# Head-modifier Relation based Non-lexical Reordering Model for Phrase-Based Translation

**Shui Liu[1], Sheng Li[1], Tiejun Zhao[1], Min Zhang[2], Pengyuan Liu[3]**
[1]School of Computer Science and Technology, Habin Institute of Technology
{liushui,lisheng,tjzhao}@mtlab.hit.edu.cn
[2]**Institute for Infocomm Research**
mzhang@i2r.a-star.edu.sg
[3]**Institute of Computational Linguistics, Peking University**
liupengyuan@pku.edu.cn

## Abstract

Phrase-based statistical MT (SMT) is a milestone in MT. However, the translation model in the phrase based SMT is structure free which greatly limits its reordering capacity. To address this issue, we propose a non-lexical head-modifier based reordering model on word level by utilizing constituent based parse tree in source side. Our experimental results on the NIST Chinese-English benchmarking data show that, with a very small size model, our method significantly outperforms the baseline by 1.48% bleu score.

## 1 Introduction

Syntax has been successfully applied to SMT to improve translation performance. Research in applying syntax information to SMT has been carried out in two aspects. On the one hand, the syntax knowledge is employed by directly integrating the syntactic structure into the translation rules i.e. syntactic translation rules. On this perspective, the word order of the target translation is modeled by the syntax structure explicitly. Chiang (2005), Wu (1997) and Xiong (2006) learn the syntax rules using the formal grammars. While more research is conducted to learn syntax rules with the help of linguistic analysis (Yamada and Knight, 2001; Graehl and Knight, 2004). However, there are some challenges to these models. Firstly, the linguistic analysis is far from perfect. Most of these methods require an off-the-shelf parser to generate syntactic structure, which makes the translation results sensitive to the parsing errors to some extent. To tackle this problem, n-best parse trees and parsing forest (Mi and Huang, 2008; Zhang, 2009) are proposed to relieve the error propagation brought by linguistic analysis. Secondly, some phrases which violate the boundary of linguistic analysis are also useful in these models ( DeNeefe et al., 2007; Cowan et al. 2006). Thus, a tradeoff needs to be found between linguistic sense and formal sense.

On the other hand, instead of using syntactic translation rules, some previous work attempts to learn the syntax knowledge separately and then integrated those knowledge to the original constraint. Marton and Resnik (2008) utilize the language linguistic analysis that is derived from parse tree to constrain the translation in a soft way. By doing so, this approach addresses the challenges brought by linguistic analysis through the log-linear model in a soft way.

Starting from the state-of-the-art phrase based model Moses ( Koehn e.t. al, 2007), we propose a head-modifier relation based reordering model and use the proposed model as a soft syntax constraint in the phrase-based translation framework. Compared with most of previous soft constraint models, we study the way to utilize the constituent based parse tree structure by mapping the parse tree to sets of head-modifier for phrase reordering. In this way, we build a word level reordering model instead of phrasal/constituent level model. In our model, with the help of the alignment and the head-modifier dependency based relationship in the source side, the reordering type of each target word with alignment in source side is identified as one of pre-defined reordering types. With these reordering types, the reordering of phrase in translation is estimated on word level.

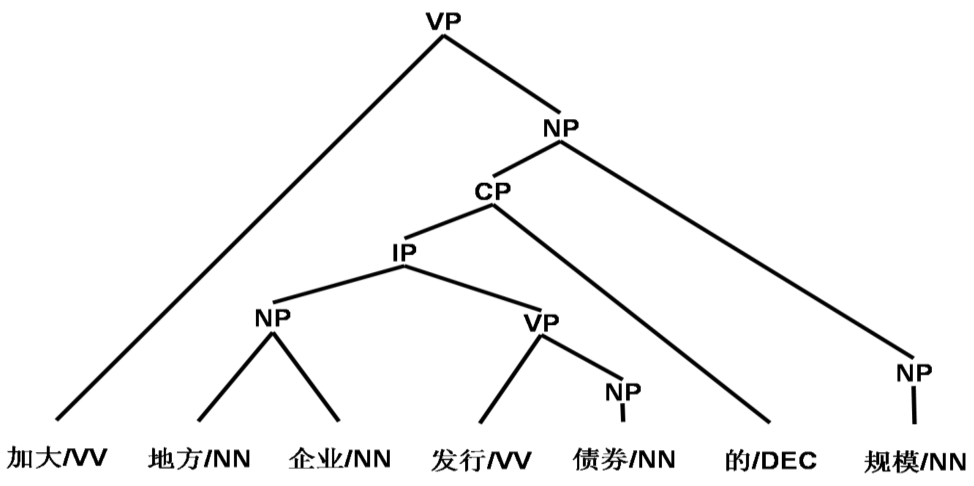

Fig 1. An Constituent based Parse Tree

## 2 Baseline

Moses, a state-of-the-art phrase based SMT system is used as our baseline system. In Moses, given the source language *f* and target language *e*, the decoder is to find:

