# Rule Filtering by Pattern for Efficient Hierarchical Translation

**Gonzalo Iglesias**[⋆] **Adrià de Gispert**[‡] **Eduardo R. Banga**[⋆] **William Byrne**[‡]
[⋆] University of Vigo. Dept. of Signal Processing and Communications. Vigo, Spain
{giglesia,erbanga}@gts.tsc.uvigo.es
[‡] University of Cambridge. Dept. of Engineering. CB2 1PZ Cambridge, U.K.
{ad465,wjb31}@eng.cam.ac.uk

## Abstract

We describe refinements to hierarchical translation search procedures intended to reduce both search errors and memory usage through modifications to hypothesis expansion in cube pruning and reductions in the size of the rule sets used in translation. Rules are put into syntactic classes based on the number of non-terminals and the pattern, and various filtering strategies are then applied to assess the impact on translation speed and quality. Results are reported on the 2008 NIST Arabic-to-English evaluation task.

## 1 Introduction

Hierarchical phrase-based translation (Chiang, 2005) has emerged as one of the dominant current approaches to statistical machine translation. Hiero translation systems incorporate many of the strengths of phrase-based translation systems, such as feature-based translation and strong target language models, while also allowing flexible translation and movement based on hierarchical rules extracted from aligned parallel text. The approach has been widely adopted and reported to be competitive with other large-scale data driven approaches, e.g. (Zollmann et al., 2008).

Large-scale hierarchical SMT involves automatic rule extraction from aligned parallel text, model parameter estimation, and the use of cube pruning k-best list generation in hierarchical translation. The number of hierarchical rules extracted far exceeds the number of phrase translations typically found in aligned text. While this may lead to improved translation quality, there is also the risk of lengthened translation times and increased memory usage, along with possible search errors due to the pruning procedures needed in search.

We describe several techniques to reduce memory usage and search errors in hierarchical translation. Memory usage can be reduced in cube pruning (Chiang, 2007) through smart memoization, and spreading neighborhood exploration can be used to reduce search errors. However, search errors can still remain even when implementing simple phrase-based translation. We describe a 'shallow' search through hierarchical rules which greatly speeds translation without any effect on quality. We then describe techniques to analyze and reduce the set of hierarchical rules. We do this based on the structural properties of rules and develop strategies to identify and remove redundant or harmful rules. We identify groupings of rules based on non-terminals and their patterns and assess the impact on translation quality and computational requirements for each given rule group. We find that with appropriate filtering strategies rule sets can be greatly reduced in size without impact on translation performance.

### 1.1 Related Work

The search and rule pruning techniques described in the following sections add to a growing literature of refinements to the hierarchical phrase-based SMT systems originally described by Chiang (2005; 2007). Subsequent work has addressed improvements and extensions to the search procedure itself, the extraction of the hierarchical rules needed for translation, and has also reported contrastive experiments with other SMT architectures.

*Hiero Search Refinements* Huang and Chiang (2007) offer several refinements to cube pruning to improve translation speed. Venugopal et al. (2007) introduce a Hiero variant with relaxed constraints for hypothesis recombination during parsing; speed and results are comparable to those of cube pruning, as described by Chiang (2007). Li and Khudanpur (2008) report significant improvements in translation speed by taking unseen n-grams into account within cube pruning to minimize language model requests. Dyer et al. (2008)

*Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 380–388,
Athens, Greece, 30 March – 3 April 2009. 

extend the translation of source sentences to translation of input lattices following Chappelier et al. (1999).

*Extensions to Hiero* Blunsom et al. (2008) discuss procedures to combine discriminative latent models with hierarchical SMT. The Syntax-Augmented Machine Translation system (Zollmann and Venugopal, 2006) incorporates target language syntactic constituents in addition to the synchronous grammars used in translation. Shen at al. (2008) make use of target dependency trees and a target dependency language model during decoding. Marton and Resnik (2008) exploit shallow correspondences of hierarchical rules with source syntactic constituents extracted from parallel text, an approach also investigated by Chiang (2005). Zhang and Gildea (2006) propose binarization for synchronous grammars as a means to control search complexity arising from more complex, syntactic, hierarchical rules sets.

