Expressive Hierarchical Rule Extraction for Left-to-Right Translation

Maryam Siahbani msiahban@cs.sfu.ca
Anoop Sarkar anoop@cs.sfu.ca
School of Computing Science,
Simon Fraser University,
Burnaby, V5A 1S6, Canada

Abstract

Left-to-right (LR) decoding Watanabe et al. (2006) is a 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 the constrained SCFG grammar used in LR-Hiero (GNF) with at most two non-terminals is unable to account for some complex phrasal reordering. Allowing more non-terminals in the rules results in a more expressive grammar. LR-decoding can be used to decode with SCFGs with more than two non-terminals, but the CKY decoders used for Hiero systems cannot deal with such expressive grammars due to a blowup in computational complexity. In this paper we present a dynamic programming algorithm for GNF rule extraction which efficiently extracts sentence level SCFG rule sets with an arbitrary number of non-terminals. We analyze the performance of the obtained grammar for statistical machine translation on three language pairs.

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. This is due to the fact that the target side is built inside-out from sub-spans (Heafield et al., 2011, 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) was the first to propose a left-to-right (LR) decoding algorithm for Hiero (henceforth we refer to LR decoding for Hiero as LR-Hiero) 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, synchronous context-free grammar (SCFG) rules are constrained to be prefix-lexicalized on target side, aka Greibach Normal Form (GNF). Throughout this paper we abuse the notation for simplicity and use the term GNF grammars for such SCFGs[1]. Siahbani et al. (2013) propose an augmented version of LR decoding to

[1] Although any monolingual context-free grammar can be converted to Greibach Normal Form, there is no algorithm

address some limitations in the original LR-Hiero algorithm in terms of translation quality and time efficiency.

Hiero (and LR-Hiero) rule extraction heuristics apply constraints on the length of initial phrase pairs considered for rule extraction, number and configuration of non-terminals in order to avoid excessively large grammars. Thus, obtained rules cannot capture all possible alignments on language pairs with complex reordering. Allowing more non-terminals in the rules is not practical in CKY based decoders because the computational complexity of decoding increases exponentially with the increase in the rank of the grammar (that is, the number of non-terminals permitted in the right hand side of the CFG rules). However, LR decoding is a viable alternative which can efficiently apply these types of rules while keeping quadratic time complexity by using a variant of the dotted rules used in the Earley parsing algorithm for parsing monolingual CFGs.

Standard Hiero rule extraction used to extract GNF grammars (Watanabe et al., 2006; Siahbani et al., 2013) is a brute-force search algorithm which considers all possible replacement of sub-phrases with non-terminals. Despite the constraints on rule configuration, rule extraction is still a bottleneck and it is usually achieved by way of parallelization and optimization. Increasing the length of initial phrase pairs or number of non-terminals exponentially increases the time complexity. In this paper we propose a dynamic programming algorithm for GNF rule extraction that is linear in the output length (the number of GNF rules). We use this algorithm to extract GNF rules with different number of non-terminals including sentence level rules and analyze the effect of these rules in LR-Hiero translation system on three language pairs: Chinese-English, Czech-English and German-English.

2 Left-to-Right Decoding

LR-Hiero uses a constrained lexicalized SCFG. The target-side rules are constrained to be prefix lexicalized, for simplicity called *GNF rules*[2]:

$$X \rightarrow \langle \gamma, \bar{b}\, \beta \rangle \tag{1}$$

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, the target string is generated from left to right.

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). 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 source side X_1 is instantiated to span $[4, n]$, for input of length n. A worked out example of how the decoder works is shown in Figure 1. 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 which includes future cost and a score computed based on feature values (using a log-linear model). 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 $S_0, \ldots, S_n$, where S_p contains hypotheses covering p source words just as in stack decoding for phrase-based SMT (Koehn et al., 2003).

---

to convert an arbitrary SCFG to a weakly equivalent SCFG with rules constrained to be prefix-lexicalized on the target side.

[2]Greibach Normal Form (GNF). Just the target side is prefix lexicalized (GNF form), not the synchronous grammar.

[3]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.

Algorithm 1 LR-Hiero Decoding with CP

1: Input sentence: $\mathbf{f} = f_0 f_1 \dots f_n$
2: $\mathcal{F} = \text{FutureCost}(\mathbf{f})$ (Precompute future cost[3] 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)
17: **return** $\underset{h \in S_n}{\text{argmin}}\ h_c$
18: **Merge**($CubeList, \mathcal{F}$)
19: $heapQ = \{\}$
20: **for** each (H, R) in $cubeList$ **do**
21: h' = getBestHypotheses($(H, R), \mathcal{F}$) (best hypotheses of cubes)
22: push($heapQ, (h'_c, h', [H, R])$) (Push new hyp in the queue)
23: $hypList = \{\}$
24: **while** $|heapQ| > 0$ and $|hypList| < K$ **do**
25: $(h'_c, h', [H, R])$ = pop($heapQ$) (pop the best hypothesis)
26: push($heapQ, GetNeighbours([H, R])$ (Push neighbours to queue)
27: Add h' to $hypList$
28: **return** $hypList$

To fill stack S_i we consider hypotheses in each stack S_p[4], 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. $GetBestHypotheses((H, R), \mathcal{F})$ 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 $heapQ$ (line 22 in Algorithm 1). Iteratively the K best hypotheses in the queue are popped (line 25) and for each hypothesis its neighbours in the cube are added to the priority queue (line 26). Decoding finishes when stack S_n has been filled.

