Two Improvements to Left-to-Right Decoding for Hierarchical Phrase-based Machine Translation

Maryam Siahbani and **Anoop Sarkar**
School of Computing Science
Simon Fraser University
Burnaby BC. Canada
msiahban,anoop@cs.sfu.ca

Abstract

Left-to-right (LR) decoding (Watanabe et al., 2006) is promising decoding algorithm for hierarchical phrase-based translation (Hiero) that visits input spans in arbitrary order producing the output translation in left to right order. This leads to far fewer language model calls, but while LR decoding is more efficient than CKY decoding, it is unable to capture some hierarchical phrase alignments reachable using CKY decoding and suffers from lower translation quality as a result. This paper introduces two improvements to LR decoding that make it comparable in translation quality to CKY-based Hiero.

1 Introduction

Hierarchical phrase-based translation (Hiero) (Chiang, 2007) uses a lexicalized synchronous context-free grammar (SCFG) extracted from word and phrase alignments of a bitext. Decoding for Hiero is typically done with CKY-style decoding with time complexity $O(n^3)$ for source input with n words. Computing the language model score for each hypothesis within CKY decoding requires two histories, the left and the right edge of each span, due to the fact that the target side is built inside-out from sub-spans (Heafield et al., 2011; Heafield et al., 2013).

LR-decoding algorithms exist for phrase-based (Koehn, 2004; Galley and Manning, 2010) and syntax-based (Huang and Mi, 2010; Feng et al., 2012) models and also for hierarchical phrase-based models (Watanabe et al., 2006; Siahbani et al., 2013), which is our focus in this paper.

Watanabe et al. (2006) first proposed left-to-right (LR) decoding for Hiero (LR-Hiero henceforth) which uses beam search and runs in $O(n^2b)$ in practice where n is the length of source sentence and b is the size of beam (Huang and Mi, 2010). To simplify target generation, SCFG rules are constrained to be prefix-lexicalized on target side aka Griebach Normal Form (GNF). Throughout this paper we abuse the notation for simplicity and use the term GNF grammars for such SCFGs. This constraint drastically reduces the size of grammar for LR-Hiero in comparison to Hiero grammar (Siahbani et al., 2013). However, the original LR-Hiero decoding algorithm does not perform well in comparison to current state-of-the-art Hiero and phrase-based translation systems. Siahbani et al. (2013) propose an augmented version of LR decoding to address some limitations in the original LR-Hiero algorithm in terms of translation quality and time efficiency.

Although, LR-Hiero performs much faster than Hiero in decoding and obtains BLEU scores comparable to phrase-based translation system on some language pairs, there is still a notable gap between CKY-Hiero and LR-Hiero (Siahbani et al., 2013). We show in this paper using instructive examples that CKY-Hiero can capture some complex phrasal re-orderings that are observed in language pairs such as Chinese-English that LR-Hiero cannot (c.f. Sec.3).

We introduce two improvements to LR decoding of GNF grammars: (1) We add *queue diversity* to the cube pruning algorithm for LR-Hiero, and (2) We extend the LR-Hiero decoder to capture all the hierarchical phrasal alignments that are reachable in CKY-Hiero (restricted to using GNF grammars). We evaluate our modifications on three language pairs and show that LR-Hiero can reach the translation scores comparable to CKY-Hiero in two language pairs, and reduce the gap between Hiero and LR-Hiero on the third one.

2 LR Decoding with Queue Diversity

LR-Hiero uses a constrained lexicalized SCFG which we call a GNF grammar: $X \rightarrow \langle \gamma, \bar{b}\, \beta \rangle$ where γ is a string of non-terminal and terminal symbols, $\bar{b}$ is a string of terminal symbols and β is a possibly empty sequence of non-terminals. This ensures that as each rule is used in a derivation,

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 221–226,
October 25-29, 2014, Doha, Qatar.