*Hierarchical rule extraction* Zhang et al. (2008) describe a linear algorithm, a modified version of shift-reduce, to extract phrase pairs organized into a tree from which hierarchical rules can be directly extracted. Lopez (2007) extracts rules on-the-fly from the training bitext during decoding, searching efficiently for rule patterns using suffix arrays.

*Analysis and Contrastive Experiments* Zollman et al. (2008) compare phrase-based, hierarchical and syntax-augmented decoders for translation of Arabic, Chinese, and Urdu into English, and they find that attempts to expedite translation by simple schemes which discard rules also degrade translation performance. Lopez (2008) explores whether lexical reordering or the phrase discontiguity inherent in hierarchical rules explains improvements over phrase-based systems. Hierarchical translation has also been used to great effect in combination with other translation architectures (e.g. (Sim et al., 2007; Rosti et al., 2007)).

### 1.2 Outline

The paper proceeds as follows. Section 2 describes memoization and spreading neighborhood exploration in cube pruning intended to reduce memory usage and search errors, respectively. A detailed comparison with a simple phrase-based system is presented. Section 3 describes pattern-based rule filtering and various procedures to select rule sets for use in translation with an aim to improving translation quality while minimizing rule set size. Finally, Section 4 concludes.

## 2 Two Refinements in Cube Pruning

Chiang (2007) introduced cube pruning to apply language models in pruning during the generation of k-best translation hypotheses via the application of hierarchical rules in the CYK algorithm. In the implementation of Hiero described here, there is the parser itself, for which we use a variant of the CYK algorithm closely related to CYK+ (Chappelier and Rajman, 1998); it employs hypothesis recombination, without pruning, while maintaining back pointers. Before k-best list generation with cube pruning, we apply a *smart memoization* procedure intended to reduce memory consumption during k-best list expansion. Within the cube pruning algorithm we use *spreading neighborhood exploration* to improve robustness in the face of search errors.

### 2.1 Smart Memoization

Each cell in the chart built by the CYK algorithm contains all possible derivations of a span of the source sentence being translated. After the parsing stage is completed, it is possible to make a very efficient sweep through the backpointers of the CYK grid to count how many times each cell will be accessed by the k-best generation algorithm. When k-best list generation is running, the number of times each cell is visited is logged so that, as each cell is visited for the last time, the k-best list associated with each cell is deleted. This continues until the one k-best list remaining at the top of the chart spans the entire sentence. Memory reductions are substantial for longer sentences: for the longest sentence in the tuning set described later (105 words in length), smart memoization reduces memory usage during the cube pruning stage from 2.1GB to 0.7GB. For average length sentences of approx. 30 words, memory reductions of 30% are typical.

### 2.2 Spreading Neighborhood Exploration

In generation of a k-best list of translations for a source sentence span, every derivation is transformed into a cube containing the possible translations arising from that derivation, along with their translation and language model scores (Chiang, 2007). These derivations may contain non-terminals which must be expanded based on hypotheses generated by lower cells, which them-

| HIERO MJ1 | HIERO | HIERO SHALLOW |
|---|---|---|
| $X \rightarrow \langle V_2V_1, V_1V_2 \rangle$ | $X \rightarrow \langle \gamma, \alpha \rangle$ | $X \rightarrow \langle \gamma_s, \alpha_s \rangle$ |
| $X \rightarrow \langle V, V \rangle$ | $\gamma, \alpha \in (\{X\} \cup \mathbf{T})^+$ | $X \rightarrow \langle V, V \rangle$ |
| $V \rightarrow \langle s, t \rangle$ | | $V \rightarrow \langle s, t \rangle$ |
| $s, t \in \mathbf{T}^+$ | | $s, t \in \mathbf{T}^+; \gamma_s, \alpha_s \in (\{V\} \cup \mathbf{T})^+$ |

Table 1: Hierarchical grammars (not including glue rules). **T** is the set of terminals.

selves may contain non-terminals. For efficiency each cube maintains a queue of hypotheses, called here the *frontier queue*, ranked by translation and language model score; it is from these frontier queues that hypotheses are removed to create the k-best list for each cell. When a hypothesis is extracted from a frontier queue, that queue is updated by searching through the neighborhood of the extracted item to find novel hypotheses to add; if no novel hypotheses are found, that queue necessarily shrinks. This shrinkage can lead to search errors. We therefore require that, when a hypothesis is removed, new candidates must be added by exploring a neighborhood which spreads from the last extracted hypothesis. Each axis of the cube is searched (here, to a depth of 20) until a novel hypothesis is found. In this way, up to three new candidates are added for each entry extracted from a frontier queue.