3 Rule Extraction

Hiero uses a synchronous context free grammar (SCFG), $X \rightarrow \langle \gamma, \alpha \rangle$, where X is a non-terminal, γ and α are strings of terminals and non-terminals (Chiang, 2005, 2007). Unlike typical SCFGs, the rules are lexicalized on the right hand side with at least one aligned word pair in source and target.

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

rules	hypotheses $\langle h_t, h_s, h_{cov}, h_c \rangle$
	$\langle$ <s> , $[[0,8]]$, $---------$, 0$\rangle$
G 1) $X \rightarrow \langle$ *schuler* X_1 / *students* $X_1 \rangle$	$\langle$ <s> *students* , $[[1,8]]$, $*-------$, 3.5$\rangle$
G 2) $X \rightarrow \langle X_1$ *heban* X_2 / *have* $X_1 X_2 \rangle$	$\langle$ <s> *students have* , $[[1,6][7,8]]$, $*-----*-$, 5.7$\rangle$
3) $X \rightarrow \langle X_1$ *noch nicht* X_2 / *not yet* $X_2 X_1 \rangle$	$\langle$ <s> *students have not yet* , $[[5,6][1,3][7,8]]$, $*--**-*-$, 10.2$\rangle$
4) $X \rightarrow \langle$ *gemacht* / *done* $\rangle$	$\langle$ <s> *students have not yet done* , $[[1,3][7,8]]$, $*--****-$, 12.4$\rangle$
5) $X \rightarrow \langle$ *ihre arbeit* / *their work* $\rangle$	$\langle$ <s> *students have not yet done their work* , $[[7,8]]$, $*******-$, 15.1$\rangle$
6) $X \rightarrow \langle$ ./. $\rangle$	$\langle$ <s> *students have not yet done their work* . </s> , $[]$, $********$, 15.6$\rangle$

Figure 1: The process of translating a German-English sentence pair in LR-Hiero. Word alignment is shown in Figure 4 (a). Left side shows the rules used in the derivation (G indicates glue rules as defined in Watanabe et al. (2006)). The hypotheses column shows 4-tuple partial hypotheses: the translation prefix, h_t, the ordered list of yet-to-be-covered spans, h_s, source word coverage vector, h_{cov} and cost h_c (cost includes future cost and hypothesis cost, but we just show hypothesis cost in this figure).

Chiang (2007) places certain constraints on the extracted rules in order to simplify decoding. This includes limiting the maximum number of non-terminals (rule arity) to two and disallowing any rule with consecutive non-terminals on the source language side. It further limits the length of the initial phrase-pair to a *maximum phrase length*. For translating sentences longer than the maximum phrase pair length, the decoder relies on additional glue rules $S \rightarrow \langle X, X \rangle$ and $S \rightarrow \langle SX, SX \rangle$ that allow monotone combination of phrases. The glue rules are used when no rules could match or the span length is larger than the maximum phrase-pair length.

LR-Hiero generates the target hypotheses left to right, but for synchronous context-free grammar (SCFG) as used in Hiero. Therefore LR-Hiero restricts the grammar to GNF rules (equation 1). The rules are obtained from a word and phrase aligned bitext using a rule extraction algorithm (see Section 3.1). To overcome data sparsity and obtain better generalization, four glue rules are added for each terminal rule $\langle \bar{f}, \bar{e} \rangle$. The glue rules allow reordering as well as monotone combination of phrases :

$$\begin{aligned} X &\rightarrow \langle \bar{f} X_1, \bar{e} X_1 \rangle \qquad X \rightarrow \langle X_1 \bar{f} X_2, \bar{e} X_1 X_2 \rangle \\ X &\rightarrow \langle X_1 \bar{f}, \bar{e} X_1 \rangle \qquad X \rightarrow \langle X_1 \bar{f} X_2, \bar{e} X_2 X_1 \rangle \end{aligned} \tag{2}$$

3.1 Hiero Rule Extraction

The Hiero grammar extraction (Chiang, 2007) starts from the set of initial phrases that are identified by growing the word alignments into longer phrases. Given the initial phrases of a sentence pair, the extraction algorithm first designates the smaller initial phrases as terminal rules. Then it extracts hierarchical rules by substituting the smaller spans within the larger phrases by a non-terminal X. It extracts all possible rules from the initial phrases subject to a maximum of two non-terminals in a rule such that they are not adjacent 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. LR-Hiero uses similar method for grammar extraction, except any rules violating GNF form on the target side are excluded (Watanabe et al., 2006; Siahbani et al., 2013).