Algorithm 1: LR-Hiero Decoding

1: Input sentence: $\mathbf{f} = f_0 f_1 \dots f_n$
2: $\mathcal{F}$ = FutureCost($\mathbf{f}$) (Precompute future cost[1] for spans)
3: $S_0 = \{\}$ (Create empty initial stack)
4: $h_0 = (\langle s \rangle, [[0, n]], \emptyset, \mathcal{F}_{[0,n]})$ (Initial hypothesis 4-tuple)
5: Add h_0 to S_0 (Push initial hyp into first Stack)
6: **for** $i = 1, \dots, n$ **do**
7: $cubeList = \{\}$ (MRL is max rule length)
8: **for** $p = max(i - \text{MRL}, 0), \dots, i - 1$ **do**
9: $\{G\}$ = Grouped(S_p) (based on the first uncovered span)
10: **for** $g \in \{G\}$ **do**
11: $[u, v] = g_{span}$
12: R = GetSpanRules($[u, v]$)
13: **for** $R_s \in R$ **do**
14: $cube = [g_{hyps}, R_s]$
15: Add $cube$ to $cubeList$
16: S_i = Merge($cubeList, \mathcal{F}$) (Create stack S_i and add new hypotheses to it, see Figure 1)
17: **return** $\arg\max(S_n)$

18: **Merge**($CubeList, \mathcal{F}$)
19: $heapQ = \{\}$
20: **for** each (H, R) in $cubeList$ **do**
21: $hypList$ = getBestHypotheses($(H, R), \mathcal{F}, d$) (d best hypotheses of each cube)
22: **for** each h' in $hypList$ **do**
23: push($heapQ, (h'_c, h', [H, R])$ (Push new hyp in queue)
24: $hypList = \{\}$
25: **while** $|heapQ| > 0$ and $|hypList| < K$ **do**
26: $(h'_c, h', [H, R])$ = pop($heapQ$) (pop the best hypothesis)
27: push($heapQ, GetNeighbours([H, R])$ (Push neighbours to queue)
28: Add h' to $hypList$
29: **return** $hypList$

the target string is generated from left to right. The rules are obtained from a word and phrase aligned bitext using the rule extraction algorithm in (Watanabe et al., 2006).

LR-Hiero decoding uses a top-down depth-first search, which strictly grows the hypotheses in target surface ordering. Search on the source side follows an Earley-style search (Earley, 1970), the dot jumps around on the source side of the rules based on the order of nonterminals on the target side. This search is integrated with beam search or cube pruning to find the k-best translations.

Algorithm 1 shows the pseudocode for LR-Hiero decoding with cube pruning (Chiang, 2007) (CP). LR-Hiero with CP was introduced in (Siahbani et al., 2013). In this pseudocode, we have introduced the notion of *queue diversity* (explained below). However to understand our change we need to understand the algorithm in more detail.

[1]The future cost is precomputed in a way similar to the phrase-based models (Koehn et al., 2007) using only the terminal rules of the grammar.

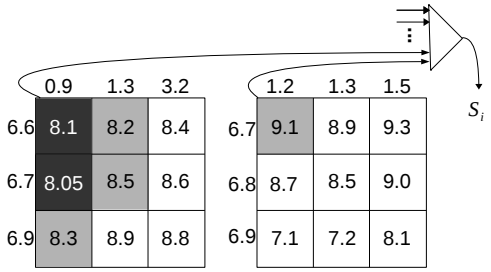


Figure 1: Cubes (grids) are fed to a priority queue (triangle) and generated hypotheses are iteratively popped from the queue and added to stack S_i. Lower scores are better. Scores of rules and hypotheses appear on the top and left side of the grids respectively. Shaded entries are hypotheses in the queue and black ones are popped from the queue and added to S_i.

Each source side non-terminal is instantiated with the legal spans given the input source string, e.g. if there is a Hiero rule $\langle aX_1, a'X_1 \rangle$ and if a only occurs at position 3 in the input then this rule can be applied to span $[3, i]$ for all i, $4 < i \leq n$ for input of length n and source side X_1 is instantiated to span $[4, i]$. A worked out example of how the decoder works is shown in Figure 2. Each partial hypothesis h is a 4-tuple (h_t, h_s, h_{cov}, h_c): consisting of a translation prefix h_t, a (LIFO-ordered) *list* h_s of uncovered spans, source words coverage set h_{cov} and the hypothesis cost h_c. The initial hypothesis is a null string with just a sentence-initial marker $\langle s \rangle$ and the list h_s containing a span of the whole sentence, $[0, n]$. The hypotheses are stored in stacks $S_0, \dots, S_n$, where S_p contains hypotheses covering p source words just like in stack decoding for phrase-based SMT (Koehn et al., 2003).

To fill stack S_i we consider hypotheses in each stack S_p[2], which are first partitioned into a set of groups $\{G\}$, based on their first uncovered span (line 9). Each group g is a 2-tuple (g_{span}, g_{hyps}), where g_{hyps} is a list of hypotheses which share the same first uncovered span g_{span}. Rules matching the span g_{span} are obtained from routine *GetSpanRules*. Each g_{hyps} and possible R_s create a cube which is added to $cubeList$.