Chiang (2007) describes an initialization procedure in which these frontier queues are seeded with a single candidate per axis; we initialize each frontier queue to a depth of $b^{N_{nt}+1}$, where $N_{nt}$ is the number of non-terminals in the derivation and $b$ is a search parameter set throughout to 10. By starting with deep frontier queues and by forcing them to grow during search we attempt to avoid search errors by ensuring that the universe of items within the frontier queues does not decrease as the k-best lists are filled.

### 2.3 A Study of Hiero Search Errors in Phrase-Based Translation

Experiments reported in this paper are based on the NIST MT08 Arabic-to-English translation task. Alignments are generated over all allowed parallel data, ($\sim$150M words per language). Features extracted from the alignments and used in translation are in common use: target language model, source-to-target and target-to-source phrase translation models, word and rule penalties, number of usages of the glue rule, source-to-target and target-to-source lexical models, and three rule

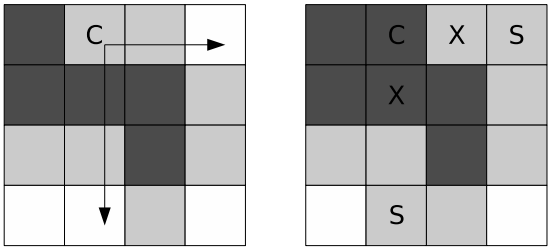


Figure 1: Spreading neighborhood exploration within a cube, just before and after extraction of the item C. Grey squares represent the frontier queue; black squares are candidates already extracted. Chiang (2007) would only consider adding items X to the frontier queue, so the queue would shrink. Spreading neighborhood exploration adds candidates S to the frontier queue.

count features inspired by Bender et al. (2007). MET (Och, 2003) iterative parameter estimation under IBM BLEU is performed on the development set. The English language used model is a 4-gram estimated over the parallel text and a 965 million word subset of monolingual data from the English Gigaword Third Edition. In addition to the MT08 set itself, we use a development set *mt02-05-tune* formed from the odd numbered sentences of the NIST MT02 through MT05 evaluation sets; the even numbered sentences form the validation set *mt02-05-test*. The *mt02-05-tune* set has 2,075 sentences.

We first compare the cube pruning decoder to the TTM (Kumar et al., 2006), a phrase-based SMT system implemented with Weighted Finite-State Tansducers (Allauzen et al., 2007). The system implements either a monotone phrase order translation, or an MJ1 (maximum phrase jump of 1) reordering model (Kumar and Byrne, 2005). Relative to the complex movement and translation allowed by Hiero and other models, MJ1 is clearly inferior (Dreyer et al., 2007); MJ1 was developed with efficiency in mind so as to run with a minimum of search errors in translation and to be easily and exactly realized via WFSTs. Even for the

large models used in an evaluation task, the TTM system is reported to run largely without pruning (Blackwood et al., 2008).

The Hiero decoder can easily be made to implement MJ1 reordering by allowing only a restricted set of reordering rules in addition to the usual glue rule, as shown in left-hand column of Table 1, where **T** is the set of terminals. Constraining Hiero in this way makes it possible to compare its performance to the exact WFST TTM implementation and to identify any search errors made by Hiero.

Table 2 shows the lowercased IBM BLEU scores obtained by the systems for *mt02-05-tune* with monotone and reordered search, and with MET-optimised parameters for MJ1 reordering. For Hiero, an N-best list depth of 10,000 is used throughout. In the monotone case, all phrase-based systems perform similarly although Hiero does make search errors. For simple MJ1 reordering, the basic Hiero search procedure makes many search errors and these lead to degradations in BLEU. Spreading neighborhood expansion reduces the search errors and improves BLEU score significantly but search errors remain a problem. Search errors are even more apparent after MET. This is not surprising, given that *mt02-05-tune* is the set over which MET is run: MET drives up the likelihood of good hypotheses at the expense of poor hypotheses, but search errors often increase due to the expanded dynamic range of the hypothesis scores.

