RC24910 (W0912-014) December 2, 2009
Computer Science

IBM Research Report

A Categorization of Hiero Reordering Rules

Bing Zhao
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

A Categorization of Hiero Reordering Rules

Bing Zhao
IBM T.J. Watson Research
zhaob@us.ibm.com

Abstract

We propose to classify the Hiero grammar into refined subcategories, each of which is characterized with a specific reordering pattern. The subcategories enable us to encourage more of certain reordering patterns across a language-pair with a proper mixture-weight, to form a better grammar representation for machine translation. Specifically, Hiero grammar is classified according to the positions and the relative orders of the nonterminals (variables) and lexical items: variables with lexical items defining their boundaries, together with the monotone/nonmonotone alignment nature. We propose to learn the mixture-weight in two simple ways, relying on minimum error rate training and maximum likelihood training, respectively. Improved translation results were obtained from a state-of-the-art Arabic-English Hiero system.

1 Introduction

Hiero grammar (Chiang, 2007) has been widely applied in current translation systems, and achieved state-of-the-art performances in many evaluations, due to its ability to capture rich reordering patterns across language pairs. However, when we are doing general translation tasks, we have to handle hundreds of millions of Hiero plausible reordering rules. For instance, the number of *unique* Hiero rules relevant to one single 24-token source-sentence is 7,609 (already pruned with a frequency cut of 5). Such a size of grammar incurs a huge overhead cost for developing machine translation engines, and significantly slows down the decoding process. Worse is, many Hiero rules generate much more spurious ambiguities during the decoding, which often yield undesired sub-optimal translations and consequently confuse automatic optimization processes for translation.

A closer check of the grammar reveals the rewriting rules often have different *values* of reordering patterns for decoding. For instance, rules with monotone reordering such as "$a\ X \leftrightarrow A\ X$"[1] have generally less value in improving translation quality than "$a\ X \leftrightarrow A_1\ X\ A_2$", which introduces a "fork" alignment on the target side. The above rules, including a variable X without lexical boundaries, often generate significant spurious ambiguities during the decoding. These problems motivated us to re-examine the Hiero rule in hope of pinpointing the potential useful Hiero rules from the vast number of plausible reordering rules extracted from parallel data. Our anticipated solution should remove the redundancy from the grammar. As a result of smaller grammar size, Hiero rules will have sharper frequency distributions, which can lead to potential better translations. Also, the categorization of grammar allows one to encourage certain reordering patterns more by applying mixture-weights to such categories. For language-pairs, such as Arabic-English, we expect to have relative more swaps than monotones for certain type of Hiero rules, and for Spanish-English, we expect to use more monotone ones. Overall, a detailed classification of the reordering patterns enables us to have more control and flexibilities in configuring the Hiero grammar for better translations.

The reset of the paper is structured as follows: in section 2, the Hiero reordering rules are sorted into subcategories according to the positions of the nonterminals, together with the reordering patterns the rule encodes; in section 3, we describe two ways to learn the mixture weights for combining the categorized Hiero rules; in section 4, detailed experiments are reported on the applications of the proposed mixtures of Hiero rules.

2 Classify Hiero Reordering Rules

Word reordering patterns are captured in the relative positions of terminals (lexical items) and nonterminals (variables) in Hiero rule context. Depending on the positions in the context, the ambiguities in rewriting the nonterminals typically vary even for nonterminals in the same rule during the decoding. Generally speaking, the rule like "$X\ a \leftrightarrow A\ X$" generates a long-range reordering for the lexical item "a", with relatively low costs. Unfortunately, the reordering is usually wrong in our observations.

To reduce such ambiguities generated by a Hiero rule, people tried length distributions for each nonterminal in each Hiero rule. The length distributions chosen was simple gaussian as in Shen et al. (2009). However, due to the cap of the underlying phrase-pairs' length[2], the mean and

[1] Notations used for a Hiero rule in this paper: the lefthand side is the source string, and the righthand side is target string; the lowercased letters in the source are aligned to the uppercased ones in the target.

