An Efficient Shift-Reduce Decoding Algorithm for Phrased-Based Machine Translation

Yang Feng, Haitao Mi, Yang Liu and Qun Liu
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
{fengyang,htmi,yliu,liuqun}@ict.ac.cn

Abstract

In statistical machine translation, decoding without any reordering constraint is an NP-hard problem. Inversion Transduction Grammars (ITGs) exploit linguistic structure and can well balance the needed flexibility against complexity constraints. Currently, translation models with ITG constraints usually employs the cube-time CYK algorithm. In this paper, we present a shift-reduce decoding algorithm that can generate ITG-legal translation from left to right in linear time. This algorithm runs in a *reduce-eager* style and is suited to phrase-based models. Using the state-of-the-art decoder Moses as the baseline, experiment results show that the shift-reduce algorithm can significantly improve both the accuracy and the speed on different test sets.

1 Introduction

In statistical machine translation, for the diversity of natural languages, the word order of source and target language may differ and searching through all possible translations is NP-hard (Knight, 1999). So some measures have to be taken to reduce search space: either using a search algorithm with pruning technique or restricting possible reorderings.

Currently, beam search is widely used (Tillmann and Ney, 2003; Koehn, 2004) to reduce search space. However, the pruning technique adopted by this algorithm is not risk-free. As a result, the best partial translation may be ruled out during pruning. The more aggressive the pruning is, the more likely the best translation escapes. There should be a tradeoff between the speed and the accuracy. If some heuristic knowledge is employed to guide the search, the search algorithm can discard some implausible hypotheses in advance and focus on more possible ones.

Inversion Transduction Grammars (ITGs) permit a minimal extra degree of ordering flexibility and are particularly well suited to modeling ordering shifts between languages (Wu, 1996; Wu, 1997). They can well balance the needed flexibility against complexity constraints. Recently, ITG has been successfully applied to statistical machine translation (Zens and Ney, 2003; Zens et al., 2004; Xiong et al., 2006). However, ITG generally employs the expensive CYK parsing algorithm which runs in cube time. In addition, the CYK algorithm can not calculate language model exactly in the process of decoding, as it can not catch the full history context of the left words in a hypothesis.

In this paper, we introduce a shift-reduce decoding algorithm with ITG constraints which runs in a left-to-right manner. This algorithm parses source words in the order of their corresponding translations on the target side. In the meantime, it gives all candidate ITG-legal reorderings. The shift-reduce algorithm is different from the CYK algorithm, in particular:

- It produces translation in a left-to-right manner. As a result, language model probability can be calculated more precisely in the light of full history context.

- It decodes much faster. Applied with distor-

Coling 2010: Poster Volume, pages 285–293,
Beijing, August 2010

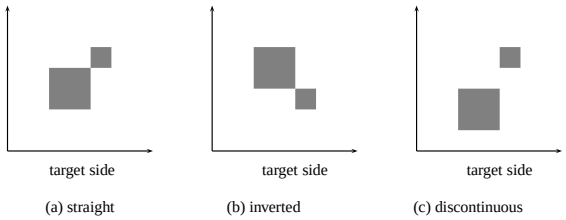


Figure 1: Orientation of two blocks.

tion limit, shift-reduce decoding algorithm can run in linear time, while the CYK runs in cube time.

- It holds ITG structures generated during decoding. That is to say, it can directly give ITG-legal spans, which leads to faster decoding. Furthermore, it can be extended to syntax-based models.

We evaluated the performance of the shift-reduce decoding algorithm by adding ITG constraints to the state-of-the-art decoder Moses. We did experiments on three data sets: NIST MT08 data set, NIST MT05 data set and China Workshop on Machine Translation 2007 data set. Compared to Moses, the improvements of the accuracy are 1.59, 0.62, 0.8 BLEU score, respectively, and the speed improvements are 15%, 24%, 30%, respectively.