Our aim in these experiments was to demonstrate that spreading neighborhood exploration can aid in avoiding search errors. We emphasize that we are not proposing that Hiero should be used to implement reordering models such as MJ1 which were created for completely different search procedures (e.g. WFST composition). However these experiments do suggest that search errors may be an issue, particularly as the search space grows to include the complex long-range movement allowed by the hierarchical rules. We next study various filtering procedures to reduce hierarchical rule sets to find a balance between translation speed, memory usage, and performance.

| | Monotone | | MJ1 | | MJ1+MET | |
|---|---|---|---|---|---|---|
| | BLEU | SE | BLEU | SE | BLEU | SE |
| a | 44.7 | - | 47.2 | - | 49.1 | - |
| b | 44.5 | 342 | 46.7 | 555 | 48.4 | 822 |
| c | 44.7 | 77 | 47.1 | 191 | 48.9 | 360 |

Table 2: Phrase-based TTM and Hiero performance on *mt02-05-tune* for TTM (a), Hiero (b), Hiero with spreading neighborhood exploration (c). SE is the number of Hiero hypotheses with search errors.

## 3 Rule Filtering by Pattern

Hierarchical rules $X \rightarrow \langle\gamma,\alpha\rangle$ are composed of sequences of terminals and non-terminals, which we call *elements*. In the source, a maximum of two non-adjacent non-terminals is allowed (Chiang, 2007). Leaving aside rules without non-terminals (i.e. phrase pairs as used in phrase-based translation), rules can be classed by their number of non-terminals, $\mathrm{N}_{nt}$, and their number of elements, $\mathrm{N}_e$. There are 5 possible classes: $\mathrm{N}_{nt}.\mathrm{N}_e = 1.2, 1.3, 2.3, 2.4, 2.5$.

During rule extraction we search each class separately to control memory usage. Furthermore, we extract from alignments only those rules which are relevant to our given test set; for computation of backward translation probabilities we log general counts of target-side rules but discard unneeded rules. Even with this restriction, our initial ruleset for *mt02-05-tune* exceeds 175M rules, of which only 0.62M are simple phrase pairs.

The question is whether all these rules are needed for translation. If the rule set can be reduced without reducing translation quality, both memory efficiency and translation speed can be increased. Previously published approaches to reducing the rule set include: enforcing a minimum span of two words per non-terminal (Lopez, 2008), which would reduce our set to 115M rules; or a minimum count (mincount) threshold (Zollmann et al., 2008), which would reduce our set to 78M (mincount=2) or 57M (mincount=3) rules. Shen et al. (2008) describe the result of filtering rules by insisting that target-side rules are well-formed dependency trees. This reduces their rule set from 140M to 26M rules. This filtering leads to a degradation in translation performance (see Table 2 of Shen et al. (2008)), which they counter by adding a dependency LM in translation. As another reference point, Chiang (2007) reports Chinese-to-English translation experiments based on 5.5M rules.

Zollmann et al. (2008) report that filtering rules

en masse leads to degradation in translation performance. Rather than apply a coarse filtering, such as a mincount for all rules, we follow a more syntactic approach and further classify our rules according to their *pattern* and apply different filters to each pattern depending on its value in translation. The premise is that some patterns are more important than others.

### 3.1 Rule Patterns

| **Class** $N_{nt}.N_e$ | **Rule Pattern** ⟨**source** , **target**⟩ | **Types** |
|---|---|---|
| 1.2 | ⟨wX$_1$ , wX$_1$⟩ | 1185028 |
| | ⟨wX$_1$ , wX$_1$w⟩ | 153130 |
| | ⟨wX$_1$ , X$_1$w⟩ | 97889 |
| 1.3 | ⟨wX$_1$w , wX$_1$w⟩ | 32903522 |
| | ⟨wX$_1$w , wX$_1$⟩ | 989540 |
| 2.3 | ⟨X$_1$wX$_2$ , X$_1$wX$_2$⟩ | 1554656 |
| | ⟨X$_2$wX$_1$ , X$_1$wX$_2$⟩ | 39163 |
| 2.4 | ⟨wX$_1$wX$_2$ , wX$_1$wX$_2$⟩ | 26901823 |
| | ⟨X$_1$wX$_2$w , X$_1$wX$_2$w⟩ | 26053969 |
| | ⟨wX$_1$wX$_2$ , wX$_1$wX$_2$w⟩ | 2534510 |
| | ⟨wX$_2$wX$_1$ , wX$_1$wX$_2$⟩ | 349176 |
| | ⟨X$_2$wX$_1$w , X$_1$wX$_2$w⟩ | 259459 |
| 2.5 | ⟨wX$_1$wX$_2$w , wX$_1$wX$_2$w⟩ | 61704299 |
| | ⟨wX$_1$wX$_2$w , wX$_1$X$_2$w⟩ | 3149516 |
| | ⟨wX$_1$wX$_2$w , X$_1$wX$_2$w⟩ | 2330797 |
| | ⟨wX$_2$wX$_1$w , wX$_1$wX$_2$w⟩ | 275810 |
| | ⟨wX$_2$wX$_1$w , wX$_1$X$_2$w⟩ | 205801 |