$$e_{best} = \text{argmax}_e\, p ( e \mid f ) \, pLM ( e ) \, \omega^{length(e)} \quad (1)$$

where p(e|f) can be computed using phrase translation model, distortion model and lexical reordering model. pLM(e) can be computed using the language model. $\omega^{length(e)}$ is word penalty model.

Among the above models, there are three reordering-related components: language model, lexical reordering model and distortion model. The language model can reorder the local target words within a fixed window in an implied way. The lexical reordering model and distortion reordering model tackle the reordering problem between adjacent phrase on lexical level and alignment level. Besides these reordering model, the decoder induces distortion pruning constraints to encourage the decoder translate the leftmost uncovered word in the source side firstly and to limit the reordering within a certain range.

## 3 Model

In this paper, we utilize the constituent parse tree of source language to enhance the reordering capacity of the translation model. Instead of directly employing the parse tree fragments (Bod, 1992; Johnson, 1998) in reordering rules (Huang and Knight, 2006; Liu 2006; Zhang and Jiang 2008), we make a mapping from trees to sets of head-modifier dependency relations (Collins 1996 ) which can be obtained from the constituent based parse tree with the help of head rules ( Bikel, 2004 ).

### 3.1 Head-modifier Relation

According to Klein and Manning (2003) and Collins (1999), there are two shortcomings in n-ary Treebank grammar. Firstly, the grammar is too coarse for parsing. The rules in different context always have different distributions. Secondly, the rules learned from training corpus cannot cover the rules in testing set.

Currently, the state-of-the-art parsing algorithms (Klein and Manning, 2003; Collins 1999) decompose the n-ary Treebank grammar into sets of head-modifier relationships. The parsing rules in these algorithms are constructed in the form of finer-grained binary head-modifier dependency relationships. Fig.2 presents an example of head-modifier based dependency tree mapped from the constituent parse tree in Fig.1.

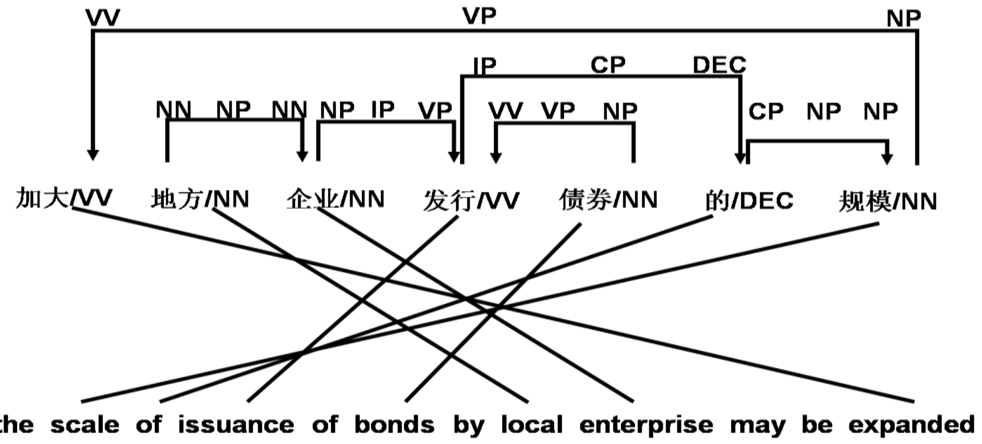

Fig. 2. Head-modifier Relationships with Aligned Translation

Moreover, there are several reasons for which we adopt the head-modifier structured tree as the main frame of our reordering model. Firstly, the dependency relationships can reflect some underlying binary long distance dependency relations in the source side. Thus, binary dependency structure will suffer less from the long distance reordering constraint. Secondly, in head-modifier relation, we not only can utilize the context of dependency relation in reordering model, but also can utilize some well-known and proved helpful context (Johnson, 1998) of constituent base parse tree in reordering model. Finally, head-modifier relationship is mature and widely adopted method in full parsing.