2 Decoding with ITG constraints

In this paper, we employ the shift-reduce algorithm to add ITG constraints to phrase-based machine translation model. It is different from the traditional shift-reduce algorithm used in natural language parsing. On one hand, as natural language parsing has to cope with a high degree of ambiguity, it need take ambiguity into consideration. As a result, the traditional one often suffers *shift-reduce* divergence. Nonetheless, the shift-reduce algorithm in this paper does not pay attention to ambiguity and acts in a *reduce-eager* manner. On the other hand, the traditional algorithm can not ensure that all reorderings observe ITG constraints, so we have to modify the traditional algorithm to import ITG constraints.

We will introduce the shift-reduce decoding algorithm in the following two steps: First, we will deduce how to integrate the shift-reduce algorithm and ITG constraints and show its correctness (Section 2.1). Second, we will describe the shift-reduce decoding algorithm in details (Section 2.2).

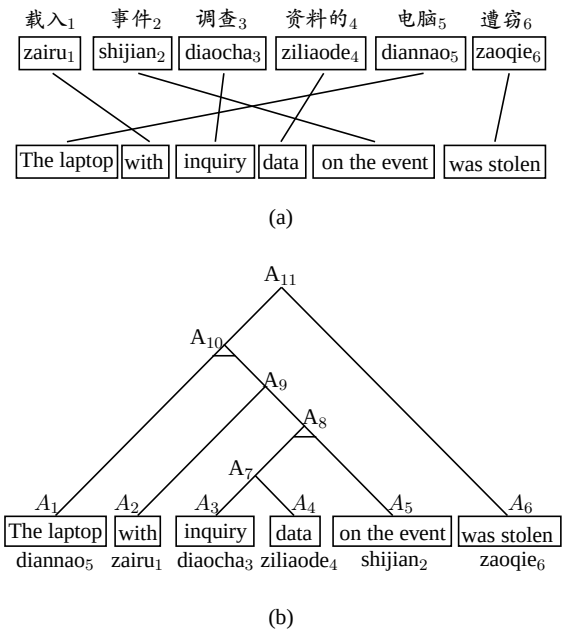


Figure 2: A Chinese-to-English sentence pair and its corresponding ITG tree.

2.1 Adding ITG constraints

In the process of decoding, a source phrase is regarded as a block and a source sentence is seen as a sequence of blocks. The orientation of two blocks whose translations are adjacent on the target side can be straight, inverted or discontinuous, as shown in Figure 1. According to ITG, two blocks which are straight or inverted can be merged into a single block. For parsing, different mergence order of a sequence of continuous blocks may yield different derivations. In contrast, the phrase-based machine translation does not compute reordering probabilities hierarchically, so the mergence order will not impact the computation of reordering probabilities. As a result, the shift-reduce decoding algorithm need not take into consideration the shift-reduce divergence. It merges two continuous blocks as soon as possible, acting in a reduce-eager style.