Table 3: Hierarchical rule patterns classed by number of non-terminals, $N_{nt}$, number of elements $N_e$, source and target patterns, and types in the rule set extracted for *mt02-05-tune*.

Given a rule set, we define *source patterns* and *target patterns* by replacing every sequence of non-terminals by a single symbol 'w' (indicating word, i.e. terminal string, $w \in \mathbf{T}^+$). Each hierarchical rule has a unique source and target pattern which together define the *rule pattern*.

By ignoring the identity and the number of adjacent terminals, the rule pattern represents a natural generalization of any rule, capturing its structure and the type of reordering it encodes. In total, there are 66 possible rule patterns. Table 3 presents a few examples extracted for *mt02-05-tune*, showing that some patterns are much more diverse than others. For example, patterns with two non-terminals ($N_{nt}$=2) are richer than patterns with $N_{nt}$=1, as they cover many more distinct rules. Additionally, patterns with two non-terminals which also have a monotonic relationship between source and target non-terminals are much more diverse than their reordered counterparts.

Some examples of extracted rules and their corresponding pattern follow, where Arabic is shown in Buckwalter encoding.

Pattern ⟨wX$_1$ , wX$_1$w⟩ :
⟨w+ qAl X$_1$ , the X$_1$said⟩
Pattern ⟨wX$_1$w , wX$_1$⟩ :
⟨fy X$_1$kAnwn Al>wl , on december X$_1$⟩
Pattern ⟨wX$_1$wX$_2$ , wX$_1$wX$_2$w⟩ :
⟨Hl X$_1$lAzmp X$_2$ , a X$_1$solution to the X$_2$crisis⟩

### 3.2 Building an Initial Rule Set

We describe a greedy approach to building a rule set in which rules belonging to a pattern are added to the rule set guided by the improvements they yield on *mt02-05-tune* relative to the monotone Hiero system described in the previous section. We find that certain patterns seem not to contribute to any improvement. This is particularly significant as these patterns often encompass large numbers of rules, as with patterns with matching source and target patterns. For instance, we found no improvement when adding the pattern ⟨X$_1$w,X$_1$w⟩, of which there were 1.2M instances (Table 3). Since concatenation is already possible under the general glue rule, rules with this pattern are redundant. By contrast, the much less frequent reordered counterpart, i.e. the ⟨wX$_1$,X$_1$w⟩ pattern (0.01M instances), provides substantial gains. The situation is analogous for rules with two non-terminals ($N_{nt}$=2).

Based on exploratory analyses (not reported here, for space) an initial rule set was built by excluding patterns reported in Table 4. In total, 171.5M rules are excluded, for a remaining set of 4.2M rules, 3.5M of which are hierarchical. We acknowledge that adding rules in this way, by greedy search, is less than ideal and inevitably raises questions with respect to generality and repeatability. However in our experience this is a robust approach, mainly because the initial translation system runs very fast; it is possible to run many exploratory experiments in a short time.

| | Excluded Rules | Types |
|---|---|---|
| a | $\langle X_1 w, X_1 w \rangle$ , $\langle w X_1, w X_1 \rangle$ | 2332604 |
| b | $\langle X_1 w X_2, * \rangle$ | 2121594 |
| c | $\langle X_1 w X_2 w, X_1 w X_2 w \rangle$ , $\langle w X_1 w X_2, w X_1 w X_2 \rangle$ | 52955792 |
| d | $\langle w X_1 w X_2 w, * \rangle$ | 69437146 |
| e | $N_{nt}.N_e$= 1.3 w mincount=5 | 32394578 |
| f | $N_{nt}.N_e$= 2.3 w mincount=5 | 166969 |
| g | $N_{nt}.N_e$= 2.4 w mincount=10 | 11465410 |
| h | $N_{nt}.N_e$= 2.5 w mincount=5 | 688804 |

Table 4: Rules excluded from the initial rule set.