This algorithm is a brute-force search which considers all possible replacement of sub-phrases with non-terminals. Although Hiero and LR-Hiero use initial phrase pairs of limited length (usually 10) and grammar is limited to at most two non-terminals, rule extraction is still a bottleneck and it is generally achieved by way of parallelization and optimization. Increasing the length of initial phrase pairs or number of non-terminals exponentially increases the time complexity. In section 3.3 we propose a dynamic programming algorithm for GNF rule

extraction from a sentence pair that is linear in the output length (the number of GNF rules).

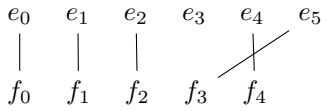


Figure 2: Example phrase pair with alignments.

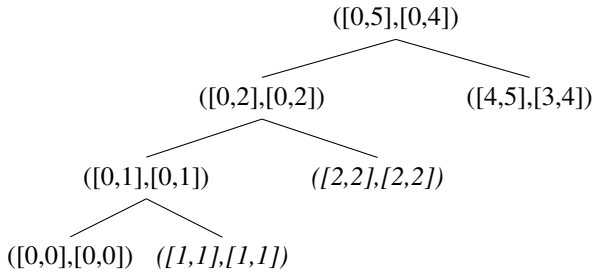


Figure 3: Decomposed alignment tree for the example alignment in Fig. 2.

3.2 Phrase Pair Extraction

Unlike Hiero rule extraction, we do not limit the length of initial phrase pairs and extract rules from all phrase pairs (including whole sentence pairs) in the training data. A modified version of the algorithm by (Zhang et al., 2008) is used to efficiently extract phrase pairs. For a phrase pair with a given alignment as shown in Figure 2, Zhang et al. (2008) generalize the $O(n+K)$ time algorithm for computing all K common intervals of two different permutations of length n. The contiguous blocks of the alignment are captured as the nodes in the alignment tree and the tree structure (for example, phrase pair in Figure 2 is shown in Figure 3). The italicized nodes form a left-branching chain in the alignment tree and the sub-spans of this chain also lead to alignment nodes that are not explicitly captured in the tree (Please refer to (Zhang et al., 2008) for details).

3.3 GNF Extraction

We first explain the rule extraction algorithm using a working example, then discuss correctness of the algorithm. Let $pp = (\bar{f}, \bar{e})$ be a source-target phrase pair, where $\bar{f}$ and $\bar{e}$ are corresponding phrases on source and target side. We define *largest right sub-phrase*, for a target interval $[i, j]$, as the largest phrase pair (in terms of length of target side) with right boundary j on the target side, and denote it by $LRS(i, j)$:

$$\begin{aligned} LRS(i,j) &= \underset{(\bar{f},\bar{e})\in S(i,j)}{\operatorname{argmax}} |\bar{e}| \\ S(i,j) &= \{(\bar{f},\bar{e}) | (\bar{f},\bar{e}) \in \mathcal{P}, |\bar{e}| < |j-i|, \bar{e}.end() = j\} \end{aligned} \tag{3}$$

where $\mathcal{P}$ is a set of all phrase pairs, $|\bar{e}|$ denotes length of $\bar{e}$, $\bar{e}.end()$ returns the last index of the span corresponding to $\bar{e}$ (in the target sentence). S is empty set for phrase pair with target spans of length one ($i = j$). For example in Figure 4, the $LRS[1, 5]$ is $\langle noch\ nicht\ gemacht,\ not\ yet\ done \rangle$[5]. LRS can be precomputed for all span lengths in

[5]In Figure 4, we identify phrase pairs to target spans, $LRS[1, 5] = (3, 5)$.

