Re-ordering Source Sentences for SMT

Amit Sangodkar, Om P. Damani
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
E-mail: amits@it.iitb.ac.in, damani@cse.iitb.ac.in

Abstract

We propose a pre-processing stage for Statistical Machine Translation (SMT) systems where the words of the source sentence are re-ordered as per the syntax of the target language prior to the alignment process, so that the alignment found by the statistical system is improved. We take a dependency parse of the source sentence and linearize it as per the syntax of the target language, before it is used in either the training or the decoding phase. During this linearization, the ordering decisions among dependency nodes having a common parent are done based on two aspects: parent-child positioning and relation priority. To make the linearization process rule-driven, we assume that the relative word order of a dependency relation's relata does not depend either on the semantic properties of the relata or on the rest of the expression. We also assume that the relative word order of various relations sharing a relata does not depend on the rest of the expression. We experiment with a publicly available English-Hindi parallel corpus and show that our scheme improves the BLEU score.

Keywords: SMT, Statistical Machine Translation, Dependency Parsing, Reordering.

1. Introduction

In an end-to-end setting of machine translation system based on linguistic knowledge, we explore the applicability of Dependency Parsing (Marneffe et. al. 2006) based reordering for Statistical Machine Translation (SMT) where the words of the source sentence are re-ordered as per the syntax of the target language prior to the alignment process, so that the alignment found by the statistical machine translation system is improved. We experiment with a publicly available English-Hindi parallel corpus and show that our scheme improves the BLEU score.

A statistical machine translation system aligns the words of a source language sentence with the words in the translation in a target language sentence in a parallel corpus and builds a phrase table. It uses this phrase table to translate a new source language sentences into target language sentences, in a process called decoding. During decoding source sentence is partitioned into phrases, translation for each phrase is looked up in a phrase table, and the resulting phrase translation fragments are reordered to generate a word ordering most suitable for the target language.

The success of any machine translation system depends on how well the source language words are aligned with the target language words during the phrase table build-up and how well the reordering mechanism is able to produce a word order that resonates with the target language syntax. It is reasonable to guess that closer the syntax of the source and target language, better will be the alignment between the source and target language phrases, resulting in an improved quality of the output sentence. For example, consider the English sentence *Ram broke the window* whose Hindi translation is *Ram ne khidki todi*. In a pre-processing phase, this English sentence can be re-ordered as per Hindi syntax to *Ram the window broke*. As can be seen, the re-ordered sentence has a better word-order matching with the Hindi sentence as compared to the original English sentence. And the hope is that such a pre-processing may improve the quality of the phrase alignment in the SMT system.

Towards this goal of better alignment, we take a dependency parse of the source sentence and linearize it as per the syntax of the target language. Unlike earlier approaches discussed in Section 3, we do not perform Tree Transformation. Instead we do a transformation similar to Descending Transfer (Boitet 2003), where the parse tree on the source side is directly linearized as per the syntax of the target language. The ordering decisions among dependency nodes having a common parent are done based on two aspects: parent-child positioning and relation priority. To make the linearization process rule-driven, we assume that the relative word order of a dependency relation's relata does not depend either on the semantic properties of the relata or on the rest of the expression. We also assume that the relative word order of various relations sharing a relata does not depend on the rest of the expression.

While we have experimented with English-Hindi language pair, we believe that our approach should be tried for other language pairs where a dependency parser is available for the source language and the syntax of source and target language differs substantially.

2. System Architecture

The architecture of our system is shown in Figure 1. In this system, the words of the source sentence are reordered as per the syntax of the target language before being fed to the statistical machine translation system. This reordering takes place in following steps:

- A dependency parse (Marneffe et. al. 2006) tree of the source sentence is obtained.
- As explained in Section 4.1, the dependency parse is processed and certain nodes are collapsed.
- This modified dependency parse tree is

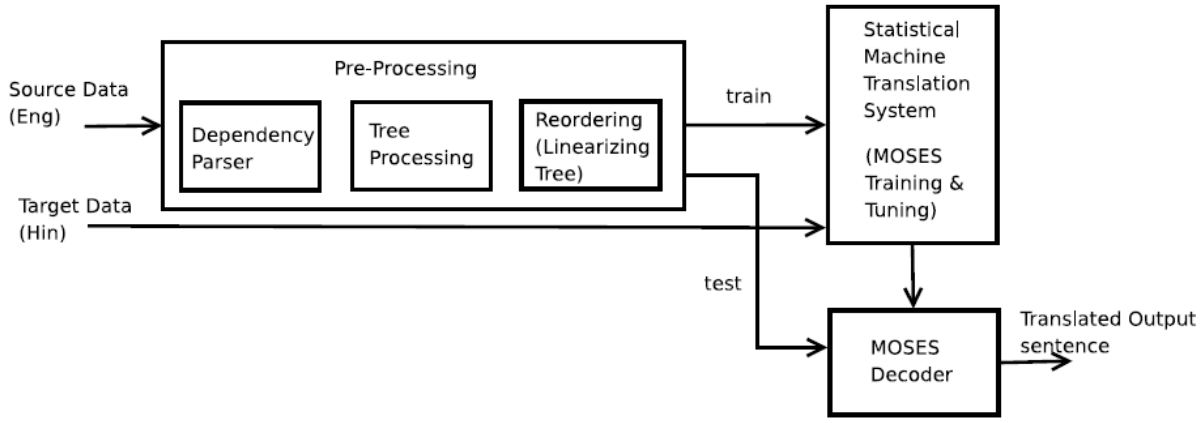

Figure 1: System Architecture

linearized as per the syntax of the target language.