Every ITG-legal sentence pair has a corre-

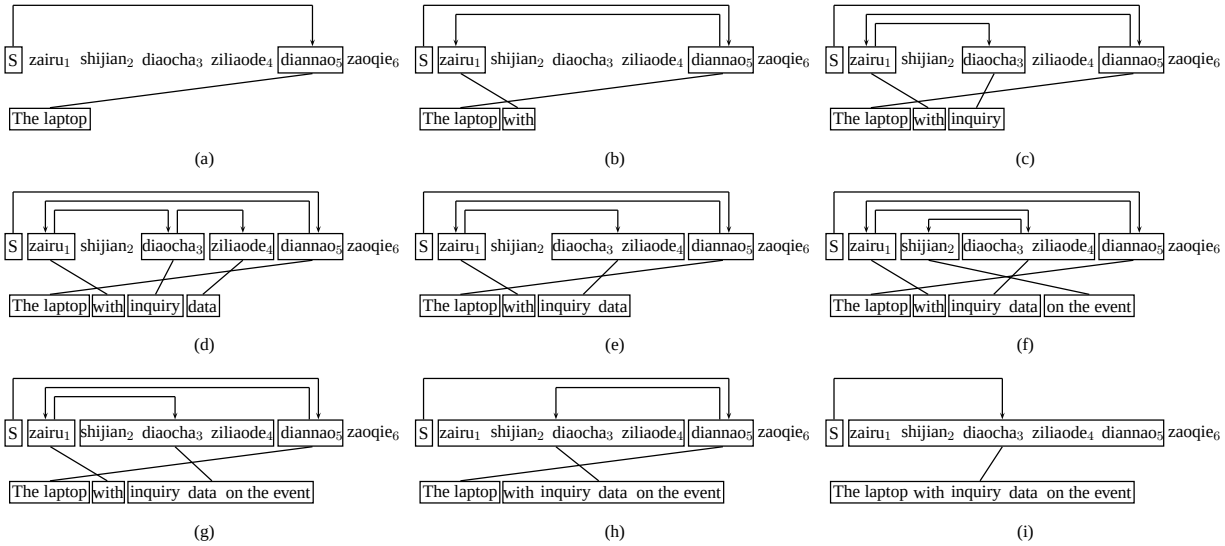


Figure 3: The partial translation procedure of the sentence in Figure 2.

sponding ITG tree, and source words covered by every node (eg. $A1, ..., A_{11}$ in Figure 2(b)) in the ITG tree can be seen as a block. By watching the tree in Figure 2, we can find that a block must be adjacent to the block either on its left or on its right, then they can be merged into a larger block. For example, A_2 matches the block $[zairu_1]$ and A_8 matches the block $[shijian_2\ diaocha_3\ ziliaode_4]$.[1] The two blocks are adjacent and they are merged into a larger block $[zairu_1\ shijian_2\ diaocha_3\ ziliaode_4]$, covered by A_9. The procedure of translating $zairu_1\ shijian_2\ diaocha_3\ ziliaode_4\ diannao_5$ is illustrated in Figure 3.

For a hypothesis during decoding, we assign it three factors: the current block, the left neighboring uncovered span and the right neighboring uncovered span. For example, in Figure 3(c), the current block is $[diaocha_3]$ and the left neighboring uncovered span is $[shijian_2]$ and the right neighboring uncovered span is $[ziliaode_4]$. $[zaoqie_6]$ is not thought of as the right neighboring block, for it is not adjacent to $[diaocha_3]$. The next covered block is $[ziliaode_4]$ (as shown in Figure 3(d)). For $[diaocha_3]$ and $[ziliaode_4]$ are adjacent, they are merged. In Figure 3(e), the current block is $[diaocha_3\ ziliaode_4]$.

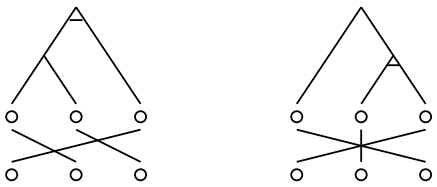
A sentence is translated with ITG constraints iff its source side can be covered by an ITG tree. That is to say, for every hypothesis during decoding, the next block to cover must be selected from the left or right neighboring uncovered span.

First, we show that if the next block to cover is selected in this way, the translation must observe ITG constraints. For every hypothesis during decoding, the immediate left and right words of the current block face the following three conditions:

(1) The immediately left word is not covered and the immediately right word is covered, then the next block to cover must be selected from the left neighboring uncovered span, eg. for the current block $[diaocha_3\ ziliaode_4]$ in Figure 3(e). In this condition, the ITG tree can be constructed in the following two ways: either all words in the left neighboring uncovered span are translated first, then this span is merged with the current span (taking three nodes as an example, this case is shown in Figure 4(a)), or the right part of the left neighboring uncovered span is merged with the current block first, then the new block is merged with the rest part of the left neighboring uncovered span (shown in Figure 4(b)). In a word, only after all words in the left neighboring uncovered span are covered, other words can be covered.

(2) The immediately right word is not covered and the immediately left word is covered. Similarly, only after all words in the right neighboring uncovered span are covered, other words can be

[1]The words within a block are sorted by their order in the source sentence.