## 3.2 Head-modifier Relation Based Reordering Model

Before elaborating the model, we define some notions further easy understanding. S=<$f_1$, $f_2$…$f_n$> is the source sentence; T=<$e_1$,$e_2$,…,$e_m$> is the target sentence; $A_S$={$a_s(i)$ | 1≤ $a_s(i)$ ≤ n } where $a_s(i)$ represents that the ith word in source sentence aligned to the $a_s(i)th$ word in target sentence; $A_T$={$a_T(i)$ | 1≤ $a_T(i)$ ≤ n } where $a_T(i)$ represents that the ith word in target sentence aligned to the $a_T(i)th$ word in source sentence; D= {( d(i), r(i) )| 0≤ d(i) ≤n} is the head-modifier relation set of the words in S where *d(i)* represents that the ith word in source sentence is the modifier of *d(i)th* word in source sentence under relationship *r(i)*; O= < $o_1$, $o_2$,…, $o_m$ > is the sequence of the reordering type of every word in target language. The reordering model probability is P(O| S, T, D, A).

**Relationship**: in this paper, we not only use the label of the constituent label as Collins (1996), but also use some well-known context in parsing to define the head-modifier relationship *r(.)*, including the POS of the modifier *m*, the POS of the head *h*, the dependency direction *d*, the parent label of the dependency label *l*, the grandfather label of the dependency relation *p*, the POS of adjacent siblings of the modifier *s*. Thus, the head-modifier relationship can be represented as a 6-tuple <*m*, *h*, *d*, *l*, *p*, *s*>.

| r(.) | relationship |
|---|---|
| r(1) | <VV, - , -, -, -, - > |
| r(2) | <NN, NN, right, NP, IP, - > |
| r(3) | <NN,VV, right, IP, CP, - > |
| r(4) | <VV, DEC, right, CP, NP, - > |
| r(5) | <NN,VV, left, VP, CP, - > |
| r(6) | <DEC, NP, right, NP, VP, - > |
| r(7) | <NN, VV, left, VP, TOP, - > |

Table 1. Relations Extracted from Fig 2.

In Table 1, there are 7 relationships extracted from the source head-modifier based dependency tree as shown in Fig.2. Please notice that, in this paper, each source word has a corresponding relation.

**Reordering type:** there are 4 reordering types for target words with linked word in the source side in our model: R= {$rm_1$, $rm_2$, $rm_3$, $rm_4$}. The reordering type of target word $a_s(i)$ is defined as follows:

- **$rm_1$**: if the position number of the ith word's head is less than i ( d(i) < i ) in source language, while the position number of the word aligned to i is less than

$a_s(d(i))$ ($a_s(i) < a_s(d(i))$ ) in target language;

- **rm$_2$**: if the position number of the ith word's head is less than i ( $d(i) < i$ ) in source language, while the position number of the word aligned to i is larger than $a_s(d(i))$ ($a_s(i) > a_s(d(i))$ ) in target language.
- **rm$_3$**: if the position number of the ith word's head is larger than i ( $d(i) > i$ ) in source language, while the position number of the word aligned to i is larger than $a_s(d(i))$ ($a_s(i) > a_s(d(i))$) in target language.
- **rm$_4$**: if the position number of the ith word's head is larger than i ( $d(i) > i$) in source language, while the position number of the word aligned to i is less than $a_s(d(i))$ ($a_s(i) < a_s(d(i))$ ) in target language.

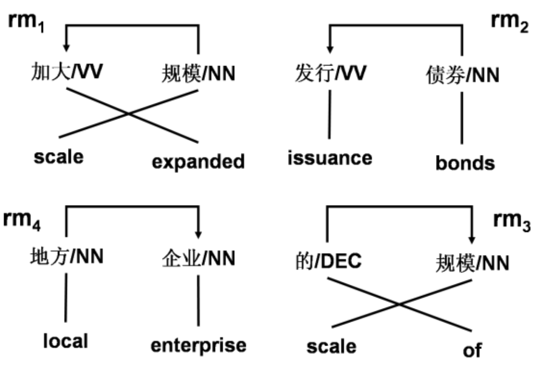

Fig. 3. An example of the reordering types in Fig. 2.

Fig. 3 shows examples of all the reordering types. In Fig. 3, the reordering type is labeled at the target word aligned to the modifier: for example, the reordering type of rm$_1$ belongs to the target word "*scale*". Please note that, in general, these four types of reordering can be divided into 2 categories: the target words order of rm$_2$ and rm$_4$ is identical with source word order, while rm$_1$ and rm$_3$ is the swapped order of source. In practice, there are some special cases that can't be classified into any of the defined reordering types: the head and modifier in source link to the same word in target. In such cases, rather than define new reordering types, we classify these special cases into these four defined reordering types: if the head is right to the modifier in source, we classify the reordering type into rm$_2$; otherwise, we classify the reordering type into rm$_4$.

