracted from the phrase. The parameter estimation then proceeds by relative frequency estimation.

Chiang proposed the total number of source side (terminals and non-terminals) terms and a maximum rule length to be 5 and 10 respectively. We found improvements with longer source side rules. In comparison, phrase-based models typically use a maximum phrase length of 7 and sometimes even longer phrases for certain language pairs. The source side length and maximum rule length are customizable parameters in the Kriya rule extraction, to facilitate experiments with different lengths.

A major issue in the extraction of Hiero grammar is the exponential size of the resulting grammar such that the full grammar cannot be held in memory for parameter estimation. Some of the existing Hiero systems use sub-sampling (Weese et al., 2011) to reduce the size of the training corpus and run the grammar extraction on the sub-sampled corpus, resulting in approximate probability estimates. In contrast Kriya uses the entire training data and we use memory optimizations and parallelization to achieve this.

The grammar extractor in Kriya is modularized to run in three phases in order to efficiently extract grammars even for large training corpora. In the first phase the extractor splits the training corpora into smaller chunks and extracts the rules for each chunk by trivial parallelization over the cluster. The second step scans the rules from the individual chunks and filters them based on the source side texts of a tuning or

test set; at the same time collecting the accurate counts for the target phrases in the filtered rules. The final step estimates the forward and reverse probabilities using relative frequency estimation.

The grammar extractor has been customized to the cluster environment (Kriya will soon support the Hadoop framework for extraction and decoding) and thus the extraction can be massively parallelized to efficiently extract Hiero grammar for large corpora. For a smaller data set, it is however possible to estimate the parameters for the full grammar by way of changing the configuration file.

4.1.1. Extracting Compact Grammars

Apart from the original Hiero-style model, Kriya grammar extractor supports the extraction of some variants that are smaller than the full grammar using different pruning[6] strategies. The main motivation for such pruned grammars is twofold, i) to reduce the grammar size and ii) to speed up the decoding enabling faster experiments. In some cases, the resulting compact grammars are suggested to improve the translation by way of reducing search errors (Iglesias et al., 2009; He et al., 2009), which has been contradicted elsewhere (Zollmann et al., 2008; Sankaran et al., 2011).

Kriya supports different approaches to prune the hierarchical phrase-based grammars and here we restrict ourselves to two methods that have been proposed earlier. Other approaches involving Bayesian methods for inducing a compact grammar (Sankaran et al., 2011) or a left-to-right decoding model similar to Watanabe et al. (2006) are under active research as mentioned later in the future directions.

First, the Hiero grammar can be simplified to have just 1 non-terminal, instead of 2 as proposed by Chiang. This grammar eliminates large number of rules, many of which turn out to be composed rules (He et al., 2009) that can be constructed by combining two or more smaller rules leading to spurious ambiguity (Chiang, 2007) during translation. Such 1 non-terminal grammar has been shown to have BLEU scores similar to the full Hiero grammar (Sankaran et al., 2011) for some language pairs such as English-Spanish, but suffer a reduction of about 1 BLEU point for Chinese-English and Urdu-English (Zollmann et al., 2008).

Another alternative is pruning based on rule patterns (Iglesias et al., 2009) which can reduce the size of the grammar. Kriya supports pattern-based filtering, and this is optionally triggered by using a separate configuration file specifying the patterns in the training process. The list of rule patterns to be included/excluded in the extraction process can be specified in the configuration file, in a notation similar to the one used by Iglesias et al. (2009).

[6]Some earlier works use the word *filtering* for this. We prefer *pruning* (or simplification) to indicate the case where some rules are removed that are otherwise applicable in decoding a given tuning/test set, while reserving the word *filtering* to the process of removing rules that will never be applicable for the tuning/test set. The latter is thus risk-free while the former is lossy.