(a) (b)

Figure 4: The two ways that the current block is merged with its left neighboring uncovered span. The third node in the first row denotes the current block, the first and second nodes in the first row denote left and right parts of the left neighboring uncovered span, respectively.

covered.

(3) The immediately left and right words are neither covered. The next block can be selected from either the left or the right neighboring uncovered span until the immediate left or right word is covered.

The above operations can be performed recursively until the whole source sentence is merged into a single block, so the reordering observes ITG constraints.

Now, we show that translation which is not generated in the above way must violate ITG constraints.

If the next block is selected out of the neighboring uncovered spans, the current block can be neither adjacent to the last covered block nor adjacent to the selected next block, so the current block can not be merged with any block and the whole sentence can not be covered by an ITG tree. As in Figure 3(b), if the next block to cover is $[zaoqie_6]$, then $[zairu_1]$ is neither adjacent to $[diannao_5]$ nor adjacent to $[zaoqie_6]$.

We can conclude that if we select the next block from the left or right neighboring uncovered span of the current block, then the translation must observe ITG constraints.

2.2 Shift-Reduce Decoding Algorithm

In order to generate the translation with ITG constraints, the shift-reduce algorithm have to keep trace of covered blocks, left and right neighboring uncovered spans. Formally, the shift-reduce decoding algorithm uses the following three stacks:

- S_t: the stack for covered blocks. The blocks are pushed in the order that they are covered, not the order that they are in the source sentence.

- S_l : the stack for the left uncovered spans of the current block. When a block is pushed into S_t, its corresponding left neighboring uncovered span is pushed into S_l.

- S_r :the stack for the right uncovered spans of the current block. When a block is pushed into S_t, its corresponding right neighboring uncovered span is pushed into S_r.

A translation configuration is a triple $c = \langle S_t,\ S_l,\ S_r \rangle$. Given a source sentence $f = f_1, f_2, ..., f_m$, we import a virtual start word and the whole translation procedure can be seen as a sequence of transitions from c_s to c_t, where $c_s = \langle [0],\ \emptyset,\ [1, m] \rangle$ is the initial configuration, $c_t = \langle [0, m],\ \emptyset,\ \emptyset \rangle$ is the terminal configuration. The configuration for Figure 3 (e) is $\langle [0][5][1][3, 4],\ [2],\ [6] \rangle$.

We define three types of transitions from a configuration to another . Assume the current configuration $c = \langle\ [f_{t_1 1}, f_{t_1 2}]...[f_{t_k 1}, f_{t_k 2}],\ [f_{l_1 1}, f_{l_1 2}]...[f_{l_u 1}, f_{l_u 2}],\ [f_{r_v 1}, f_{r_v 2}]...[f_{r 1 1}, f_{r 1 2}]\ \rangle$, then :

- Transitions *LShift* pop the top element $[f_{l_u 1}, f_{l_u 2}]$ from S_l and select a block $[i, j]$ from $[f_{l_u 1}, f_{l_u 2}]$ to translate. In addition, they push $[i, j]$ into S_t, and if $i \neq f_{l_u 1}$, they push $[f_{l_u 1}, i - 1]$ into S_l, and if $j \neq f_{l_u 2}$, they push $[j + 1, f_{l_u 2}]$ into S_r. The precondition to operate the transition is that S_l is not null and the top span of S_l is adjacent to the top block of S_t. Formally, the precondition is $f_{l_u 2} + 1 = f_{t_k 1}$.

- Transitions *RShift* pop the top element $[f_{r_v 1}, f_{r_v 2}]$ of S_r and select a block $[i, j]$ from $[f_{r_v 1}, f_{r_v 2}]$ to translate. In addition, they push $[i, j]$ into S_t, and if $i \neq f_{r_v 1}$, they push $[f_{r_v 1}, i-1]$ into S_l, and if $j \neq f_{r_v 2}$, they push $[j + 1, f_{r_v 2}]$ into S_r. The precondition is that S_r is not null and the top span of S_r is

adjacent to the top block of S_t. Formally, the precondition is $f_{t_k 2} + 1 = f_{r_v 1}$.