Algorithm 2 GNF Rule Extraction

Input $\mathbf{f}(f_1 \dots f_n), \mathbf{e}(e_1 \dots e_m), \mathbf{A}$ ($\mathbf{A}$ is alignment)
$\mathcal{P} =$ ExtractPhrases$(\mathbf{f}, \mathbf{e}, \mathbf{A})$ (generate all possible phrase pairs, increasingly sorted based on length of target side)
$LRS =$ RightSubPhrases$(\mathcal{P}, m)$ (precompute largest right sub-phrases)
$\mathcal{R} = \{\}$
for $pp \in \mathcal{P}$ **do**
  $(i, j) = \bar{e}_{pp}$ (target span of pp)
  $R_{i,j} = \{\}$ (rules for target span [i,j])
  $curr\_rule = pp$ (create a terminal rule)
  AddRule($R_{i,j}$, $curr\_rule$)
  $t = j$
  while $t \geq i$ **do**
    $pp' = LRS[(i, t)]$
    if pp' is None **then**
      break
    $(k, t) = \bar{e}_{pp'}$ (target span of pp')
    for each $r \in R_{k,t}$ **do**
      $r' =$ Substitute($curr\_rule$, pp', r)
      AddRule($R_{i,j}$, r')
    $curr\_rule =$ Substitute($curr\_rule$, pp', X) (replace subphrase with a non-terminal)
    AddRule($R_{i,j}$, $curr\_rule$)
    $t = k - 1$
  $LRS[(i, j)] = pp$ (update LRS)
  Add $R_{i,j}$ to $\mathcal{R}$
return $\mathcal{R}$

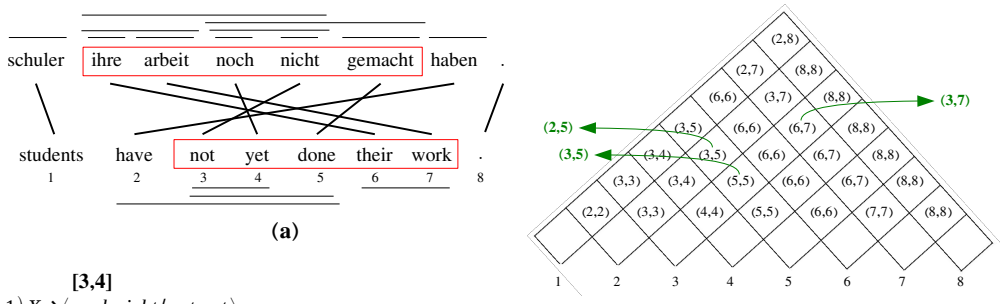
RightSubPhrases$(\mathcal{P}, m)$
  $LRS = \{\}$
  for $l = 2, \dots, m$ **do**
    for $i = 1, \dots, m - l$ **do**
      $j = i + l - 1$
      if $\exists\, pp \in \mathcal{P}, \bar{e}_{pp} == (i + 1, j)$ **then**
        $LRS[(i, j)]$= pp
      elif $(i + 1, j) \in LRS$ **then**
        $LRS[(i, j)]$= $LRS[(i + 1, j)]$
  return LRS

$O(n^2)$, where n is target sentence length (routine *RightSubPhrases* in Algorithm 2). Figure 4 (b) shows the chart of LRS computed by *RightSubPhrases* for sentence pair in Figure 4 (a). Each cell corresponds to a span on the target side.

Algorithm 2 shows the pseudocode for GNF rule extraction. It is a dynamic programming algorithm that extracts GNF rules for phrase pairs (gradually from small to large phrase pairs). It works bottom up and fills a chart, R, on the target sentence. Each cell $R_{i,j}$ keeps all rules that can cover a phrase pair $pp = (\bar{f}, \bar{e})$, where $\bar{e}$ corresponds to span $[i, j]$ on the target sentence[6]. At the end, it returns $\mathcal{R}$ which is the union of rules for all target spans (i.e. all possible phrase pairs).

First, routine *ExtractPhrases* extracts all phrase pairs $\mathcal{P}$ and sorts them increasingly based

[6] If there is not such a phrase pair, $R_{i,j}$ will be left empty.



[3,4]

1) $X \rightarrow \langle noch\ nicht / not\ yet \rangle$
2) $X \rightarrow \langle X_1\ nicht / not\ X_1 \rangle$

[6,7]

1) $X \rightarrow \langle ihre\ arbeit / their\ work \rangle$
2) $X \rightarrow \langle ihre\ X_1 / their\ X_1 \rangle$

[3,5]

1) $X \rightarrow \langle noch\ nicht\ gemacht / not\ yet\ done \rangle$
2) $X \rightarrow \langle noch\ nicht\ X_1 / not\ yet\ X_1 \rangle$
3) $X \rightarrow \langle X_2\ nicht\ X_1 / not\ X_2\ X_1 \rangle$

[3,7]

i=3, t=7
LRS(3,7)=[6,7]
k=6

curr_rule $\langle ihre\ arbeit\ noch\ nicht\ gemacht / not\ yet\ done\ their\ work \rangle$

1) $X \rightarrow \langle ihre\ arbeit\ noch\ nicht\ gemacht / not\ yet\ done\ their\ work \rangle$
2) $X \rightarrow \langle ihre\ X_1\ noch\ nicht\ gemacht / not\ yet\ done\ their\ X_1 \rangle$
3) $X \rightarrow \langle X_1\ noch\ nicht\ gemacht / not\ yet\ done\ X_1 \rangle$

t=5
LRS(3,5)=[3,5]
k=3

curr_rule $\langle X_1\ noch\ nicht\ gemacht / not\ yet\ done\ X_1 \rangle$

4) $X \rightarrow \langle X_1\ noch\ nicht\ X_2 / not\ yet\ X_2\ X_1 \rangle$
5) $X \rightarrow \langle X_1\ X_3\ nicht\ X_2 / not\ X_3\ X_2\ X_1 \rangle$

t=2

curr_rule $\langle X_1\ X_2 / X_2\ X_1 \rangle$

(c)