**Probability estimation:** we adopt maximum likelihood (ML) based estimation in this paper. In ML estimation, in order to avoid the data sparse problem brought by lexicalization, we discard the lexical information in source and target language:

$$P(O|S,T,D,A) = \prod_{i=1}^{m} P(o_i \mid -,-, r(a_T(i))) \tag{2}$$

where $o_i \in \{rm1, rm2, rm3, rm4\}$ is the reordering type of ith word in target language.

To get a non-zero probability, additive smooth ing( Chen and Goodman, 1998) is used:

$$P(o_i \mid -,-, F(r(a_t(i)))) = \frac{F(o_i, r(a_T(i))) + \alpha}{\sum_{o_i \in R} F(r(a_t(i))) + |O| \times \alpha} = \frac{F(o_i, m_{a_T(i)}, h_{a_T(i)}, d_{a_T(i)}, l_{a_T(i)}, p_{a_T(i)}, s_{a_T(i)}) + \alpha}{\sum_{o_i \in R} F(m_{a_T(i)}, h_{a_T(i)}, d_{a_T(i)}, l_{a_T(i)}, p_{a_T(i)}, s_{a_T(i)}) + |O| \times \alpha} \tag{3}$$

where F(.) is the frequency of the statistic event in training corpus. For a given set of dependency relationships mapping from constituent tree, the reordering type of ith word is confined to two types: it is whether one of rm1 and rm2 or rm3 and rm4. Therefore, |O|=2 instead of |O|=4 in (2). The parameter α is an additive factor to prevent zero probability. It is computed as:

$$\alpha = \frac{1}{C \times \sum_{o_i \in R} F(m_{a_T(i)}, h_{a_T(i)}, d_{a_T(i)}, l_{a_T(i)}, p_{a_T(i)}, s_{a_T(i)})} \tag{4}$$

where c is a constant parameter(c=5 in this paper).

In above, the additive parameter α is an adaptive parameter decreasing with the size of the statistic space. By doing this, the data sparse problem can be relieved.

## 4 Apply the Model to Decoder

Our decoding algorithm is exactly the same as (Kohn, 2004). In the translation procedure, the decoder keeps on extending new phrases without overlapping, until all source words are translated. In the procedure, the order of the target words in decoding procedure is fixed. That is, once a hypothesis is generated, the order of target words cannot be changed in the future. Taking advantage of this feature, instead of computing a totally new reordering score for a newly generated hypothesis, we merely calculate the reordering score of newly extended part of the hypothesis in decoding. Thus, in decoding, to compute the reordering score, the reordering types of each target word in the newly extended phrase need to be identified.

The method to identify the reordering types in decoding is proposed in Fig.4. According to the definition of reordering, the reordering type of the target word is identified by the direction of head-modifier dependency on the source side, the alignment between the source side and target side, and the relative translated order of word pair under the head-modifier relationship. The direction of dependency and the alignment can be obtained in input sentence and phrase table. While the relative translation order needs to record during decoding. A word index is employed to record the order. The index is constructed in the form of true/false array: the index of the source word is set with true when the word has been translated. With the help of this index, reordering type of every word in the phrase can be identified.

```
1: **Input**: alignment array $A_T$; the *Start* is the
     start position of the phrase in the source side;
     head-modifier relation *d(.)*; source word in-
     dex *C*, where *C[i]=true* indicates that the
     *ith* word in source has been translated.
2: **Output:** reordering type array *O* which re-
     serves the reordering types of each word in
     the target phrase
3: for i = 1, |$A_T$| do
4:   P $\leftarrow a_T(i)$ + Start
5:   if (d (P)<P) then
6:     if C [d(p)] = false then
7:       O[i] $\leftarrow rm_1$
8:     else
9:       O[i] $\leftarrow rm_2$
10:    end if
11:  else
12:    if C[d(p)] = true then
13:      O[i] $\leftarrow rm_3$
14:    else
15:      O[i] $\leftarrow rm_4$
16:    end if
17:  end if
18: C[p] $\leftarrow$true //update word index
19: end for
```

Fig. 4. Identify the Reordering Types of Newly Extended Phrase

After all the reordering types in the newly extended phrase are identified, the reordering scores of the phrase can be computed by using equation (3).