- Transitions *Reduce* pop the top two blocks $[f_{t_{k-1}1}, f_{t_{k-1}2}]$, $[f_{t_k 1}, f_{t_k 2}]$ from S_t and push the merged span $[f_{t_{k-1}1}, f_{t_k 2}]$ into S_t. The precondition is that the top two blocks are adjacent. Formally, the precondition is $f_{t_{k-1}2} + 1 = f_{t_k 1}$

The transition sequence of the example in Figure 2 is listed in Figure 5. For the purpose of efficiency, transitions *Reduce* are integrated with transitions *LShift* and *RShift* in practical implementation. Before transitions *LShift* and *RShift* push $[i, j]$ into S_t, they check whether $[i, j]$ is adjacent to the top block of S_t. If so, they change the top block into the merged block directly.

In practical implementation, in order to further restrict search space, distortion limit is applied besides ITG constraints: a source phrase can be covered next only when it is ITG-legal and its distortion does not exceed distortion limit. The distortion d is calculated by $d = |start_i - end_{i-1} - 1|$, where $start_i$ is the start position of the current phrase and end_{i-1} is the last position of the last translated phrase.

Transition	S_t	S_l	S_r
	$[0]$	$\emptyset$	$[1,6]$
RShift	$[0][5]$	$[1,4]$	$[6]$
LShift	$[0][5][1]$	$\emptyset$	$[2,4][6]$
RShift	$[0][5][1][3]$	$[2]$	$[4][6]$
RShift	$[0][5][1][3][4]$	$[2]$	$[6]$
Reduce	$[0][5][1][3,4]$	$[2]$	$[6]$
LShift	$[0][5][1][3,4][2]$	$\emptyset$	$[6]$
Reduce	$[0][5][1][2,4]$	$\emptyset$	$[6]$
Reduce	$[0][5][1,4]$	$\emptyset$	$[6]$
Reduce	$[0][1,5]$	$\emptyset$	$[6]$
Reduce	$[0,5]$	$\emptyset$	$[6]$
RShift	$[0,5][6]$	$\emptyset$	$\emptyset$
Reduce	$[0,6]$	$\emptyset$	$\emptyset$

Figure 5: Transition sequence for the example in Figure 2. The top nine transitions correspond to Figure 3 (a), ... , Figure 3 (i), respectively.

3 Related Work

Galley and Manning (2008) present a hierarchical phrase reordering model aimed at improving non-local reorderings. Via the hierarchical mergence of two blocks, the orientation of long distance words can be computed. Their shift-reduce algorithm does not import ITG constraints and admits the translation violating ITG constraints.

Zens et al. (2004) introduce a left-to-right decoding algorithm with ITG constraints on the alignment template system (Och et al., 1999). Their algorithm processes candidate source phrases one by one through the whole search space and checks if the candidate phrase complies with ITG constraints. Besides, their algorithm checks validity via cover vector and does not formalize ITG structure. The shift-reduce decoding algorithm holds ITG structure via three stacks. As a result, it can offer ITG-legal spans directly and decode faster. Furthermore, with the help of ITG structure, it can be extended to syntax-based models easily.

Xiong et al. (2006) propose a BTG-based model, which uses the context to determine the orientation of two adjacent spans. It employs the cube-time CYK algorithm.

4 Experiments

We compare the shift-reduce decoder with the state-of-the-art decoder Moses (Koehn et al., 2007). The shift-reduce decoder was implemented by modifying the normal search algorithm of Moses to our shift-reduce algorithm, without cube pruning (Huang and Chiang, 2005). We retained the features of Moses: four translation features, three lexical reordering features (straight, inverted and discontinuous), linear distortion, phrase penalty, word penalty and language model, without importing any new feature. The decoding configurations used by all the decoders, including beam size, phrase table limit and so on, were the same, so the performance was compared **fairly**.