[2,8]

1) $X \rightarrow \langle ihre\ arbeit\ noch\ nicht\ gemacht\ haben\ . / have\ not\ yet\ done\ their\ work\ . \rangle$
2) $X \rightarrow \langle ihre\ arbeit\ noch\ nicht\ gemacht\ haben\ X_1 / have\ not\ yet\ done\ their\ work\ X_1 \rangle$
3) $X \rightarrow \langle ihre\ X_2\ noch\ nicht\ gemacht\ haben\ X_1 / have\ not\ yet\ done\ their\ X_2\ X_1 \rangle$
4) $X \rightarrow \langle X_2\ noch\ nicht\ gemacht\ haben\ X_1 / have\ not\ yet\ done\ X_2\ X_1 \rangle$
5) $X \rightarrow \langle X_2\ noch\ nicht\ X_3\ haben\ X_1 / have\ not\ yet\ X_3\ X_2\ X_1 \rangle$
6) $X \rightarrow \langle X_2\ haben\ X_1 / have\ X_2\ X_1 \rangle$

$\langle X_2\ X_3\ X_1 / X_3\ X_2\ X_1 \rangle$

(d)

Figure 4: GNF rule extraction for a German-English sentence pair. (a) bars above (below) the source (target) words indicate phrase-pairs. (b) LRS chart for this sentence, filled by *RightSubPhrase* (green arrows shows some cells corresponding to phrase pairs which are updated during rule extraction). The span above each set of rules shows the target side of the corresponding phrase pair. (c) Extracting rules for span [3,7]: rule #2 is created using rules of span [6,7], #3 replacing [6,7] with non-terminal, rules #4, #5 created from span [3,5]. Invalid rules are shown in grey. (d) Extracting rules for span [2,8].

on their target length (line 2). LRS is computed for all target spans by *RightSubPhrases*. Then, in a *for* loop on all phrase pairs, chart of the rules will be filled in a bottom up manner, small to large spans (lines 5-23). For each initial phrase pair a terminal rule is created and added to $R_{i,j}$ (line 8). Then, using rules from smaller phrase pairs, more rules are generated (line 11-21).

The largest right sub-phrase, pp' is obtained for initial phrase pair pp in line 12 (note that t is set to the right boundary of pp (i.e. j) at the beginning). Target span of pp', $[k, t]$, is used to retrieve rules for pp', stored in $R_{k,t}$. Replacing each rule of pp' in our $curr\_rule$[7], (*Substitute*

[7] It is the initial phrase pair pp at the beginning.

routine) results in a new rule for pp (lines 16-18). And as the last rule that can be generated using pp', the whole pp' in $curr\_phr$ is replaced with a non-terminal (line 19).

All rules for pp which includes rules from pp' have been generated, thus we can safely replace pp' with a non-terminal and continue to generate more rules by replacing other parts of pp with non-terminals. $curr\_rule$ is updated to the last rule, t is updated to the index span of pp' on the target (line 21)[8]. The algorithm repeats the loop to find another sub-phrase pair in $curr\_rule$, and continues until no sub-phrase pair is found (or we reach the beginning of target phrase (t equals i)). When all the rules for pp are computed, $LRS[(i,j)]$ is updated to pp so that it can be used in larger phrases. In fact $LRS[(i,j)]$ should always show the largest right subphrase which its rules have already been extracted. Note that only if $[i,j]$ corresponds to a phrase-pair, $LRS[(i,j)]$ is updated (in Figure 4(b), some updated cells are shown in green).

Figure 4(c) shows how algorithm extracts rules for span [3,7]: at first $curr\_rule$ is equal to the initial phrase pair. rule #1 is a terminal rule; $LRS[3,7]$ is $\langle ihre\ arbeit,\ their\ work\rangle$ (target span [6,7]), rule #2 is generated using rules for [6,7]; rule #3 is the result of replacing [6,7] with a non-terminal. Then $curr\_rule$ and t are updated. $LRS[3,5]$ is sub-phrase pair $\langle noch\ nicht\ gemacht,\ not\ yet\ done\rangle$, (rules for this phrase pair have been already computed and consequently $LRS[3,5]$ has been updated to ([4,6],[3,5])). Rules #4 and #5 are generated using rules for span [3,5]. Then $curr\_rule$ and t are updated. As no lexical item remains in the target side of $curr\_rule$, the algorithm stops.