## 5 Preprocess the Alignment

In Fig. 4, the word index is to identify the reordering type of the target translated words. Actually, in order to use the word index without ambiguity, the alignment in the proposed algorithm needs to satisfy some constraints.

Firstly, every word in the source must have alignment word in the target side. Because, in the decoding procedure, if the head word is not covered by the word index, the algorithm cannot distinguish between the head word will not be translated in the future and the head word is not translated yet. Furthermore, in decoding, as shown in Fig.4, the index of source would be set with true only when there is word in target linked to it. Thus, the index of the source word without alignment in target is never set with true.

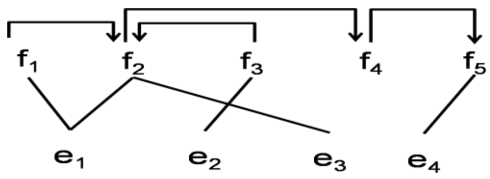

Fig. 5. A complicated Example of Alignment in Head-modifier based Reordering Model

Secondly, if the head word has more than one alignment words in target, different alignment possibly result in different reordering type. For example, in Fig. 5, the reordering type of $e_2$ is different when $f_2$ select to link word $e_1$ and e3 in the source side.

To solve this problem, we modify the alignment to satisfy following conditions: a) each word in source just has only one alignment word in target, and b) each word in target has at most one word aligned in source as its anchor word which decides the reordering type of the target word.

To make the alignment satisfy above constraints, we modify the alignment in corpus. In order to explain the alignment preprocessing, the following notions are defined: if there is a link between the source word $f_j$ and target word $e_i$, let $l(e_i, f_j) = 1$, otherwise $l(e_i, f_j) = 0$; the source word $f_j \in F_{1\text{-to-}N}$, iff $\sum_i l(e_i,f_j) > 1$, such as the source word $f_2$ in Fig. 5; the source word $f_j \in F_{NULL}$, iff $\sum_i l(e_i,f_j) = 0$, such as the source word $f_4$ in Fig. 5; the target word $e_i \in E_{1\text{-to-}N}$, iff $\sum_j l(e_i,f_j) > 1$, such as the target word $e_1$ in Fig. 5.

In preprocessing, there are 3 types of operation, including *DiscardLink($f_j$)*, *BorrowLink($f_j$)* and *FindAnchor($e_i$)*:

**DiscardLink( $f_j$)** : if the word $f_j$ in source with more than one words aligned in target, i.e. $f_j \in F_{1\text{-to-}N}$; We set the target word $e_n$ with $l(e_n, f_j) = 1$, where $e_n = \operatorname{argmax}_i\ p(e_i \mid f_j)$ and $p(e_i \mid f_j)$ is estimated by ( Koehn e.t. al, 2003), while set rest of words linked to $f_j$ with $l(e_n, f_j) = 0$.

**BorrowLink( $f_j$)**: if the word $f_j$ in source without a alignment word in target, i.e. $f_j \in F_{NULL}$; let $l(e_i,f_j)=1$ where $e_i$ aligned to the word $f_j$, which is the nearest word to $f_j$ in the source side; when there are two words nearest to $f_j$ with alignment words in the target side at the same time, we select the alignment of the left word firstly .

**FindAnchor($e_i$)**: for the word $e_i$ in target with more than one words aligned in source , i.e. $e_i \in E_{1\text{-to-}N}$; we select the word $f_m$ aligned to $e_i$ as its anchor word to decide the reordering type of $e_i$, where $f_m = \operatorname{argmax}_j\ p(e_i \mid f_j)$ and $p(f_j \mid e_i)$ is estimated by ( Koehn et al, 2003); For the rest of words aligned to $e_i$, we would set their word indexes with true in the update procedure of decoding in the 18th line of Fig.4.