First, we will show the performance of shift-reduce algorithm on three data sets with large training data sets (Section 4.1). Then, we will analyze the performance elaborately in terms of accuracy, speed and search ability with a smaller

training data set (Section 4.2). All experiments were done on Chinese-to-English translation tasks and all results are reported with case insensitive BLEU score. Statistical significance were computed using the sign-test described in Collins et al. (Collins et al., 2005).

4.1 Performance Evaluation

We did three experiments to compare the performance of the shift-reduce decoder, Moses and the decoder with ITG constraints using cover vector (denoted as CV). [2] The shift-reduce decoder decoded with two sets of parameters: one was tuned by itself (denoted as SR) and the other was tuned by Moses (denoted as SR-same), using MERT (Och, 2003). Two searching algorithms of Moses are considered: one is the normal search algorithm without cubing pruning (denoted as Moses), the other is the search algorithm with cube pruning (denoted as Moses-cb). For all the decoders, the distortion limit was set to 6, the nbest size was set to 100 and the phrase table limit was 50.

In the first experiment, the development set is part of NIST MT06 data set including 862 sentences, the test set is NIST MT08 data set and the training data set contains 5 million sentence pairs. We used a 5-gram language model which were trained on the Xinhua and AFP portion of the Gigaword corpus. The results are shown in Table 1(a).

In the second experiment, the development data set is NIST MT02 data set and the test set is NIST MT05 data set. Language model and the training data set are the same to that of the first experiment. The result is shown in Table 1(b).

In the third experiment, the development set is China Workshop on Machine Translation 2008 data set (denoted as CWMT08) and the test set is China Workshop on Machine Translation 2007 data set (denoted as CWMT07). The training set contains 2 Million sentence pairs and the language model are a 6-gram language model trained on the Reuter corpus and English corpus. Table 1(c) gives the results.

In the above three experiments, SR decoder improves the accuracy by 1.59, 0.62, 0.8 BLEU score ($p < .01$), respectively, and improves the speed by 15%, 24%, 30%, respectively. we can see that SR can improve both the accuracy and the speed while SR-same can increase the speed significantly with a slight improvement on the accuracy. As both SR and CV decode with ITG constraints, they match each other on the accu-

[2]The decoder CV is implemented by adding the ITG constraints to Moses using the algorithm described in (Zens et al., 2004).

	NIST06	NIST08	speed
Moses	30.24	25.08	4.827
Moses-cb	30.27	23.80	1.501
CV	30.35	26.23**	4.335
SR-same	——	25.09	3.856
SR	30.47	26.67**	4.126

(a)

	NIST02	NIST05	speed
Moses	35.68	35.80	7.142
Moses-cb	35.42	35.03	1.811
CV	35.45	36.56**	6.276
SR-same	——	35.84	5.008
SR	35.99*	36.42**	5.432

(b)

	CWMT08	CWMT07	speed
Moses	27.75	25.91	3.061
Moses-cb	27.82	25.16	0.548
CV	27.71	26.58**	2.331
SR-same	——	25.97	1.988
SR	28.14*	26.71**	2.106

(c)

Table 1: Performance comparison. Moses: Moses without cube pruning, Moses-cb: Moses with cube pruning, CV: the decoder using cover vector, SR-same: the shift-reduce decoder decoding with parameters tunes by Moses, SR: the shift-reduce decoder with parameters tuned by itself. The second column stands for develop set, the third column stands for test set and speed column shows the average time (seconds) of translating one sentence in the test set. **: significance at the .01 level.

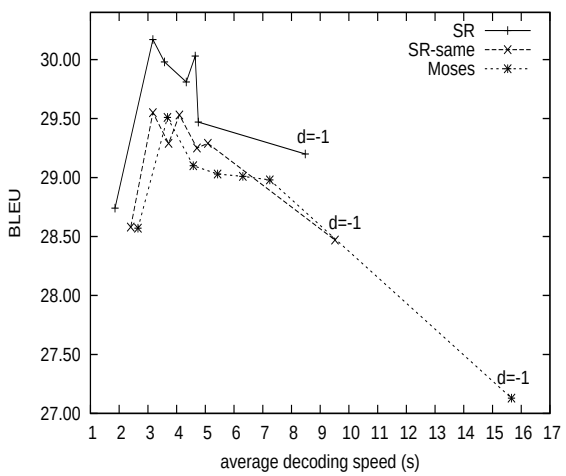


Figure 6: Performance comparison on NIST05. For a curve, the dots correspond to distortion limit $4, 6, 8, 10, 14$ and no distortion from left to right. $d = -1$ stands for no distortion limit.