[2] In our system, the maximum length for a block (phrase pair) is 12-tokens on the source side, and the mean is 3.5 tokens, and variance is 3.9 tokens for rewriting a nonterminal X, using Arabic-English MaxEnt-Aligned (Ittycheriah and Roukos, 2005) parallel data.

variance are far away [3] from the expectation in real decoding time. Manipulations and tuning have to be applied to accommodate the expected ranges for rewriting a nonterminal during the decoding time.

In this paper, we characterize the nonterminals in the Hiero rule context, by examining if the nonterminals have lexical items to define their boundaries on the source side. We also combine the categories with the rules' alignment, by evaluating the relative positions among terminals and nonterminals: *monotone* versus *nonmonotone*, to refine the clusters further. In particular, we classify the following four basic categories for Hiero grammar.

2.1 LO:Left-side is open

In a Hiero rule, a nonterminal is at the *left-most* side of the source phrase, or at the very beginning of the source side. The other variable (if there is one) is bounded with lexical items. Examples are shown as in Eqn. 1 and Eqn. 2.

$$X_1\ a\ X_2\ b \quad \leftrightarrow \quad X_1\ A\ X_2\ B, \tag{1}$$

$$X_1\ a\ X_2\ b \quad \leftrightarrow \quad X_2\ A\ X_1\ B. \tag{2}$$

Nonterminal "X_1" has no lexical item to define its left boundary, and it can be instantiated to cover any source tokens to the left side of the matched point of 'a' in a test sentence during the decoding. We say the left-side is open for X_1. In a way, this rule gives more reordering freedom than necessary in practice.

2.2 RO:Right-side is open

The Hiero rule contains one nonterminal at the rightmost of the source phrase, while the other variable (if any) is bounded with lexical items. Similar to the above description in section 2.1, there is no lexical item to bound the instantiation of the nonterminal X_2 in Eqn. 3 and Eqn. 4. These rules are at risk of being misused during the decoding time, and generate undesired reorderings especially for Eqn. 4.

$$a\ X_1\ b\ X_2 \leftrightarrow X_1\ A\ X_2\ B, \tag{3}$$

$$a\ X_1\ b\ X_2 \leftrightarrow X_2\ A\ X_1\ B. \tag{4}$$

2.3 BO:Both Sides are open

In this case, a Hiero rule has two variables, and they are at the rightmost and leftmost source positions, respectively. This is what we called "both sides are open". During the decoding time, once the middle lexical items are matched, this rule will fire for all the rest source tokens to the left and right, and generate more reorderings than necessary. Shown in Eqn. 5, there is one lexical item in the middle, and X_1 and X_2 are both unbounded for instantiations in the decoding process.

$$X_1\ b\ X_2 \leftrightarrow X_1\ B\ X_2, \tag{5}$$

$$X_1\ b\ X_2 \leftrightarrow X_2\ B\ X_1. \tag{6}$$

[3] Gaussian eliminates any points two variances away from the mean.

2.4 NO:Non-Open

As shown in Eqn. 7 and Eqn. 8, all the nonterminals are bounded by the source side lexical items in the context, a and c. The freedom to rewrite these nonterminals is much less than any of the other three cases. The lexical items here introduce more informative evidences for word reorderings; such rules are applied only when the context is matched exactly.

$$a\ X_1\ b\ X_2\ c \leftrightarrow A\ X_1\ B\ X_2\ C, \tag{7}$$

$$a\ X_1\ b\ X_2\ c \leftrightarrow A\ X_2\ B\ X_1\ C. \tag{8}$$

Besides the four nonoverlap categories, on the positions of the source nonterminals in the Hiero rule context, we can also have two additional useful labels for each Hiero rule, considering the reordering nature: monotone and nonmonotone, as detailed in section 2.5.