With these operations, the required alignment can be obtained by preprocessing the origin alignment as shown in Fig. 6.

```
1: Input: set of alignment A between target language e and source language f
2: Output: the 1-to-1 alignment required by the model
3: foreach $f_i \in F_{1\text{-to-}N}$ do
4:   DiscardLink( $f_i$ )
5: end for
6: foreach $f_i \in F_{NULL}$ do
7:   BorrowLink( $f_i$ )
8: end for
9: foreach $e_i \in E_{1\text{-to-}N}$ do
10:   FindAnchor($e_i$ )
11: endfor
```

Fig. 6. Alignment Pre-Processing algorithm

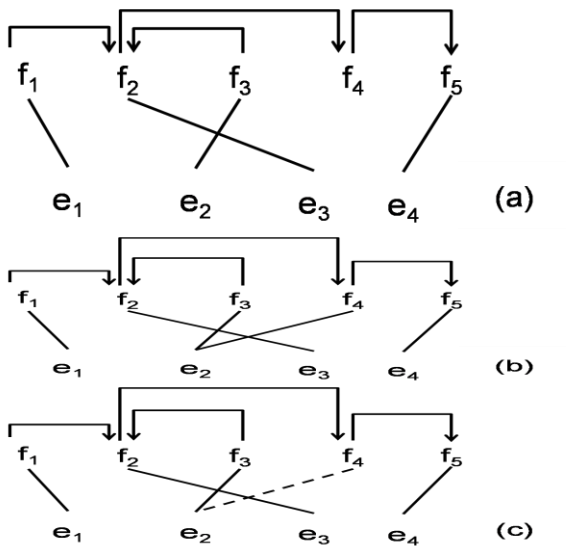

Fig. 7. An Example of Alignment Preprocessing.

An example of the preprocess the alignment in Fig. 5 is shown in Fig. 7 : firstly, *DiscardLink($f_2$)* operation discards the link between $f_2$ and $e_1$ in (a); then the link between $f_4$ and $e_3$ is established by operation *BorrowLink($f_4$ )* in (b); at last, *FindAnchor($e_3$)* select $f_2$ as the anchor word of $e_3$ in source in (c). After the preprocessing, the reordering type of $e_3$ can be identified. Furthermore, in decoding, when the decoder scans over $e_2$, the word index sets the word index of $f_3$ and $f_4$ with true. In this way, the never-true word indexes in decoding are avoided.

## 6 Training the Reordering Model

Before training, we get the required alignment by alignment preprocessing as indicated above. Then we train the reordering model with this alignment: from the first word to the last word in the target side, the reordering type of each word is identified. In this procedure, we skip the words without alignment in source. Finally, all the statistic events required in equation (3) are added to the model.

In our model, there are 20,338 kinds of relations with reordering probabilities which are much smaller than most phrase level reordering models on the training corpus FBIS.

Table 1 is the distribution of different reordering types in training model.

| **Type of Reordering** | **Percentage %** |
|---|---|
| $rm_1$ | 3.69 |
| $rm_2$ | 27.61 |
| $rm_3$ | 20.94 |
| $rm_4$ | 47.75 |

Table 1: Percentage of different reordering types in model

From Table 1, we can conclude that the reordering type $rm_2$ and $rm_4$ are preferable in reordering which take over nearly 3/4 of total number of reordering type and are identical with word order of the source. The statistic data indicate that most of the words order doesn't change in our head-modifier reordering view. This maybe can explain why the models (Wu, 1997; Xiong, 2006; Koehn, et., 2003) with limited capacity of reordering can reach certain performance.

## 7 Experiment and Discussion

### 7.1 Experiment Settings

We perform Chinese-to-English translation task on NIST MT-05 test set, and use NIST MT-02 as our tuning set. FBIS corpus is selected as our training corpus, which contains 7.06M Chinese words and 9.15M English words. We use GIZA++(Och and Ney, 2000) to make the corpus aligned. A 4-gram language model is trained using Xinhua portion of the English Gigaword corpus (181M words). All models are tuned on BLEU, and evaluated on both BLEU and NIST score.

To map from the constituent trees to sets of head-modifier relationships, firstly we use the Stanford parser (Klein, 2003) to parse the source of corpus FBIS, then we use the head-finding rules in (Bikel, 2004) to get the head-modifier dependency sets.

In our system, there are 7 groups of features. They are:

1. Language model score (1 feature)
2. word penalty score (1 feature)
3. phrase model scores (5 features)
4. distortion score (1 feature)
5. lexical RM scores (6 features)
6. Number of each reordering type (4 features)
7. Scores of each reordering type (4 features, computed by equation (3))

In these feature groups, the top 5 groups of features are the baseline model, the left two group scores are related with our model.

In decoding, we drop all the OOV words and use default setting in Moses: set the distortion limitation with 6, beam-width with 1/100000, stack size with 200 and max number of phrases for each span with 50.