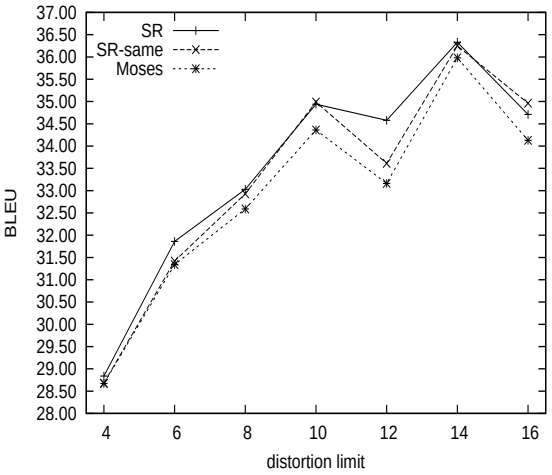
racy. However, the speed of SR is faster than CV. Cube pruning can improve decoding speed dramatically, but it is not risk-free pruning technology, so the BLEU score declines obviously.

4.2 Performance Analysis

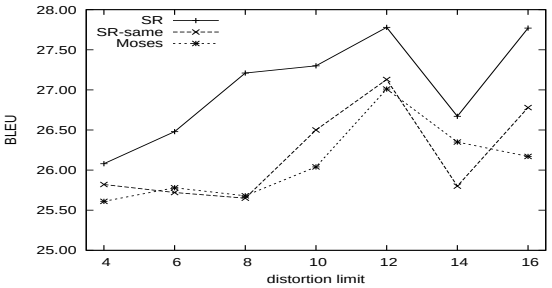
We make performance analysis with the same experiment configuration as the second experiment in Section 4.1, except that the training set in the analysis experiment is FBIS corpus, including 289k sentence pairs. In the following experiments, Moses employs the normal search algorithm without cube pruning.

For the decoders employ the linear distortion feature, the distortion limit will influence the translation accuracy. Besides, with different distortion limit, the proportion of ITG-legal translation generated by Moses will differ. The smaller the distortion limit is, the greater the proportion is. So we first compare the performance with different distortion limit.

We compare the shift-reduce decoder with Moses using different distortion limit. The results are shown in Figure 6. When distortion limit is set to 6, every decoder gets a peak value and SR has an improvement of 0.66 BLEU score over Moses. From the curves, we can see that the BLEU score of SR-same with distortion limit 8



(a) ITG set



(b) rest set

Figure 7: Accuracy comparison on the ITG set and rest set of NIST05. The ITG set includes the sentences the translations of which generated by Moses are ITG-legal, and the rest set contains the rest sentences. distortion limit $= 16$ denotes no distortion limit.

is lower than that of Mose with distortion limit 6. This is because the decoding speed of SR-same with distortion limit 8 is not faster than that of Moses with distortion limit 6. On the whole, compared to Moses, SR-same can improve the accuracy slightly with much faster decoding speed, and SR can obtain improvements on both the accuracy and the speed.