4.2. Kriya Decoder

Kriya currently supports decoding with hierarchical phrase-based models employing CKY-style chart parsing. Given a source sentence f, the decoder finds the target side yield $\hat{e}$ of the best scoring derivation obtained by applying rules in the synchronous context-free grammar.

$$\hat{e} = \mathscr{Y}_e\left(\arg\max_{d \in D(f)} P(d)\right) \tag{4}$$

where, $D(f)$ is the set of derivations attainable from the learned grammar for the source sentence f. The model over derivations $P(d)$ is formulated as a log-linear model (Och and Ney, 2002) employing a set of features $\{\phi_1, \ldots, \phi_M\}$ apart from a language model feature that scores the target yield as $P_{lm}(e)$. The model can be written by factorizing derivation d into its component rules R_d as below.

$$P(d) \propto \left(\prod_{i=1}^{M} \prod_{r \in R_d} \phi_i(r)^{\lambda_i}\right) P_{lm}(e)^{\lambda_{lm}} \tag{5}$$

where, λ_i is the corresponding weight of the feature ϕ_i. The feature weights λ_i are optimized against some evaluation metric (Och, 2003), typically BLEU (Papineni et al., 2002). The default settings in Kriya support the standard features as will be mentioned later.

The decoder parses the source sentence with a modified version of CKY parser with the target side of corresponding derivations simultaneously yielding the candidate translations. The rule parameters and other features are used to score the derivations along with the language model score of the target translation as in Equation 5.

The derivation starts from the leaf cells of the CKY chart corresponding to the source side tokens and proceeds bottom-up. For each cell in the CKY chart, the decoder identifies the applicable rules and analogously to monolingual parsing, the non-terminals in these rules should have corresponding entries in the respective antecedent cells. The target side of the production rules yield the translation for the source span and the translations in the top-most cell correspond to the entire sentence. We encourage readers to refer to Chiang (2007) for more details.

Similar to Chiang, we use cube-pruning, specifically its lazier version (Huang and Chiang, 2007) to integrate the language model scoring in the decoding process. We introduce a novel approach in improving the heuristic language model score by taking into account the likely position of the target hypothesis fragment in the final translation which we explain in detail in the next section.

4.2.1. Novel Enhancements in Cube-pruning

The traditional phrase-based decoders using beam search generate the target hypotheses in the left-to-right order. In contrast, CKY decoders in Hiero-style systems

can freely combine target hypotheses generated in intermediate cells with hierarchical rules in the higher cells. Thus the generation of the target hypotheses are fragmented and out of order in Hiero, compared to the left to right order preferred by n-gram language models.

This leads to challenges in the estimation of language model scores for partial target hypothesis, which is being addressed in different ways in the existing Hiero-style systems. Some systems add a sentence initial marker (<s>) to the beginning of each path and some other systems have this implicitly in the derivation through the translation models. Thus the language model scores for the hypothesis in the intermediate cell are approximated, with the true language model score (taking into account sentence boundaries) being computed in the last cell that spans the entire source sentence.

We introduce a novel improvement in computing the language model scores: for each of the target hypothesis fragments, our approach finds the best position for the fragment in the final sentence and uses the corresponding score. We compute three different scores corresponding to the three positions where the fragment can end up in the final sentence, viz. sentence initial, middle and final: and choose the best score. As an example for fragment t_f consisting of a sequence of target tokens, we compute LM scores for i) <s> t_f, ii) t_f and iii) t_f </s> and use the best score for pruning alone[7].

This improvement significantly reduces the search errors while performing *cube-pruning* (Chiang, 2007) at the cost of additional language model queries. For example, a partial candidate covering a non-final source span might be reordered to the final position in the target translation. If we just compute the LM score for the target fragment as is done normally, this might get pruned early on before being reordered by a production rule. Our approach instead computes the three LM scores and it would correctly use the last LM score which is likely to be the best, for pruning.

Our experiments indicated a small but consistent increase in the BLEU scores and also reduction in the search errors due to this improvement in the computation of LM scores and we also found this to be especially helpful in the shallow-n decoding setting (Section 4.2.2) as we later discuss in experiments (Section 5).

However, additional queries to the language model result in a slight reduction in the decoding speed. This could be partly addressed by saving the three LM scores for both left and right edges with the hypothesis and reusing them appropriately when either or both edges remain unchanged. Secondly following the general strategy, we exploit n-gram state information in KENLM (Heafield, 2011) to query the language model for incremental target fragment following a stored state.