### 7.2 Results and Discussion

We take the replicated Moses system as our baseline. Table 2 shows the results of our model. In the table, Baseline model is the model including feature group 1, 2, 3 and 4. $Baseline_{rm}$ model is the Baseline model with feature group 5. H-M model is the Baseline model with feature group 6 and 7. $H\text{-}M_{rm}$ model is the $Baseline_{rm}$ model with feature group 6 and 7.

| **Model** | **BLEU%** | **NIST** |
|---|---|---|
| Baseline | 27.06 | 7.7898 |
| $Baseline_{rm}$ | 27.58 | 7.8477 |
| H-M | 28.47 | 8.1491 |
| $H\text{-}M_{rm}$ | 29.06 | 8.0875 |

Table 2: Performance of the Systems on NIST-05(bleu4 case-insensitive).

From table 2, we can conclude that our reordering model is very effective. After adding feature group 6 and 7, the performance is improved by 1.41% and 1.48% in bleu score separately. Our reordering model is more effective than the lexical reordering model in Moses: 1.41% in bleu score is improved by adding our reordering model to Baseline model, while 0.48 is improved by adding the lexical reordering to Baseline model.

| threshold | KOR | BLEU | NIST |
|---|---|---|---|
| $\geqslant 1$ | 20,338 | 29.06 | 8.0875 |
| $\geqslant 2$ | 13,447 | 28.83 | 8.3658 |
| $\geqslant 3$ | 10,885 | 28.64 | 8.0350 |
| $\geqslant 4$ | 9,518 | 28.94 | 8.1002 |
| $\geqslant 5$ | 8,577 | 29.18 | 8.1213 |

Table 3: Performance on NIST-05 with Different Relation Frequency Threshold (bleu4 case-insensitive).

Although our model is lexical free, the data sparse problem affects the performance of the model. In the reordering model, nearly half numbers of the relations in our model occur less than three times. To investigate this, we statistic the frequency of the relationships in our model, and expertise our H-M$_{full}$ model with different frequency threshold.

In Table 3, when the frequency of relation is not less than the *threshold*, the relation is added into the reordering model; KOR is the number of relation type in the reordering model.

Table 3 shows that, in our model, many relations occur only once. However, these low-frequency relations can improve the performance of the model according to the experimental results. Although low frequency statistic events always do harm to the parameter estimation in ML, the model can estimate more events in the test corpus with the help of low frequency event. These two factors affect the experiment results on opposite directions: we consider that is the reason the result don't increase or decrease with the increasing of frequency threshold in the model. According to the results, the model without frequency threshold achieves the highest bleu score. Then, the performance drops quickly, when the frequency threshold is set with 2. It is because there are many events can't be estimated by the smaller model. Although, in the model without frequency threshold, there are some probabilities overestimated by these events which occur only once, the size of the model affects the performance to a larger extent. When the frequency threshold increases above 3, the size of model reduces slowly which makes the overestimating problem become the important factor affecting performance. From these results, we can see the potential ability of our model: if our model suffer less from data spars problem, the performance should be further improved, which is to be verified in the future.

## 8 Related Work and Motivation

There are several researches on adding linguistic analysis to MT in a "soft constraint" way. Most of them are based on constituents in parse tree. Chiang(2005), Marton and Resnik(2008) explored the constituent match/violation in hiero; Xiong (2009 a) added constituent parse tree based linguistic analysis into BTG model; Xiong (2009 b) added source dependency structure to BTG; Zhang(2009) added tree-kernel to BTG model. All these studies show promising results. Making soft constrain is an easy and efficient way in adding linguistic analysis into formal sense SMT model.

In modeling the reordering, most of previous studies are on phrase level. In Moses, the lexical reordering is modeled on adjacent phrases. In (Wu, 1996; Xiong, 2006), the reordering is also modeled on adjacent translated phrases. In hiero, the reordering is modeled on the segments of the unmotivated translation rules. The tree-to-string models (Yamada et al. 2001; Liu et al.2006) are model on phrases with syntax representations. All these studies show excellent performance, while there are few studies on word level model in recent years. It is because, we consider, the alignment in word level model is complex which limits the reordering capacity of word level models.

However, our work exploits a new direction in reordering that, by utilizing the decomposed dependency relations mapped from parse tree as a soft constraint, we proposed a novel head-modifier relation based word level reordering model. The word level reordering model is based on a phrase based SMT framework. Thus, the task to find the proper position of translated words converts to score the reordering of the translated words, which relax the tension between complex alignment and word level reordering in MT.

## 9 Conclusion and Future Work

Experimental results show our head-modifier relationship base model is effective to the baseline (enhance by 1.48% bleu score), even with limited size of model and simple parameter estimation. In the future, we will try more complicated smooth methods or use maximum entropy based reordering model. We will study the performance with larger distortion constraint, such as the performances of the distortion constraint over 15, or even the performance without distortion model.