The *Merge* routine gets the best hypotheses from all cubes (see Fig.1). Hypotheses (rows) and columns (rules) are sorted based on their scores. $GetBestHypotheses((H, R), \mathcal{F}, d)$ uses current hypothesis H and rule R to produce new hypotheses. The first best hypothesis, h' along with its score h'_c and corresponding cube (H, R) is placed in a priority queue $heapQ$ (triangle in Figure 1 and line 23 in Algorithm 1). Iteratively the K best

[2]As the length of rules are limited (at most MRL), we can ignore stacks with index less than $i - \text{MRL}$

rules	hypotheses
	$\langle s \rangle [0, 15]$
G 1)$\langle Taiguo\ shi\ X_1 / Thailand\ X_1 \rangle$	$\langle s \rangle$ Thailand [2,15]
G 2)$\langle yao\ X_1 / wants\ X_1 \rangle$	$\langle s \rangle$Thailand wants [3,15]
G 3)$\langle liyong\ X_1 / to\ utilize\ X_1 \rangle$	$\langle s \rangle$Thailand wants to utilize [4,15]
4)$\langle zhe\ bi\ qian\ X_1 / this\ money\ X_1 \rangle$	$\langle s \rangle$Thailand wants to utilize this money [7,15]
5)$\langle X_1\ zhuru\ geng\ duo\ X_2 / to\ inject\ more\ X_2 X_1 \rangle$	$\langle s \rangle$Thailand wants to utilize this money to inject more [12,15][7,9]
6)$\langle liudong\ X_1 / circulating\ X_1 \rangle$	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating [13,15][7,9]
G 7)$\langle zijin\ X_1 / capital\ X_1 \rangle$	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating capital [14,15][7,9]
8)$\langle ./. \rangle$	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating capital . [7,9]
9)$\langle xiang\ jingji / to\ the\ economy \rangle$	$\langle s \rangle$Thailand wants to utilize this money to inject more circulating capital . to the economy$\langle /s \rangle$

Figure 2: The process of translating the Chinese sentence in Figure 3(b) in LR-Hiero. Left side shows the rules used in the derivation (G indicates glue rules as defined in (Watanabe et al., 2006)). The hypotheses column shows the translation prefix and the ordered list of yet-to-be-covered spans.

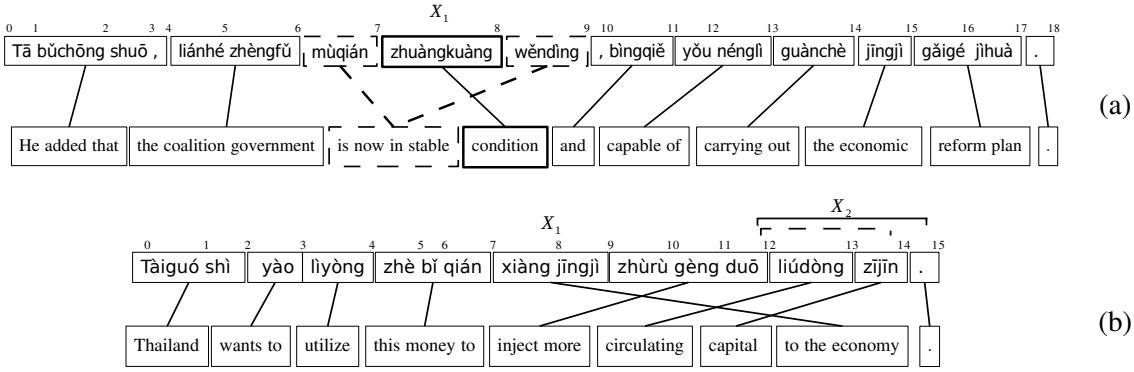


Figure 3: Two Chinese-English sentence pairs from devset data in experiments. (a) Correct rule cannot be matched to [6,18], our modifications match the rule to the first subspan [6,9] (b) LR-Hiero detects a wrong span for X_2 [12,15], we modify the rule matching match X_2 to all subspans [12,13], [12,14] and [12,15], corresponding to 3 hypotheses.

hypotheses in the queue are popped (line 26) and for each hypothesis its neighbours in the cube are added to the priority queue (line 27). Decoding finishes when stack S_n has been filled.