As a second enhancement, Kriya optionally supports improved diversity in cube-pruning by allowing a fixed (typically 3) number of candidates for each cube that are not represented in the cell. These hypotheses are included in the cell in addition to

[7]This ensures the the LM score estimates are never underestimated for pruning. We retain the LM score for fragment (case ii) for estimating the score for the full candidate sentence later.

the hypotheses pushed into the stack through cube-pruning. We found cube-pruning diversity to be marginally effective in different settings as we discuss later.

4.2.2. Shallow-n Decoding

Shallow-n grammars (de Gispert et al., 2010) are a class of grammars that restrict the number of successive hierarchical rules in a derivation in order to reduce the over-generation caused by large Hiero grammars. While this has restricted reordering capability compared to full Hiero, the degree of reordering can be customized to the requirements of specific language pairs by way of changing n. For example, Shallow-1 grammar might be sufficient for language pairs such as English-French and Arabic-English, whereas higher order shallow grammars are required for Chinese-English because of their large syntactic divergence. As a direct consequence of the reduction in the search space, shallow-n decoding results in substantially faster decoding.

Formally, a Shallow-n grammar G is defined as a 5-tuple: $G = (N, T, R, R_g, S)$, such that T is a set of finite terminals and N a set of finite non-terminals $\{X^0, \ldots, X^N\}$. As earlier R_g refers to the glue rules that rewrite the start symbol S:

$$S \rightarrow \langle X,\ X \rangle \tag{6}$$

$$S \rightarrow \langle SX,\ SX \rangle \tag{7}$$

R is the set of finite production rules in G and has two types, viz. hierarchical (8) and terminal (9). The hierarchical rules at each level n are additionally conditioned to have *at least* one X^{n-1} non-terminal in them. The $\sim$ in the hierarchical rule serves as the index for aligning the non-terminals such that the co-indexed non-terminal pair can be rewritten synchronously.

$$X^n \rightarrow \langle \gamma,\ \alpha,\ \sim \rangle,\ \gamma, \alpha \in \{\{X^{n-1}\} \cup T^+\} \tag{8}$$

$$X^0 \rightarrow \langle \gamma,\ \alpha \rangle, \qquad \gamma, \alpha \in T^+ \tag{9}$$

Kriya supports Shallow-n decoding, without requiring additional non-terminals to be explicitly created in the Hiero grammar (this is similar to other implementations of this idea). We simply keep track of the number of hierarchical nestings in the partial hypotheses stored in the decoder as part of the hypothesis state. We find the shallow grammars to be comparable to closer language pairs such as English-French and English-Spanish, but the translation performance suffers for Arabic-English and Chinese-English without additional hacks.

4.3. Optimizing Feature Weights

Kriya uses the well-known MERT algorithm (Och, 2003) for optimizing feature weights and it has been integrated with both MERT implementation in Moses (Koehn

et al., 2007) and zMERT (Zaidan, 2009). Recently we have also added an implementation of Pairwise Ranking Optimization (Hopkins and May, 2011) (PRO) within Kriya for tuning the feature weights.

5. Experiments

In this section we present experiments to evaluate Kriya on several language pairs. We use five different language pairs in our experiments - representing a wide range of diversities, such as close languages (English-French), translating into a slightly more inflected language than English (English-Spanish) and languages with high syntactic divergence (Chinese-English). Table 1 shows some statistics about the corpora used for our experiments.

Language pair	Corpus	Train/ Tune/ Test	Language Model
English-Spanish	WMT10 (Europarl + News commentary)	1.7 M/ 5078/ 2489	WMT10 *train* + UN
English-French		1.7 M/ 5078/ 2489	WMT10 *train*
French-English		1.7 M/ 5078/ 2489	Gigaword
Chinese-English	*Train*: HK + GALE Phase-1 *Tune*: MTC parts 1 & 3; *Test*: MTC part 4	2.3 M/ 1928/ 919	Gigaword
Arabic-English	ISI automatically extracted Parallel text	1.1 M/ 1982/ 987	Gigaword

Table 1. Corpus Statistics - English-French uses a 4-gram LM and other pairs use 5-gram LMs. Chinese-English experiments use four references for tuning and testing.