2.5 Additional labels: monotone v.s. nonmonotone

Rules, by which the relative positions for the nonterminals and terminals in the source is kept the same on the target side, are labeled as monotone. For instance, rules in Eqn. 5 (and in Eqn. 1, Eqn. 3, Eqn. 7) are all monotone in nature. Because the relative positions for nonterminals and lexical items do not move, we can infer the same translations using simpler grammar such as the glue rule $X_1X_2 \leftrightarrow X_1X_2$ as in ITG grammar (Wu, 1997).

On the other hand, when there are changes of the relative positions for terminals and nonterminals, we consider them nonmonotone (or *swap*). Empirically, we have observed wild and long jumps of reorderings introduced by such rules: it can move chunk/lexical-item too far away, as shown in Figure 1-(a) for Arabic word "tErD" (was exposed to), with a jump of more than 23 source tokens. There are *at least* three different cases about this reordering pattern. The first case is that the positions for the nonterminals are swapped only, such as the rule in Eqn. 6. the second case is that the positions for the lexical item are changed, such as the rule in Eqn. 10; the third case is the lexical item "a" has a one-to-many alignment to "A_1" and "A_2" as in Eqn. 11.

$$X_1\ a\ X_2 \leftrightarrow X_2\ A\ X_1, \tag{9}$$

$$X_1\ a\ X_2 \leftrightarrow X_1\ X_2\ A, \tag{10}$$

$$a\ X_1 \leftrightarrow A_1\ X_1\ A_2. \tag{11}$$

Overall, such rules result in reorderings among the nonterminals and lexical items, and such reorderings frequently involve non-desirable long jumps.

3 Learning Mixtures of Categorized Rules

We can assign two kinds of cost functions to combine the mixtures of the categorized Hiero grammar, via weights learned from typical minimum error rate training.

3.1 Binary Feature-Functions Vector

First, we associate a binary feature vector for a hiero rule, with a dimension for each subcategories. The total dimension of the binary feature vector for a Hiero rule is six, with four for basic nonoverlap subcategories, and two additional labels for the monotone/nonmontone reorderings. The binary feature functions will be the accumulated as counters of how many each type of rules are applied in constructing a translation hypothesis.

3.2 Maximum Likelihood $p(\textbf{type})$ via Chart-Parsing

Secondly, we can infer a likelihood function $p(\text{type})$, by taking the reference translation and run force decoding with a given categorized Hiero grammar. In this way, we can compute, from the derivation forest, the inside-outside fractional counts for each type of the Hiero rules (Huang and Zhou, 2009). With these fractional counts, we then normalize and get a maximum likelihood estimation for applying a particular type of rules for decoding unseen sentences. The total number of feature functions is simply one for $p(\text{type})$, where the type space is the cross product of four basic categories with two additional labels for the monotone/nonmontone reorderings.

4 Experiments

Our experiments were carried out on the MT08 Arabic-English newswire data. We illustrated the impacts of the proposed categories for Hiero grammar, by testing the individual feature functions' strength, and the combinatorial effects via the mixtures.

A chart-based decoder (Zhao and Al-Onaizan, 2008), using Hiero grammar, was applied, and the parameters were tuned on the MT06 NIST part via a Simplex-downhill algorithm on optimizing toward (TER-BLEU). The feature functions in the baseline system, included a 5-gram language model, relative frequencies and IBM Model-1 scores for both blocks and Hiero rules, word count, and phrase count. On top of it, one or six additional feature functions on the categories we described above in section 3, were integrated. Our translation models (blocks and Hiero rules) were learned from word-aligned (Ittycheriah and Roukos, 2005) subsampled parallel corpora.

4.1 On Basic Hiero Rules Categorizations

Table 1 shows that, the losses of BLEU if removing an individual type of the Hiero rules from the baseline system building. For the Hiero type1 (LO:Left-side is Open), the losses are trivial, and can be ignored in system buildings. However, if we remove the type2 (RO:Right-side is Open) Hiero rules, the losses will be noticeably large.