The language model (LM) score violates the hypotheses generation assumption of CP and can cause search errors. In Figure 1, the topmost and leftmost entry of the right cube has a score worse than many hypotheses in the left cube due to the LM score. This means the right cube has hypotheses that are ignored. This type of search error hurts LR-Hiero more than CKY-Hiero, due to the fact that hypotheses scores in LR-Hiero rely on a future cost, while CKY-Hiero uses the inside score for each hypothesis. To solve this issue for LR-Hiero we introduce the notion of *queue diversity* which is the parameter d in $GetBestHypotheses((H, R), \mathcal{F}, d)$. This parameter guarantees that each cube will produce at least d candidate hypotheses for the priority queue. d=1 in standard cube pruning for LR-Hiero (Siahbani et al., 2013). We apply the idea of diversity at queue level, before generating K best hypothesis, such that the *GetBestHypotheses* routine generates d best hypotheses from each cube and all these hypotheses are pushed to the priority queue (line 22-23). We fill each stack differently from CKY-Hiero and so queue diversity is different from lazy cube pruning (Pust and Knight, 2009) or cube growing (Huang and Chiang, 2007; Vilar and Ney, 2009; Xu and Koehn, 2012).

3 Capturing Missing Alignments

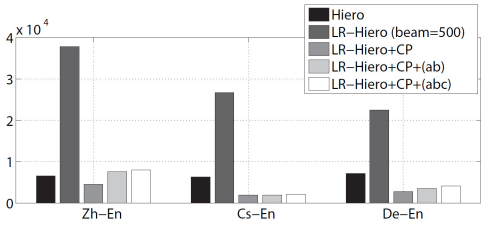
Figure 3(a) and Figure 3(b) show two examples of a common problem in LR-Hiero decoding. The decoder steps for Figure 3(b) are shown in Figure 2. The problem occurs in Step 5 of Figure 2 where rule #5 is matched to span $[7, 15]$. During decoding LR-Hiero maintains a stack (*last-in-first-out*) of yet-to-be-covered spans and tries to translate the first uncovered span (span $[7, 15]$ in Step 5). LR-Hiero should match rule #5 to span $[7, 15]$, therefore X_2 is forced to match span $[12, 15]$ which leads to the translation of span $[7, 9]$ (corresponding to X_1) being reordered around it

	Corpus	Train/Dev/Test
Cs-En	Europarl(v7) + CzEng(v0.9); News commentary(nc) 2008&2009; nc 2011	7.95M/3000/3003
De-En	Europarl(v7); WMT2006; WMT2006	1.5M/2000/2000
Zh-En	HK + GALE phase-1; MTC part 1&3; MTC part 4	2.3M/1928/919

Table 1: Corpus statistics in number of sentences. Tuning and test sets for Chinese-English has 4 references.

Model	Cs-En	De-En	Zh-En
Hiero	20.77	25.72	27.65
LR-Hiero (Watanabe et al., 2006)	20.72	25.10	25.99
LR-Hiero+CP (Siahbani et al., 2013)	20.15	24.83	-
LR-Hiero+CP (QD=1)	20.68	25.14	24.44
LR-Hiero+CP (QD=15)	-	-	26.10
LR-Hiero+CP+(ab)	20.88	25.22	26.55
LR-Hiero+CP+(abc)	20.89	25.22	26.52

(a) BLEU scores for different baselines and modifications of this paper. QD=15 for Zh-En in last three rows.



(b) Average number of language model queries.

Table 2: (a) BLEU (b) LM calls

causing the incorrect translation in Step 9. If we use the same set of rules for translation in Hiero (CKY-based decoder), the decoder is able to generate the correct translation for span $[7, 14]$ (it works bottom-up and generate best translation for each source span). Then it combines translation of $[7, 14]$ with translation of spans $[0, 7]$ and $[14, 15]$ using glue rules (monotonic combination).

In Figure 3(a) monotonic translations after span $[6, 9]$ are out of reach of the LR-Hiero decoder which has to use the non-terminals to support the reordering within span $[6, 9]$. In this example the first few phrases are translated monotonically, then for span $[6, 18]$ we have to apply rule $\langle$muqian X_1 wending, is now in stable $X_1\rangle$ to obtain the correct translation. But this rule cannot be matched to span $[6, 18]$ and the decoder fails to generate the correct translation. While CKY-Hiero can apply this rule to span $[6, 9]$, generate correct translation for this span and monotonically combine it with translation of other spans ($[0, 6]$, $[9, 18]$).