### 3.3 Shallow versus Fully Hierarchical Translation

In measuring the effectiveness of rules in translation, we also investigate whether a 'fully hierarchical' search is needed or whether a shallow search is also effective. In constrast to full Hiero, in the shallow search, only phrases are allowed to be substituted into non-terminals. The rules used in each case can be expressed as shown in the 2nd and 3rd columns of Table 1. Shallow search can be considered (loosely) to be a form of rule filtering.

As can be seen in Table 5 there is no impact on BLEU, while translation speed increases by a factor of 7. Of course, these results are specific to this Arabic-to-English translation task, and need not be expected to carry over to other language pairs, such as Chinese-to-English translation. However, the impact of this search simplification is easy to measure, and the gains can be significant enough, that it may be worth investigation even for languages with complex long distance movement.

| *mt02-05-* | *-tune* | | *-test* |
|---|---|---|---|
| **System** | **Time** | **BLEU** | **BLEU** |
| HIERO | 14.0 | 52.1 | 51.5 |
| HIERO - shallow | 2.0 | 52.1 | 51.4 |

Table 5: Translation performance and time (in seconds per word) for full vs. shallow Hiero.

### 3.4 Individual Rule Filters

We now filter rules individually (not by class) according to their number of translations. For each fixed $\gamma \notin \mathbf{T}^+$ (i.e. with at least 1 non-terminal), we define the following filters over rules $X \rightarrow \langle \gamma, \alpha \rangle$:

- **Number of translations (NT)**. We keep the **NT** most frequent $\alpha$, i.e. each $\gamma$ is allowed to have at most **NT** rules.
- **Number of reordered translations (NRT)**. We keep the **NRT** most frequent $\alpha$ with monotonic non-terminals and the **NRT** most frequent $\alpha$ with reordered non-terminals.
- **Count percentage (CP)**. We keep the most frequent $\alpha$ until their aggregated number of counts reaches a certain percentage **CP** of the total counts of $X \rightarrow \langle \gamma, * \rangle$. Some $\gamma$'s are allowed to have more $\alpha$'s than others, depending on their count distribution.

Results applying these filters with various thresholds are given in Table 6, including number of rules and decoding time. As shown, all filters achieve at least a 50% speed-up in decoding time by discarding 15% to 25% of the baseline rules. Remarkably, performance is unaffected when applying the simple **NT** and **NRT** filters with a threshold of 20 translations. Finally, the **CM** filter behaves slightly worse for thresholds of 90% for the same decoding time. For this reason, we select **NRT**=20 as our general filter.

| *mt02-05-* | *-tune* | | | *-test* |
|---|---|---|---|---|
| **Filter** | **Time** | **Rules** | **BLEU** | **BLEU** |
| baseline | 2.0 | 4.20 | 52.1 | 51.4 |
| **NT**=10 | 0.8 | 3.25 | 52.0 | 51.3 |
| **NT**=15 | 0.8 | 3.43 | 52.0 | 51.3 |
| **NT**=20 | 0.8 | 3.56 | 52.1 | 51.4 |
| **NRT**=10 | 0.9 | 3.29 | 52.0 | 51.3 |
| **NRT**=15 | 1.0 | 3.48 | 52.0 | 51.4 |
| **NRT**=20 | 1.0 | 3.59 | 52.1 | 51.4 |
| **CP**=50 | 0.7 | 2.56 | 51.4 | 50.9 |
| **CP**=90 | 1.0 | 3.60 | 52.0 | 51.3 |

Table 6: Impact of general rule filters on translation (IBM BLEU), time (in seconds per word) and number of rules (in millions).