We carried out the second experiment, using each type of rules *only* for the system building, and results are in the middle part of Table 1, marked by "w/type" in the setups. The type2(RO) Hiero rules have the smallest drop from baseline; using only type4(NO:None Open) Hiero rules, however, we had the largest drop from the baseline Hiero system.

Setups	BLEUr4n4	TER
Baseline	54.44	40.24
-type1(LO)	54.27	40.15
-type2(RO)	53.87	40.53
-type3(BO)	54.00	40.40
-type4(NO)	54.09	40.37
w/type1(LO)	53.27	41.21
w/type2(RO)	53.88	40.41
w/type3(BO)	53.06	41.29
w/type4(NO)	52.84	41.38
SimpMix(4)	54.71	40.14
MLMix(4)	54.79	40.10
MonoSwap	54.62	40.22
SimpMix+MonoSwap	54.95	39.98
MLMix+MonoSwap	55.03	39.90

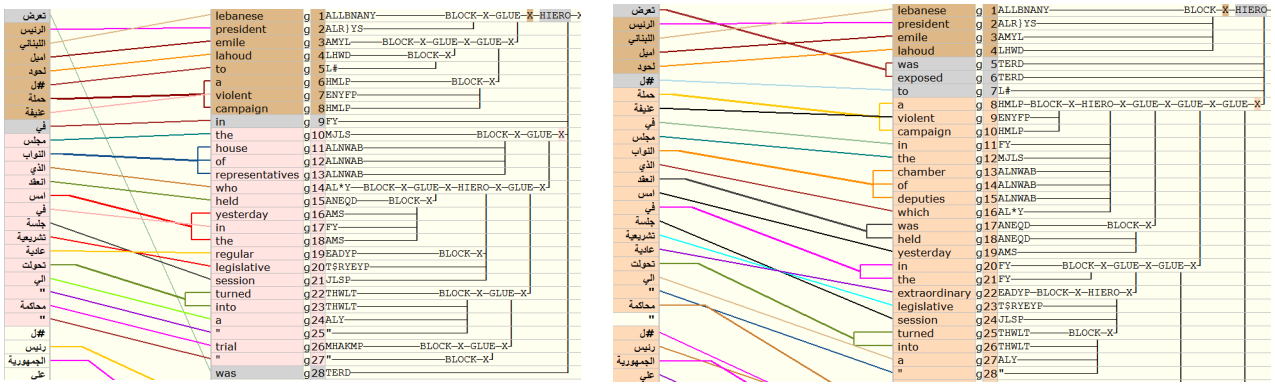
Table 1: On using categorized Hiero grammar. Baseline is a Hiero system without discriminating the sub categories; "-typeN" is the the baseline with removing a certain type of Hiero rules. SimpMix(4) is the one learned by simplex-downhill algorithms for four basic categories, and MLMix(4) is the cost function learned by inside-outside parsing for four basic types. SimpMix+MonoSwap integrated two more feature functions of monotone/nonmonotone, and MLMix+MonoSwap integrated the cross-product of the them.

Overall, the results show that different type of Hiero rules have different values for decoding. Some of them such as type1(LO) might have larger redundancies than other types; some of them are more valuable, like type2(RO) Hiero rules have more reordering values than others in our experiments for Arabic-English. This motivated us on weighting them properly for optimal combinations of the proposed categories, to encourage more particular types of reordering rules for better translations.

4.2 On Mixing Categorized Hiero rules

In Table 1, setup of "SimpMix(4)" is the approach as described in section 3.1 for combining four basic types of Hiero rules described sequentially in sections 2.1-2.4. Comparing with the setup of "MLMix(4)", the performances are almost the same in BLEUr4n4. Comparing with baseline, BLEU scores are improved from 54.44 to 54.79 using "MLMix(4)" — 0.35 improvement in BLEU.