First we present experimental results to benchmark Kriya's performance in all these language pairs by comparing it with the well-known Moses phrase-based system. We used standard settings for Moses in all these experiments except for the maximum phrase length, which we set to 7. For Kriya models, we set the total source side terms to be 7 for Chinese-English and Arabic-English and 5 for others. We ran Kriya in standard setting which includes 8 features: inverse and direct phrase translation probabilities $p(f|e)$ and $p(e|f)$; inverse and direct lexical weights $p_l(f|e)$ and $p_l(e|f)$; phrase penalty; word penalty; glue rule penalty and language model.

For both Moses and Kriya, we trained lower-cased models for Chinese-English and Arabic-English, while training true-cased models for the rest. All the experiments described here use MERT (Och, 2003) for optimizing the weights of the features. The BLEU (Papineni et al., 2002) scores are computed using the official NIST evaluation script.[8]

[8] ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

Language Pair	Moses		Kriya	
	Model size	*BLEU*	*Model size*	*BLEU*
English-Spanish	154.6	28.12	632.8	**28.19**
English-French	81.8	23.48	519.9	**23.54**
French-English	81.9	26.15	439.9	***26.63***
Arabic-English	68.0	37.31	331.5	***37.74***
Chinese-English	83.6	24.48	286.1	***25.96***

Table 2. Model sizes and BLEU Scores - Model sizes are in millions of rules. Bold face indicates best BLEU score for each language pair and italicized figures point to statistically significant improvements assuming significance level $\alpha = 0.1$.

Table 2 lays out the BLEU scores as well as the model sizes of the Moses and Kriya phrase tables. As shown, the hierarchical phrase-based system always has larger models (unfiltered phrase table size), ranging between 342.2% and 635.5% of their phrase-based counterparts. In terms of BLEU scores, Kriya results in higher BLEU scores in all the language pairs with the best improvement coming for Chinese-English, confirming the results of Chiang (2007). Further, Kriya achieves statistically significant improvements for Arabic-English and English-French experiments.

As mentioned earlier, the huge model size of the Hiero systems slows down decoding and earlier research has proposed two different approaches for this: Shallow-n decoding as opposed to the full decoding restricts the depth of non-glue hierarchical rules in the derivation. Orthogonal to this, more compact models that are substantially smaller than the full Hiero models can be used with full decoding. In this experiment we compare the basic variants of these two approaches in terms of BLEU scores and model size.

In the Shallow decoding setting, we use shallow-1 thus restricting the hierarchical rules in the grammar to directly rewrite into terminal rules with the glue rules freely combining the hierarchical rules. We compare this with a simpler Hiero grammar consisting of one non-terminal and this generally results in a compact model compared to the original Hiero model. Note that these two ideas are orthogonal and hence can be combined; however we generally find them to result in poor performance and so we ignore the combination experiments here.

The experimental results are summarized in Tables 3 and 4. We find the simpler model consisting of one non-terminal employing full decoding to be competitive to the full model for closer language pairs such as French-English and Arabic-English at the same time clocking higher decoding speed. However, we see a reduction in the BLEU score for Chinese-English as has also been found by Zollmann et al. (2008). We thus hypothesize that 1 NT models have the same expressive power as the regular Hiero models (with 2 non-terminals), at least for languages with little syntactic diver-

Language Pair	Original (2 NT)/ Shallow-1	Compact (1 NT)/ Full	
	BLEU	*BLEU*	*Model size*
English-Spanish	27.70	**28.15**	351.3 (55.5%)
English-French	23.22	**23.48**	290.3 (55.8%)
French-English	**26.67**	**26.66**	248.2 (56.4%)
Arabic-English	37.15	**37.71**	161.4 (49.0%)
Chinese-English	24.04	25.25	154.2 (53.9%)

Table 3. Shallow-1 decoding vs. Compact (1 NT) model - Bold face indicates BLEU scores comparable to the original Hiero model in Table 2. Size of the compact 1 NT model as a % of original Hiero model is given within the brackets.