AddRule verifies the rule configuration like the number of non-terminals and non-adjacent non-terminals on the source side. If the rule is valid, it is added to the corresponding cell, $R_{i,j}$ (e.g. rule #5 is not valid because of adjacent non-terminals on the source side).

3.3.1 Correctness

We show that for a given phrase pair, this algorithm extracts all possible Hiero style SCFG rules which are in GNF format on the target side (the same as Hiero brute-force rule extraction). Given a phrase pair $pp = (\bar{f}, \bar{e})$ with target span $[i,j]$, $LRS(i,j)$ shows the largest subproblem that can be optimally used to generate rules for pp, denoted by $R_{i,j}$.
Optimal structure: $R_{i,j}$ consists of two disjoint sets

$$\begin{aligned} R_s &= \{r | r = Substitute(pp, pp', r') \forall r' \in R_{i',j}\} \\ R_x &= \{r | r.pos(pp') = X\} \end{aligned} \tag{4}$$

where $r.pos(pp')$ denotes the interval of pp' in r. R_s is the set of rules obtained by replacing pp' in pp with each rule of $R_{i',j}$, while R_x is the set of rules having non-terminal X in position of pp' (in source and target side). GNF rules on the target side (equation 1), ends with a non-terminal (if there is one) and there is no lexical item between non-terminals. Assuming this we can consider two states for each $r \in R_{i,j}$: (a) r has some lexical term in $r.pos(pp')$; (b) r has a non-terminal in position pp'. Case (a) is equal to set R_s: this type of rules can have non-terminal just in the interval of pp' (because any non-terminal out of pp' violates GNF format on the target side). And if there is a rule of this type it corresponds to a rule in $R_{i',j}$. Consequently case (b) is equal to set R_x. It means that any $r \notin R_s$, should replace a non-terminal instead of pp' (otherwise violates GNF format on the target side). Computing R_x corresponds to a smaller problem $[i, i'-1]$ (let's define $t = i'-1$) which can be solved in a similar way. If $[i,t]$ is target side of a phrase pair (like $[3,5]$ in Figure 4(c)), we just need to use rules in $R_{i,t}$ to generate more rules and keep all valid rules. Otherwise we repeat the process: find the largest sub-problem, $LRS[i,t] = (k,t)$, use $R_{k,t}$ to generate more rules, then replace $[k,t]$ in the rule

[8] $[i,t]$ always shows the lexical part of the target side.

	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 parallel-tex + GALE ph-1; MTC parts 1&3; MTC part4	2.3M/1928/919

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

Model (msl)	Cs-En	De-En	Zh-En
SCFG (7)	1,961.6	858.5	471.8
GNF (7)	306.3	116.0	100.9
GNF-4 (10)	380.9	214.9	190.0

Table 2: Model sizes in millions of rules. Maximum source length (msl) is shown in brakets.

Model	Cs-En	De-En	Zh-En
SCFG	318	351	187
GNF-2	278	300	132
GNF-4	306	375	163

Table 3: No. of sentence covered in forced decoding.

with a non-terminal, update the rule and continue. It stops when target side is entirely covered by non-terminals (Figure 4(d) shows an example of this type).

Using this optimal structure, we iteratively solve the problem in three steps: (1) find the largest sub-problem ($LRS(i, j)$), (2) use its solution to generate some rules (R_s), (3) reduce the problem to a smaller problem (R_x).

Unaligned words in the target language (not present in our example) makes the computation of LRS more complex. For example if target word index j is unaligned, then $LRS[i, j]$ for all $i<j$ will be empty and the algorithm stops without considering subphrases at the left side of the unaligned word. To avoid this problem, unaligned words on the target side will be attached to the closest left phrase pair (if it exists) during computation of LRS.

4 Experiments

To evaluate our rule extraction algorithm, we use it to extract the grammar for LR-Hiero on three language pairs: German-English (De-En), Czech-English (Cs-En) and Chinese-English (Zh-En). Table 1 shows the details of datasets.

We use 2 baselines: (i) LR-Hiero in Python (we use the implementation described in (Siahbani and Sarkar, 2014)); (ii) Kriya (Sankaran et al., 2012b), an open-source implementation of Hiero in Python (available on `https://github.com/sfu-natlang/Kriya`) which performs comparably to other open-source Hiero systems. Both systems are in Python and use the same LM wrapper which allows us to make a fair comparison of LM calls and time differences in decoding.

We use rule extraction of Kriya to extract Hiero (SCFG) and modify it to extract LR-Hiero (GNF). Both grammars use similar configuration and settings: rule arity 2, maximum source length 7, initial phrase pairs of length at most 10. We use our rule extraction algorithm to extract GNF rules from all initial phrase pairs (any length), rule arity 1 to 4, maximum source rule length 10. Like Hiero, we filter rules with adjacent non-terminals on the source side. Terminal rules are constrained to maximum source rule length 7. We use rule count estimation heuristic similar to Hiero. Table 2 shows model sizes for LR-Hiero (GNF), Hiero (SCFG) and GNF grammar with at most 4 non-terminals (GNF-4). Typical Hiero rule extraction excludes phrase-pairs with unaligned words on boundaries (loose phrases). We include loose phrase-pairs as terminal rules in all GNF grammars.