For the first approach, as in section 3.1, there are six more dimensions added to the baseline. This becomes a challenge for optimization algorithms, because most are generally effective for a handful feature functions only.



(a) Hiero Baseline

(b) Hiero with Weighted-Mixed Subcategories

Figure 1: Reordering derivation trees using Hiero grammar from our chart-based decoder: baseline without clustering is as in (a), and with mixtures of clusters is as in (b). (a) showed a long jump for the word "tErD", a lexical item in the Hiero rule tErD X_1 fy $X_2 \leftrightarrow X1$ in_{fy} X_2 was_{tErD}; (b) showed the hypothesis for the same source part, with mixture-weighted Hiero reordering rules, presenting a much better word reordering for translations. ITG style GLUE rule is X_1 $X_2 \leftrightarrow X_1$ X_2. The distribution of $p(\text{type})$ is inferred via chart-parsing, and weights were learned via a simplex-downhill optimization algorithm.

For the second approach using chart-parsing, there is only one more dimension added to the baseline system, and this relatively reduces the burden on the optimization side, and the results are empirically more robust.

With further refined categories using the monotone/nonmonotone labels for each Hiero rule, as described in section 2.5, we can extend the above experimental setups into "SimpMix+MonoSwap" and "MLMix+MonoSwap". Similar observations were obtained, with maximum-likelihood approach slightly outperform binary-feature one. Both approaches outperform baseline by a margin of 0.59 and 0.51 BLEU points, respectively. On the other hand, by simply using only monotone/nonmonotone labels for a Hiero rule, the results are shown in the setup of "MonoSwap", and we still got an improvement of 0.18 BLEU, indicating the utilities of a simple effective classification for Hiero rules.

Overall, via mixing the proposed categories of the Hiero grammar, better grammar representations were obtained to improve Arabic-English translations.

4.3 Translation Examples

Figure 1 shows two translation hypotheses, one from the baseline system as in (a), the other from the setup "MLMix+MonoSwap" as shown in Table 1. Detailed derivation trees from our chart-based decoder are shown in the figure, including both the rule-level and word-level alignment. In a), a src word "tErD" was wrongly moved to more than 23 source tokens away, and translated into word "was". In (b), it is translated into the right verb-phrase "was exposed", with correct word order, using the Hiero rule of "tErD X_1 l# $X_2 \leftrightarrow$ X_1 was exposed to X_2", which belongs to our type2(RO: right-side is open), with nonmonotone reordering.

5 Conclusion

In this paper, we propose to classify the Hiero grammar into refined subcategories. A Hiero rule is characterized into four nonoverlapping categories by evaluating the nponterminals' left and right boundaries on the source side, and marked further with monotone or reordering alignment nature. We have obtained improved translation qualities over a strong Hiero system applied in GALE evaluations. Our results reveal there are redundancies and values for certain types of Hiero grammar, and fine-grained clusters are desired. Automatic clustering for the grammar is worthwhile for future research.

References

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

Songfang Huang and Bowen Zhou. 2009. An EM algorithm for SCFG in formal syntax-based translation. In *Proc. ICASSP*, pages 4813–4816, Taiwan, China, April.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of HLT-EMNLP*, pages 89–96, Vancouver, Canada, October.

Libin Shen, Jinxi Xu, Bing Zhang, Spyros Matsoukas, and Ralph Weischedel. 2009. Effective use of linguistic and contextual information for statistical machine translation. In *Proceedings of EMNLP*, pages 72–80, Singapore, August.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. In *Computational Linguistics*, volume 23(3), pages 377–403.

Bing Zhao and Yaser Al-Onaizan. 2008. Generalizing local and non-local word-reordering patterns for syntax-based machine translation. In *Proceedings of EMNLP*, pages 572–581, Honolulu, Hawaii, October.