Model	Decoding level	Decoding time
Original (2 NT)	Full	0.71
Original (2 NT)	Shallow-1	0.24
Compact (1 NT)	Full	0.50

Table 4. Kriya Decoding time (in secs/word) for Chinese-English translation

gence. They also reduce the model size almost by half achieving a highest reduction of 51% for Arabic-English.

Shallow-1 decoding achieves highest decoding speed among the three but suffers a small reduction in the BLEU score except for French-English and incurs a larger reduction of 1.9 BLEU points for Chinese-English. It is three times faster than full decoding and twice faster than the 1 NT model. Higher order shallow decoding (not shown here), for example shallow-2 for Arabic-English and shallow-3 for Chinese-English achieve competitive performance but shallow-3 case suffers substantial reduction in decoding speed and is only marginally faster than full decoding.

In analyzing the effect of our novel approach of LM integration, we compare our approach to the naive approach of computing heuristic LM score that prefixes a beginning of sentence marker (<s>) to the candidate hypothesis. We believe that our approach will reduce search errors by finding better scoring candidates than the usual ones and we look at two parameters to test this: translation quality as measured by BLEU scores and search errors. In measuring search errors, we compared the model scores of the candidates (with one-to-one correspondence) in the N-best lists obtained by the two approaches and computed the percent of high scoring candidates in each N-best list.

In the shallow setting, our method improved the BLEU scores by 0.4 for both Arabic-English and Chinese-English. Our approach also obtained a much better N-

best list, with 94.6% and 77.3% of candidates in the N-best list having better scores than the naive approach for Chinese-English and Arabic-English respectively. In the full decoding setting the improvements, were lower with 69% and 57% of candidates obtaining better model scores for the two language pairs in the same order and the BLEU score increasing by 0.3.

We also found cube-pruning diversity to be useful in our experiments in Arabic-English and Chinese-English. We set cube-pruning diversity to be 3 (where, top-3 hypotheses from each unrepresented cube being added to the hypotheses in the corresponding CKY-chart cell) and in different settings involving full decoding and shallow-n decoding, the translation quality improved by a small margin of 0.25 BLEU points.

6. Future Directions

Kriya continues to be in active development and we are planning to add several new features. Currently, it supports the TORQUE cluster environment for parallelizing training and optimization processes. We are currently working towards supporting the MapReduce framework, specifically using a Hadoop setup. In recent developments in the Kriya decoder, we are exploring a new left-to-right decoder similar to Watanabe et al. (2006) in order to take advantage of its straight-forward language model integration in order to achieve a faster decoding time. Furthermore, we plan to extend the novel *ensemble* framework (Razmara et al., 2012) for decoding, which has already been implemented in Kriya. The ensemble decoder dynamically combines information from multiple translation and/or language models and can better exploit training data from different domains. This can particularly be useful in scenarios such as domain adaptation, multi-source translation. In inducing better SCFG grammars, we are working on efficient alternatives to the usual heuristic rule extraction approach, extending our earlier work using a Bayesian model (Sankaran et al., 2011). Finally, we are interested in incorporating syntax in our research with Kriya, in order to exploit the divergence between different languages.

7. Conclusion

We presented *Kriya* – a new implementation of hierarchical phrase-based machine translation which has novel features and which achieves competitive performance in several language pairs. Kriya's grammar extractor can efficiently extract Hiero grammars from large training sets and supports extraction of several compact Hiero grammar variants. The decoder currently uses CKY-based decoding, and we are currently working on left-to-right decoding to speed up the decoder. Kriya is under active development and several new features are being planned with specific focus on Bayesian models for extracting compact grammars, ensemble decoding, support for MapReduce framework and so on. We also presented experimental results on five language

pairs using the Kriya system comparing different variants of the basic Hiero-style decoder and different Hiero-style grammars.

Bibliography

Chiang, David. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of Annual Meeting of Association of Computational Linguistics*, pages 263–270, 2005.