### 3.5 Pattern-based Rule Filters

In this section we first reconsider whether reintroducing the monotonic rules (originally excluded as described in rows 'b', 'c', 'd' in Table 4) affects performance. Results are given in the upper rows of Table 7. For all classes, we find that reintroducing these rules increases the total number of rules

| *mt02-05-* | | *-tune* | | | *-test* |
|---|---|---|---|---|---|
| $\mathbf{N}_{nt}.\mathbf{N}_e$ | Filter | **Time** | **Rules** | **BLEU** | **BLEU** |
| baseline **NRT**=20 | | 1.0 | 3.59 | 52.1 | 51.4 |
| 2.3 | +monotone | 1.1 | 4.08 | 51.5 | 51.1 |
| 2.4 | +monotone | 2.0 | 11.52 | 51.6 | 51.0 |
| 2.5 | +monotone | 1.8 | 6.66 | 51.7 | 51.2 |
| 1.3 | mincount=3 | 1.0 | 5.61 | 52.1 | 51.3 |
| 2.3 | mincount=1 | 1.2 | 3.70 | 52.1 | 51.4 |
| 2.4 | mincount=5 | 1.8 | 4.62 | 52.0 | 51.3 |
| 2.4 | mincount=15 | 1.0 | 3.37 | 52.0 | 51.4 |
| 2.5 | mincount=1 | 1.1 | 4.27 | 52.2 | 51.5 |
| 1.2 | mincount=5 | 1.0 | 3.51 | 51.8 | 51.3 |
| 1.2 | mincount=10 | 1.0 | 3.50 | 51.7 | 51.2 |

Table 7: Effect of pattern-based rule filters. Time in seconds per word. Rules in millions.

substantially, despite the NRT=20 filter, but leads to degradation in translation performance.

We next reconsider the mincount threshold values for $\mathrm{N}_{nt}.\mathrm{N}_e$ classes 1.3, 2.3, 2.4 and 2.5 originally described in Table 4 (rows 'e' to 'h'). Results under various mincount cutoffs for each class are given in Table 7 (middle five rows). For classes 2.3 and 2.5, the mincount cutoff can be reduced to 1 (i.e. all rules are kept) with slight translation improvements. In contrast, reducing the cutoff for classes 1.3 and 2.4 to 3 and 5, respectively, adds many more rules with no increase in performance. We also find that increasing the cutoff to 15 for class 2.4 yields the same results with a smaller rule set. Finally, we consider further filtering applied to class 1.2 with mincount 5 and 10 (final two rows in Table 7). The number of rules is largely unchanged, but translation performance drops consistently as more rules are removed.

Based on these experiments, we conclude that it is better to apply separate mincount thresholds to the classes to obtain optimal performance with a minimum size rule set.

### 3.6 Large Language Models and Evaluation

Finally, in this section we report results of our shallow hierarchical system with the 2.5 mincount=1 configuration from Table 7, after including the following N-best list rescoring steps.

- *Large-LM rescoring*. We build sentence-specific zero-cutoff stupid-backoff (Brants et al., 2007) 5-gram language models, estimated using $\sim$4.7B words of English newswire text, and apply them to rescore each 10000-best list.
- *Minimum Bayes Risk (MBR)*. We then rescore the first 1000-best hypotheses with MBR, taking the negative sentence level BLEU score as the loss function to minimise (Kumar and Byrne, 2004).

Table 8 shows results for *mt02-05-tune*, *mt02-05-test*, the NIST subsets from the MT06 evaluation (*mt06-nist-nw* for newswire data and *mt06-nist-ng* for newsgroup) and *mt08*, as measured by lowercased IBM BLEU and TER (Snover et al., 2006). Mixed case NIST BLEU for this system on *mt08* is 42.5. This is directly comparable to official MT08 evaluation results[1].

## 4 Conclusions

This paper focuses on efficient large-scale hierarchical translation while maintaining good translation quality. Smart memoization and spreading neighborhood exploration during cube pruning are described and shown to reduce memory consumption and Hiero search errors using a simple phrase-based system as a contrast.

We then define a general classification of hierarchical rules, based on their number of non-terminals, elements and their patterns, for refined extraction and filtering.