In both these cases, CKY-Hiero has no difficulty in reaching the target sentence *with the same GNF rules*. The fact that we have to process spans as they appear in the stack in LR-Hiero means that we cannot combine arbitrary adjacent spans to deal with such cases. So purely bottom-up decoders such as CKY-Hiero can capture the alignments in Figure 3 but LR-Hiero cannot.

We extend the LR-Hiero decoder to handle such cases by making the GNF grammar more expressive. Rules are partitioned to three types based on the right boundary in the source and target side. The rhs after the $\Rightarrow$ shows the new rules we create within the decoder using a new non-terminal X_r to match the right boundary.

$$\begin{aligned} &\text{(a)}\ \langle \gamma\bar{a}, \bar{b}\beta\rangle \Rightarrow \langle \gamma\bar{a}X_r, \bar{b}\beta X_r\rangle \\ &\text{(b)}\ \langle \gamma X_n, \bar{b}\beta X_n\rangle \Rightarrow \langle \gamma X_n X_r, \bar{b}\beta X_n X_r\rangle \\ &\text{(c)}\ \langle \gamma X_n, \bar{b}\beta X_m\rangle \Rightarrow \langle \gamma X_n X_r, \bar{b}\beta X_m X_r\rangle \end{aligned} \quad (1)$$

where γ is a string of terminals and non-terminals, $\bar{a}$ and $\bar{b}$ are terminal sequences of source and target respectively, β is a possibly empty sequence of non-terminals and X_n and X_m are different non-terminals distinct from X_r[3]. The extra non-terminal X_r lets us add a new yet-to-be-covered span to the bottom of the stack at each rule application which lets us match any two adjacent spans just as in CKY-Hiero. This captures the missing alignments that could not be previously captured in the LR-Hiero decoder[4].

In Table 4 we translated devset sentences using forced decoding to show that our modifications to LR-Hiero in this section improves the alignment coverage when compared to CKY-Hiero.

4 Experiments

We evaluate our modifications to LR-Hiero decoder on three language pairs (Table 1): German-English (De-En), Czech-English (Cs-En) and Chinese-English (Zh-En).

[3]In rule type (c) X_n will be in β and X_m will be in γ.

[4]For the sake of simplicity, in rule type (b) we can merge X_n and X_r as they are in the same order on both source and target side.

We use a 5-gram LM trained on the Gigaword corpus and use KenLM (Heafield, 2011). We tune weights by minimizing BLEU loss on the dev set through MERT (Och, 2003) and report BLEU scores on the test set. Pop limit for Hiero and LR-Hiero+CP is 500 and beam size LR-Hiero is 500. Other extraction and decoder settings such as maximum phrase length, etc. were identical across settings. To make the results comparable we use the same feature set for all baselines, Hiero as well (including new features proposed by (Siahbani et al., 2013)).

We use 3 baselines: (i) our implementation of (Watanabe et al., 2006): LR-Hiero with beam search (LR-Hiero) and (ii) LR-Hiero with cube pruning (Siahbani et al., 2013): (LR-Hiero+CP); and (iii) Kriya, an open-source implementation of Hiero in Python, which performs comparably to other open-source Hiero systems (Sankaran et al., 2012).

Table 3 shows model sizes for LR-Hiero (GNF) and Hiero (SCFG). Typical Hiero rule extraction excludes phrase-pairs with unaligned words on boundaries (loose phrases). We use similar rule extraction as Hiero, except that exclude non-GNF rules and include loose phrase-pairs as terminal rules.

Table 2a shows the translation quality of different systems in terms of BLEU score. Row 3 is from (Siahbani et al., 2013)[5]. As we discussed in Section 2, LR-Hiero+CP suffers from severe search errors on Zh-En (1.5 BLEU) but using queue diversity (QD=15) we fill this gap. We use the same QD(=15) in next rows for Zh-en. For Cs-En and De-En we use regular cube pruning (QD=1), as it works as well as beam search (compare rows 4 and 2).

We measure the benefit of the new modified rules from Section 3: (*ab*): adding modifications for rules type (a) and (b); (*abc*): modification of all rules. We can see that for all language pairs (ab) constantly improves performance of LR-Hiero, significantly better than LR-Hiero+CP and LR-Hiero (p-value$<$0.05) on Cs-En and Zh-En, evaluated by MultEval (Clark et al., 2011). But modifying rule type (c) does not show any improvement due to spurious ambiguity created by type (c) rules.