Chiang, David. Hierarchical phrase-based translation. *Computational Linguistics*, 33, 2007.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

de Gispert, Adrià, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. Hierarchical phrase-based translation with weighted finite-state transducers and Shallow-n grammars. *Computational Linguistics*, 36, 2010.

Dyer, Chris, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the Annual Meeting of Association of Computational Linguistics 2010 System Demonstrations track*, ACLDemos '10, pages 7–12, 2010.

Federico, Marcello, Nicola Bertoldi, and Mauro Cettolo. IRSTLM: an open source toolkit for handling large scale language models. In *INTERSPEECH 2008, 9th Annual Conference of the International Speech Communication Association*, pages 1618–1621. ISCA, 2008.

He, Zhongjun, Yao Meng, and Hao Yu. Discarding monotone composed rule for hierarchical phrase-based statistical machine translation. In *Proceedings of the 3rd International Universal Communication Symposium*, pages 25–29, 2009.

Heafield, Kenneth. KenLM: faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, 2011.

Hopkins, Mark and Jonathan May. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362. Association for Computational Linguistics, 2011.

Huang, Liang and David Chiang. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 144–151. Association for Computational Linguistics, 2007.

Iglesias, Gonzalo, Adrià de Gispert, Eduardo R. Banga, and William Byrne. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 380–388, 2009.

Koehn, Philipp and Hieu Hoang. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic, 2007. Association for Computational Linguistics.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej

Bojar, Alexandra Constantin, and Evan Herbst. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics, 2007.

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

Li, Zhifei, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Ann Irvine, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Ziyuan Wang, Jonathan Weese, and Omar Zaidan. Joshua 2.0: A toolkit for parsing-based machine translation with syntax, semirings, discriminative training and other goodies. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 133–137. Association for Computational Linguistics, 2010.

Liu, Dong C. and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

Nelder, John A. and Roger Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

Och, Franz Josef. Minimum error rate training in statistical machine translation. In *Proceedings of the Annual Meeting of Association of Computational Linguistics*, pages 160–167, 2003.

Och, Franz Josef and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the Annual Meeting of Association of Computational Linguistics*, pages 295–302, 2002.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of Association of Computational Linguistics*, pages 311–318, 2002.

Razmara, Majid, George Foster, Baskaran Sankaran, and Anoop Sarkar. Mixing multiple translation models in statistical machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju, Republic of Korea, July 2012. Association for Computational Linguistics.

Riedmiller, Martin and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, 1993.

Sankaran, Baskaran, Gholamreza Haffari, and Anoop Sarkar. Bayesian extraction of minimal SCFG rules for hierarchical phrase-based translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 533–541, 2011.

Stein, Daniel, David Vilar, Stephan Peitz, Markus Freitag, Matthias Huck, and Hermann Ney. A guide to Jane, an open source hierarchical translation toolkit. In *The Prague Bulletin of Mathematical Linguistics, No. 95*, pages 5–18, 2011.

Stolcke, Andreas. SRILM – an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, 2002.

Talbot, David and Miles Osborne. Randomised language modelling for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 512–519, Prague, Czech Republic, 2007. Association for Computational Linguistics.

Vilar, David, Daniel Stein, Matthias Huck, and Hermann Ney. Jane: open source hierarchical translation, extended with reordering and lexicon models. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 262–270. Association for Computational Linguistics, 2010.

Watanabe, Taro, Hajime Tsukada, and Hideki Isozaki. Left-to-right target generation for hierarchical phrase-based translation. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING)*, pages 777–784. Association for Computational Linguistics, 2006.

Weese, Jonathan, Juri Ganitkevitch, Chris Callison-Burch, Matt Post, and Adam Lopez. Joshua 3.0: Syntax-based machine translation with the Thrax grammar extractor. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 478–484. Association for Computational Linguistics, 2011.

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

Zollmann, Andreas, Ashish Venugopal, Franz Och, and Jay Ponte. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1145–1152, 2008.

Address for correspondence:
Baskaran Sankaran
baskaran@cs.sfu.ca
School of Computing Science
Simon Fraser University
8888 University Dr, Burnaby
BC V5A 1S6, Canada