We split the test set into two sets: one contains the sentences, the translations of which generated by Moses are ITG-legal (denoted as ITG set) and the other contains the rest (denoted as rest set). From Figure 7, we can see that no matter on the ITG set or on the rest set, SR decoder can gain obvious accuracy improvements with all distortion

d	ITG						rest					
	Moses	SR-same	total	$<$	$=$	$>$	Moses	SR-same	total	$<$	$=$	$>$
4	28.67	28.68	1050	8	1042	0	25.61	25.82	32	0	0	32
6	31.34	31.42	758	51	705	2	25.78	25.72	324	32	2	290
8	32.59	32.93*	594	72	516	6	25.68	25.65	488	82	3	403
10	34.36	34.99**	456	80	365	11	26.04	26.50*	626	147	3	476
12	33.16	33.61**	454	63	380	11	27.01	27.13	628	165	1	462
14	35.98	36.25*	383	60	316	7	26.35	26.67*	699	203	1	495
-1	34.13	34.96**	351	39	308	4	26.17	26.78**	731	154	0	577

Table 2: Search ability comparison. The ITG set and the rest set of NIST05 were tested, respectively. On the ITG set, the following six factors are reported from left to right: BLEU score of Moses, BLEU score of SR-same, the number of sentences in the ITG set, the number of sentences the translation probabilities of which computed by Moses, compared to that computed by SR, is lower, equal and greater. The rest set goes similarly. *: significance at the .05 level, **: significance at the .01 level.

limit. While SR-same decoder only gets better results on the ITG set with all distortion limit. This may result from the use of the linear distortion feature. Moses may generate hypotheses the distortion of which is forbidden in the shift-reduce decoder. This especially sharpens on the rest set. So SR-same may suffer from an improper linear distortion parameter.

The search ability of Moses and the shift-reduce decoder are evaluated, too. The translation must be produced with the same set of parameters. In our experiments, we employed the parameters tuned by Moses. The test was done on the ITG and the rest set, respectively. The results are shown in Table 2. As the distortion limit becomes greater, the number of the ITG-legal translation generated by Moses becomes smaller. On the ITG set, translation probabilities from the shift-reduce decoder is either greater or equal to that from Moses on most sentences, and BLEU scores of shift-reduce decoder is greater than that of Moses with all distortion limit. Although the search space of shift-reduce decoder is smaller than that of Moses, shift-reduce decoder can give the translation that Moses can not reach. On the rest set, for most sentences, the translation probabilities from Moses is greater than that from shift-reduce decoder. But only when distortion limit is 6 and 8, the BLEU score of Moses is greater than that of the shift-reduce decoder. We may conclude that greater score does not certainly lead to greater BLEU score.

5 Conclusions and Future Work

In this paper, we present a shift-reduce decoding algorithm for phrase-based translation model that can generate the ITG-legal translation in linear time. The algorithm need not consider shift-reduce divergence and performs *reduce* operation as soon as possible. We compare the performance of the shift-reduce decoder with the state-of-the-art decoder Moses. Experiment results show that the shift-reduce algorithm can improve both the accuracy and the speed significantly on different test sets. We further analyze the performance and find that on the ITG set, the shift-reduce decoder is superior over Moses in terms of accuracy, speed and search ability, while on the rest set, it does not display advantage, suffering from improper parameters.

Next, we will extend the shift-reduce algorithm to syntax-based translation models, to see whether it works.

6 Acknowledgement

The authors were supported by National Natural Science Foundation of China Contract 60736014, National Natural Science Foundation of China Contract 60873167 and High Technology R&D Program Project No. 2006AA010108. We are grateful to the anonymous reviewers for their valuable comments.

References

Collins, Michael, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proc. of ACL*, pages 531–540.

Galley, Michel and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proc. of EMNLP*, pages 848–856.

Huang, Liang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT)*, pages 53–64.

Knight, Kevin. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25:607–615.

Koehn, Philipp, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of the 45th ACL, Demonstration Session*.

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

Och, Frans J., Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *Proc. of EMNLP*, pages 20–28.

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

Tillmann, Chirstoph and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29:97–133.

Wu, Dekai. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. of ACL*, pages 152–158.

Wu, Dekai. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23:377–403.

Xiong, Deyi, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proc. of ACL*, pages 521–528.

Zens, Richard and Hermann Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *Proc. of ACL*, pages 144–151.

Zens, Richard, Hermann Ney, Taro Watanable, and Eiichiro Sumita. 2004. Reordering constraints for phrase-based statistical machine translation. In *Proc. of COLING*, pages 205–211.