To evaluate our grammar, we use all grammars in LR-Hiero decoder and compare them with SCFG grammar in Hiero decoder. We use a 5-gram LM trained on the Gigaword corpus

Model	Cs-En	De-En	Zh-En
Hiero	20.77	25.72	27.69
LR-Hiero Watanabe et al. (2006)	20.72	25.05	25.99
LR-Hiero+CP	20.52	25.07	26.10
LR-Hiero+CP (GNF-1)	20.38	24.20	25.81
LR-Hiero+CP (GNF-2)	20.49	25.32	25.92
LR-Hiero+CP (GNF-3)	20.50	25.34	26.13
LR-Hiero+CP (GNF-4)	20.50	25.34	26.10

Table 4: BLEU scores for Hiero and LR-Hiero with and without cube pruning (CP). GNF-x: GNF grammars with at most x non-terminals using the proposed rule extraction algorithm.

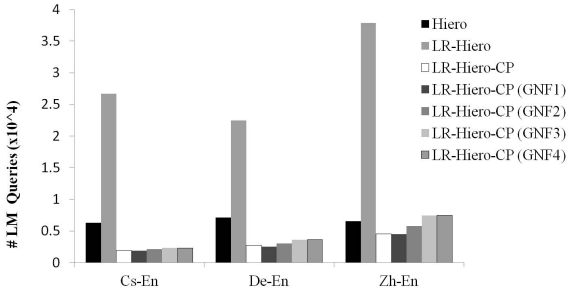


Figure 5: Average number of language model queries. (GNF4) denotes new GNF grammar with 4 non-terminals.

and use KenLM (Heafield, 2011). Pop limit for Hiero and LR-Hiero is 500. To make the results comparable we use the same feature set for all baselines which includes standard features of Hiero: two relative-frequency probabilities $p(e|f)$ and $p(f|e)$, two lexically weighted probabilities $lex(e|f)$ and $lex(f|e)$, a language model probability, word penalty, phrase penalty, and glue rule penalty, and we add distortion features (seperated for regular and glue rules in LR-Hiero) proposed by Siahbani et al. (2013). Weights are tuned by minimizing BLEU loss on the dev set through MERT (Och, 2003) and BLEU scores on test set are reported.

Table 4 shows the BLEU score for different decoders and grammars. The last 4 rows are GNF grammar with 1 to 4 non-terminals extracted by our rule extraction. To show how adding more non-terminals affect the alignment coverage, we translate the devset sentences with different grammars in forced decoding mode. We use CKY decoding for SCFG and LR-decoder for GNF grammars. Table 3 shows the size of the reachable subset by forced decoding for different grammars. It shows that adding more non-terminals considerably improves the alignment coverage on De-En and Zh-En (average 24%).

Comparing Tables 4 and 3 is interesting. While adding rules with more than 2 non-terminals does not change BLEU score it improves the alignment coverage. In our analysis we notice that LR-Decoder rarely uses rules with 3 or 4 non-terminals in K-best list. It is probably because, rules with less non-terminals are generally more frequent and hypotheses which use them have got higher score during decoding. Here we just use Hiero and LR-Hiero standard features which are not designed for rules with more complex reordering. The next step is to elaborate features for rules with 3 and 4 non-terminals[9].

To evaluate the effect of the grammars on decoding process in terms of speed, we use

[9]In another experiment not reported here, we extract rules with unlimited number of non-terminals and source rule length for Cs-En (while we keep non-adjacent non-terminals on the source side). But filtering rules on dev and test sets results in rules with at most 5 non-terminals.

number of language model calls since that directly corresponds to the number of hypotheses considered by the decoder, consequently the speed of decoder. Figure 5 shows the results in terms of average number of language model queries and times in milliseconds on a sample set of 50 sentences from test sets.

5 Related Work

Many approaches have been developed to improve SCFG rules for Hiero. Some of the works have employed generative methods using Bayesian techniques to induce SCFG (Blunsom et al., 2008, 2009; Levenberg et al., 2012; Sankaran et al., 2012a) directly from bilingual data without word alignments. de Gispert et al. (2010) extract rules based on posterior distributions provided by the HMM word-to-word alignment model, rather than a single alignment which is used in original Hiero. Most of these approaches restrict the grammar to rules with one or at most two non-terminals to be able to use the grammar in decoding (Blunsom et al., 2008; de Gispert et al., 2010; Sankaran et al., 2012a).