[5]We report results on Cs-En and De-En in (Siahbani et al., 2013). Row 4 is the same translation system as row 3 (LR-Hiero+CP). We achieve better results than our previous work (Siahbani et al., 2013) (row 4 vs. row 3) due to bug corrections and adding loose phrases as terminal rules.

Model	**Cs-En**	**De-En**	**Zh-En**
Hiero	1,961.6	858.5	471.9
LR-Hiero	266.5	116.0	100.9

Table 3: Model sizes (millions of rules).

Model	Cs-En	De-En	Zh-En
Hiero	318	351	187
LR-Hiero	278	300	132
LR-Hiero+(abc)	338	361	174

Table 4: No. of sentence covered in forced decoding of a sample of sentences from the devset. We improve the coverage by 31% for Chinese-English and more than 20% for the other two language pairs.

Figure 2b shows the results in terms of average number of language model queries on a sample set of 50 sentences from test sets. All of the baselines use the same wrapper to KenLM (Heafield, 2011) to query the language model, and we have instrumented the wrapper to count the statistics. In (Siahbani et al., 2013) we discuss that LR-Hiero with beam search (Watanabe et al., 2006) does not perform at the same level of state-of-the-art Hiero (more LM calls and less translation quality). As we can see in this figure, adding new modified rules slightly increases the number of language model queries on Cs-En and De-En so that LR-Hiero+CP still works 2 to 3 times faster than Hiero. On Zh-En, LR-Hiero+CP applies queue diversity (QD=15) which reduces search errors and improves translation quality but increases the number of hypothesis generation as well. LR-Hiero+CP with our modifications works substantially faster than LR-Hiero while obtain significantly better translation quality on Zh-En.

Comparing Table 2a with Figure 2b we can see that overall our modifications to LR-Hiero decoder significantly improves the BLEU scores compared to previous LR decoders for Hiero. We obtain comparable results to CKY-Hiero for Cs-En and De-En and remarkably improve results on Zh-En, while at the same time making 2 to 3 times less LM calls on Cs-En and De-En compared to CKY-Hiero.

Acknowledgments

This research was partially supported by NSERC, Canada RGPIN: 262313 and RGPAS: 446348 grants to the second author. The authors wish to thank Baskaran Sankaran for his valuable discussions and the anonymous reviewers for their helpful comments.

References

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

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 176–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, February.

Yang Feng, Yang Liu, Qun Liu, and Trevor Cohn. 2012. Left-to-right tree-to-string decoding with prediction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1191–1200, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michel Galley and Christopher D. Manning. 2010. Accurate non-hierarchical phrase-based translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974, Los Angeles, California, June. Association for Computational Linguistics.

Kenneth Heafield, Hieu Hoang, Philipp Koehn, Tetsuo Kiso, and Marcello Federico. 2011. Left language model state for syntactic machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 183–190, San Francisco, California, USA, 12.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. 2013. Grouping language model boundary words to speed K-Best extraction from hypergraphs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia, USA, 6.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *In Proc. of the Sixth Workshop on Statistical Machine Translation*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *In ACL 07*.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 273–283, Cambridge, MA, October. Association for Computational Linguistics.

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

Philipp Koehn, 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. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *AMTA*, pages 115–124.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Pust and Kevin Knight. 2009. Faster mt decoding through pervasive laziness. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 141–144, Boulder, Colorado, June. Association for Computational Linguistics.

Baskaran Sankaran, Majid Razmara, and Anoop Sarkar. 2012. Kriya - an end-to-end hierarchical phrase-based mt system. *The Prague Bulletin of Mathematical Linguistics (PBML)*, 97(97):83–98, apr.

Maryam Siahbani, Baskaran Sankaran, and Anoop Sarkar. 2013. Efficient left-to-right hierarchical phrase-based translation with improved reordering. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, USA, October. Association for Computational Linguistics.

David Vilar and Hermann Ney. 2009. On lm heuristics for the cube growing algorithm. In *Annual Conference of the European Association for Machine Translation*, pages 242–249, Barcelona, Spain, may.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proc. of ACL*.

Wenduan Xu and Philipp Koehn. 2012. Extending hiero decoding in moses with cube growing. *Prague Bull. Math. Linguistics*, 98:133–.