## 10 Acknowledgement

The work of this paper is funded by National Natural Science Foundation of China (grant no. 60736014), National High Technology Research and Development Program of China (863 Program) (grant no. 2006AA010108), and Microsoft Research Asia IFP (grant no. FY09-RES-THEME-158).

## References

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Lingustics*,23(3):377-403.

David Chiang. 2005. A hierarchical phrase-based model for SMT. ACL-05.263-270.

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

Kenji Yamada and K. Knight. 2001. A syntax-based statistical translation model. *ACL-01*.523-530.

Yuval Marton and Philip Resnik. 2008. Soft syntactic Constraints for Hierarchical Phrased-based Translation. *ACL-08*. 1003-1011.

Libin shen, Jinxi Xu and Ralph Weischedel. 2008. A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model. *ACL-08*. 577-585.

J. Graehl and K. Knight.2004.Training Tree transducers. In proceedings of *the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Dekai Wu. 1996. A Polynomial-Time Algorithm for Statistical Machine Translation. In proceedings of *ACL-1996*

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In proceedings of *COLING-ACL 2006*

Deyi Xiong, Min Zhang, Aiti AW and Haizhou Li. 2009a. A Syntax-Driven Bracket Model for Phrase-Based Translation. *ACL-09*.315-323.

Deyi Xiong, Min Zhang, Aiti AW and Haizhou Li. 2009b. A Source Dependency Model for Statistic Machine translation. *MT-Summit 2009*.

Och, F.J. and Ney, H. 2000. Improved statistical alignment models. In Proceedings of *ACL 38*.

Philipp Koehn, et al. Moses: Open Source Toolkit for Statistical Machine Translation, ACL 2007.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu.2003. Statistical Phrase-based Translation. In Proceedings of *HLT-NAACL*.

Philipp Koehn. 2004. A Beam Search Decoder for Phrase-Based Translation model. In: Proceeding of *AMTA-2004*,Washington

Rens Bod. 1992. Data oriented Parsing( DOP ). In Proceedings of *COLING-92*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. Computational Linguistics, 24:613-632.

Liang Huang, Kevin Knight, and Aravind Joshi. Statistical Syntax-Directed Translation with Extended Domain of Locality. 2006. In Proceedings of *the 7th AMTA*.

Yang Liu, Qun Liu, and Shouxun Lin. Tree-to-String Alignment Template for Statistical Machine Translation. 2006.In Proceedings of t*he ACL 2006*.

Min Zhang, Hongfei Jiang, Ai Ti Aw, Haizhou Li, Chew Lim Tan and Sheng Li. 2008. A Tree Sequence Alignment-based Tree-to-Tree Translation Model. *ACL-HLT-08*. 559-567.

Dan Klein, Christopher D. Manning. Accurate Unlexicalized Parsing. 2003. In Proceedings of *ACL-03*. 423-430.

M. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In Proceedings of *ACL-96*. 184-191.

M. Collins. 1999. Head-Driven Statistical Models for Natural Language Parsing. Ph.D. thesis, Univ. of Pennsylvania.

Andreas Zollmann. 2005. A Consistent and Efficient Estimator for the Data-Oriented Parsing Model. Journal of Automata, Languages and Combinatorics. 2005(10):367-388

Mark Johnson. 2002. The DOP estimation method is biased and inconsistent. *Computational Linguistics* 28, 71-76.

Daniel M. Bikel. 2004. On the Parameter Space of Generative Lexicalized Statistical Parsing Models. Ph.D. thesis. Univ. of Pennsylvania.

S. F. Chen, J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. In Proceedings of *the 34th annual meeting on Association for Computational Linguistics*, 1996.310-318.

Haitao Mi and Liang Huang. 2008. Forest-based translation Rule Extraction. *ENMLP*-08. 2006-214.

Hui Zhang, Min Zhang , Haizhou Li, Aiti Aw and Chew Lim Tan. Forest-based Tree Sequence to String Translation Model. *ACL*-09: 172-180

S DeNeefe, K. Knight, W. Wang, and D. Marcu. 2007. What can syntax-based MT learn from phrase-based MT ? In Proc. *EMNLP-CoNULL*.

Brooke Cowan, Ivona Kucerova, and Michael Collins.2006. A discriminative model for tree-to-tree translation. In Proc. *EMNLP*.