Recently Levenberg et al. (2012) propose an approach to learn grammars with unrestricted number of non-terminals but do not use the grammar directly in the decoder. The obtained SCFG rules are used to obtain the word alignments rather than the SCFG rules for decoding. Unrestricted number of non-terminals makes the induced grammar unusable in CKY based decoders.

Zhang et al. (2008) encode the word aligned sentence pair as a normalized decomposition tree (a hierarchical representation of all the phrase pairs in linear time, which yields a set of minimal Hiero (SCFG) rules. They discuss that the method can be modified to extract all Hiero rules. But the algorithm is just applied as an analytical tool for aligned bilingual data.

Syntax-based translation systems, tree-to-tree (Ding and Palmer, 2005), tree-to-string (Liu et al., 2006; Huang, 2006) and string-to-tree (Galley et al., 2006), extract sentence level rules, but they extract rules from parse trees (on source or target) rather word aligned sentence pairs which we discussed in this paper.

Braune et al. (2012) extend Hiero by extracting an additional and separate set of rules for long-distance reorderings. They modify Hiero extractor based on some analysis on long-distance German-to-English movement and filtered them based on linguistic information. New rules are applied to long spans (11 to 50) but do not improve translation quality in terms of BLEU (in some case BLEU scores reduce by 0.4). However they show that their approach helps in terms of improving the reordering between source and target (using LRscore (Birch and Osborne, 2011) evaluation scores and some manual evaluation).

6 Conclusion

We propose a dynamic programming algorithm for GNF rule extraction that is linear in the number of GNF rules. We use the sentence level GNF rules with different number of non-terminals in LR-decoder and analyze the effect of these rules in LR-Hiero translation system on different language pairs. New rules with more non-terminals improve the alignment coverage (24% on average) on language pairs with more complex reordering, while it marginally affects the decoding speed. Using rules with more non-terminals is a promising approach in Hiero translation systems which is practical using LR-decoding.

Acknowledgments

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

References

Birch, A. and Osborne, M. (2011). Reordering Metrics for MT. In *Proceedings of the Association for Computational Linguistics*, Portland, Oregon, USA. Association for Computational Linguistics.

Blunsom, P., Cohn, T., Dyer, C., and Osborne, M. (2009). A gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of Association of Computational Linguistics-09*, pages 782–790. Association for Computational Linguistics.

Blunsom, P., Cohn, T., and Osborne, M. (2008). Bayesian synchronous grammar induction. In *Proceedings of Neural Information Processing Systems-08*.

Braune, F., Gojun, A., and Fraser, A. (2012). Long distance reordering during search for hierarchical phrase-based smt. In *Proc. of EAMT 2012*.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL*, pages 263–270.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, 33.

de Gispert, A., Pino, J., and Byrne, W. (2010). Hierarchical phrase-based translation grammars extracted from alignment posterior probabilities. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 545–554. Association for Computational Linguistics.

Ding, Y. and Palmer, M. (2005). Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 541–548, Stroudsburg, PA, USA. Association for Computational Linguistics.

Feng, Y., Liu, Y., Liu, Q., and Cohn, T. (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.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., and Thayer, I. (2006). Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia. Association for Computational Linguistics.

Galley, M. and Manning, C. D. (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. Association for Computational Linguistics.

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

Heafield, K., Hoang, H., Koehn, P., Kiso, T., and Federico, M. (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.

Heafield, K., Koehn, P., and Lavie, A. (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.

Huang, L. (2006). Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA 2006*, pages 66–73.

Huang, L. and Mi, H. (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. Association for Computational Linguistics.

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

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (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*, Proc. of ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proc. of NAACL*.

Levenberg, A., Dyer, C., and Blunsom, P. (2012). A Bayesian Model for Learning SCFGs with Discontiguous Rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 223–232, Jeju Island, Korea. Association for Computational Linguistics.

Liu, Y., Liu, Q., and Lin, S. (2006). Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 609–616, Stroudsburg, PA, USA. Association for Computational Linguistics.

Och, F. J. (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.

Sankaran, B., Haffari, G., and Sarkar, A. (2012a). Compact rule extraction for hierarchical phrase-based translation. In *The 10th biennial conference of the Association for Machine Translation in the Americas (AMTA)*, San Diego, CA. Association for Computational Linguistics.

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

Siahbani, M., Sankaran, B., and Sarkar, A. (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. Association for Computational Linguistics.

Siahbani, M. and Sarkar, A. (2014). Two improvements to left-to-right decoding for hierarchical phrase-based machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar. Association for Computational Linguistics.

Watanabe, T., Tsukada, H., and Isozaki, H. (2006). Left-to-right target generation for hierarchical phrase-based translation. In *Proc. of ACL.*

Zhang, H., Gildea, D., and Chiang, D. (2008). Extracting synchronous grammar rules from word-level alignments in linear time. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pages 1081–1088, Manchester, UK.