For a large-scale Arabic-to-English task, we show that shallow hierarchical decoding is as good

[1]Full MT08 results are available at http://www.nist.gov/speech/tests/mt/2008/. It is worth noting that many of the top entries make use of system combination; the results reported here are for single system translation.

| | *mt02-05-tune* | *mt02-05-test* | *mt06-nist-nw* | *mt06-nist-ng* | *mt08* |
|---|---|---|---|---|---|
| HIERO+MET | 52.2 / 41.6 | 51.5 / 42.2 | 48.4 / 43.6 | 35.3 / 53.2 | 42.5 / 48.6 |
| +rescoring | 53.2 / 40.8 | 52.6 / 41.4 | 49.4 / 42.9 | 36.6 / 53.5 | 43.4 / 48.1 |

Table 8: Arabic-to-English translation results (lower-cased IBM BLEU / TER) with large language models and MBR decoding.

as fully hierarchical search and that decoding time is dramatically decreased. In addition, we describe individual rule filters based on the distribution of translations with further time reductions at no cost in translation scores. This is in direct contrast to recent reported results in which other filtering strategies lead to degraded performance (Shen et al., 2008; Zollmann et al., 2008).

We find that certain patterns are of much greater value in translation than others and that separate minimum count filters should be applied accordingly. Some patterns were found to be redundant or harmful, in particular those with two monotonic non-terminals. Moreover, we show that the value of a pattern is not directly related to the number of rules it encompasses, which can lead to discarding large numbers of rules as well as to dramatic speed improvements.

Although reported experiments are only for Arabic-to-English translation, we believe the approach will prove to be general. Pattern relevance will vary for other language pairs, but we expect filtering strategies to be equally worth pursuing.

## Acknowledgments

This work was supported in part by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011- 06-C-0022. G. Iglesias supported by Spanish Government research grant BES-2007-15956 (project TEC2006-13694-C03-03).

## References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of CIAA*, pages 11–23.

Oliver Bender, Evgeny Matusov, Stefan Hahn, Sasa Hasan, Shahram Khadivi, and Hermann Ney. 2007. The RWTH Arabic-to-English spoken language translation system. In *Proceedings of ASRU*, pages 396–401.

Graeme Blackwood, Adrià de Gispert, Jamie Brunning, and William Byrne. 2008. Large-scale statistical machine translation with weighted finite state transducers. In *Proceedings of FSMNLP*, pages 27–35.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-HLT*, pages 200–208.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-ACL*, pages 858–867.

Jean-Cédric Chappelier and Martin Rajman. 1998. A generalized CYK algorithm for parsing stochastic CFG. In *Proceedings of TAPD*, pages 133–137.

Jean-Cédric Chappelier, Martin Rajman, Ramón Aragüés, and Antoine Rozenknop. 1999. Lattice parsing for speech recognition. In *Proceedings of TALN*, pages 95–104.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, pages 263–270.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Markus Dreyer, Keith Hall, and Sanjeev Khudanpur. 2007. Comparing reordering constraints for SMT using efficient BLEU oracle computation. In *Proceedings of SSST, NAACL-HLT 2007 / AMTA Workshop on Syntax and Structure in Statistical Translation*.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL-HLT*, pages 1012–1020.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of HLT-NAACL*, pages 169–176.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of HLT-EMNLP*, pages 161–168.

Shankar Kumar, Yonggang Deng, and William Byrne. 2006. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 12(1):35–75.

Zhifei Li and Sanjeev Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the ACL-HLT Second Workshop on Syntax and Structure in Statistical Translation*, pages 10–18.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of EMNLP-CONLL*, pages 976–985.

Adam Lopez. 2008. Tera-scale translation models via pattern matching. In *Proceedings of COLING*, pages 505–512.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-HLT*, pages 1003–1011.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007. Combining outputs from multiple machine translation systems. In *Proceedings of HLT-NAACL*, pages 228–235.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-HLT*, pages 577–585.

Khe Chai Sim, William Byrne, Mark Gales, Hichem Sahbi, and Phil Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *Proceedings of ICASSP*, volume 4, pages 105–108.

Matthew Snover, Bonnie J. Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, pages 223–231.

Ashish Venugopal, Andreas Zollmann, and Vogel Stephan. 2007. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *Proceedings of HLT-NAACL*, pages 500–507.

Hao Zhang and Daniel Gildea. 2006. Synchronous binarization for machine translation. In *Proceedings of HLT-NAACL*, pages 256–263.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting synchronous grammar rules from word-level alignments in linear time. In *Proceedings of COLING*, pages 1081–1088.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of NAACL Workshop on Statistical Machine Translation*, pages 138–141.

Andreas Zollmann, Ashish Venugopal, Franz Och, and Jay Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of COLING*, pages 1145–1152.