The reordered sentence is then fed to the SMT system and the SMT system learns the language model and builds a phrase table based on the re-ordered sentences instead of the original sentences. We have used the Moses toolkit (Koehn et. al. 2007) for statistical machine translation and the Stanford Parser (Marneffe et. al. 2006) for the dependency parsing.

3. Related Work

The idea of reordering the source sentences as per the target language syntax is not new. Indeed, in any traditional translation system, reordering has to happen at some stage or another – an idea captured by the Vauquois triangle (Vauquois 1968; Boitet 2003), as shown in Figure 2. With the advent of statistical MT the traditional Vauquois triangle appeared to become irrelevant. However as the experience with languages with significant word-order differences grew, several researchers felt the need for syntax based reordering. The main difference between our approach and those given in (Collins et. al., 2005; Genzel, 2010; Habash, 2007; Isozaki 2010; Wang, 2007; Xia and McCord, 2004, Xu et al., 2009) is that these researchers are working on Tree Transformations – Syntactic Transfer phase in terms of the Vauquois triangle shown in Figure 2, while our approach is similar to that of Descending Transfer from Syntactic Structure to Words in the Vauquois Triangle. In traditional Vauquois triangle, the transfer happens from the source side to the target side, while in our case the transfer/reordering happens from the source side to source side only. While the work in (Collins et. al., 2005; Isozaki 2010; Wang, 2007; Xu et al., 2009) is based on manual rewrite rules, the effort in (Genzel, 2010; Habash, 2007; Xia and McCord, 2004) is focused on finding automatic tree-transformation rules.

We have used the Stanford parser for the purpose of dependency parsing. There are a number of other dependency parsers like Minipar (Sleator and Temperley 1993) and Link Parser (Lin 1998) that could also have been used in our system. These parsers differ in both, the dependency structure (which pair of words get related) and dependency typing (which is the relation for a given pair of words). Also, the granularity of the relation set is different. For instance, Link parser has a very fine-grained relation set of 106 relations whereas Stanford Parser and Minipar maintain an intermediate level of granularity with 48 and 59 relations respectively. Comparison among dependency parsers is discussed in more detail in (Marneffe et. al. 2006)`. We chose the Stanford parser since it has the right level of granularity in its dependency scheme and is a reasonably well performing parser. Also, Stanford parser is a syntactic-semantic parser i.e. relations also depict the semantics to some extent.

For a given typed dependency relation set, a number of parser optimizations can be performed. A discussion of some of the parser optimizations and other related issues can be found in (Katz-Brown 2010; Nivre 2008; Zhang and Clark 2008).

We have not separately evaluated the reordering quality but have simply evaluated its end-to-end impact in our system. Birch and Osborne (2010) have proposed LRScore, a language independent metric for evaluating the lexical and word reordering quality. For the end-to-end evaluation, we have stuck with BLEU (Papineni et. al. 2002) in this preliminary investigation and have not considered the alternatives like Meteor (Lavie and Denkowski 2009), despite being aware of its limitations (Ananthakrishnan et. al. 2007).

4. Dependency Parsing

Dependency parse is a syntactic representation expressing the binary relation between the words of a sentence. Consider the following example:

Sentence 1. *Many Bengali poets have sung songs in praise of this land.*

The dependency parse given by the Stanford Parser is:

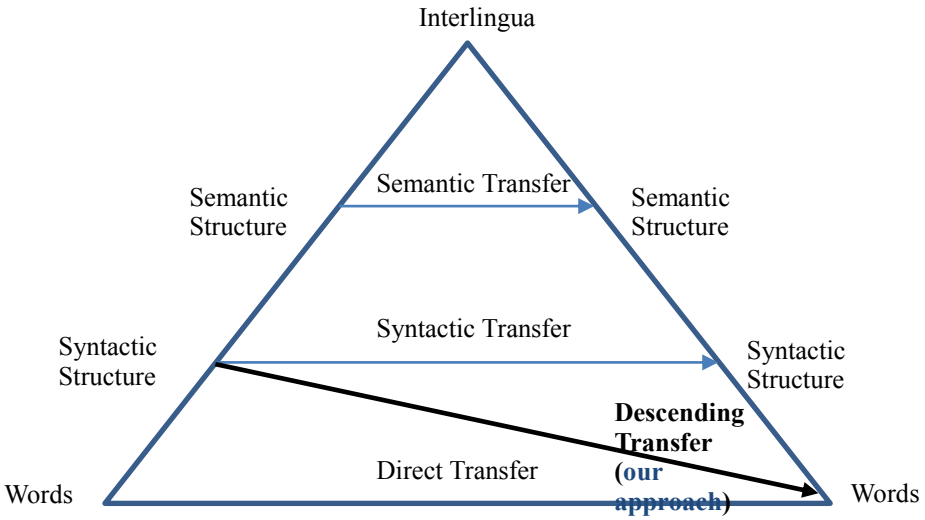

Figure 2: Vauquois triangle.

Dependency Parse:

amod (poets-3, Many-1)

nn (poets-3, Bengali-2)

nsubj (sung-5, poets-3)

aux (sung-5, have-4)

dobj (sung-5, songs-6)

prep_in (sung-5, praise-8)

det (land-11, this-10)

prep_of (praise-8, land-11)

A tree representation of the dependency parse of Sentence 1 is shown in figure 2.

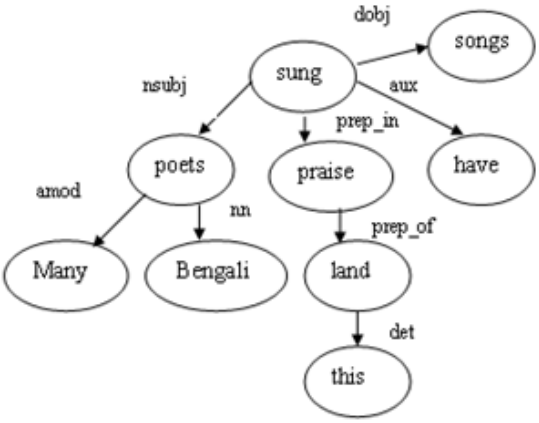

Figure 3: Dependency Parse of Sentence 1

The dependency parse of Sentence 1 consists of relations such as *nsubj (subject), dobj (direct object), amod (adjectival modifier), nn (noun-noun compound), and prep (preposition)* and so on. The detailed description of the dependency relations can be found in (Stanford Dependencies Manual, 2008). The first word of the relation is called the parent and the second word is called the child.

In Stanford Parser, there are forty-eight typed dependency relations arranged in a hierarchical manner

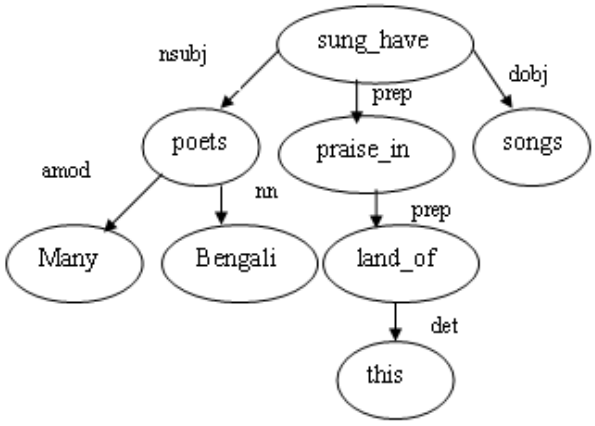

Figure 2: Modified Dependency Tree

with the most generic relation *dep* as the root. This is the *dependent* relation which is used as default when the parser fails to identify any specific relation between two semantically related words in a sentence.

We use the Stanford Parser with the *typeDependencies* option.

4.1 Dependency Tree Modification

As explained in Section 2, in our system, a dependency

Step	State
1	Stack={} Current=*sung_have* Output={}
2.1	Before-Current={*poets, praise_in, songs*}
2.2	Sorted Before-Current={*songs, praise_in, poets*}
2.3	Stack={*sung_have, songs, praise_in, poets*}
1	Stack={*sung_have, songs, praise_in*} Current=*poets*
2.1	Before-Current={*Many, Bengali*}
2.2	Sorted Before-Current={*Bengali, Many*}
2.3	Stack={*sung_have, songs, praise_in, poets, Bengali, Many*}
1	Stack={*sung_have, songs, praise_in, poets, Bengali*} Current=*Many*
3	Current=*Bengali* Output={*Many*}
3	Current=*poets* Output={*Many, Bengali*}
3	Stack={*sung_have, songs, praise_in*} Output={*Many, Bengali, poets*}
1	Stack={*sung_have, songs, praise_in*} Current=*praise_in*
2.1	Before-Current={*land_of*}
2.3	Stack={*sung_have, songs, praise_in, land_of*}
1	Stack={*sung_have, songs, praise_in*} Current=*land_of*
2.1	Before-Current={*this*}
2.3	Stack={*sung_have, songs, praise_in, land_of, this*}
1	Stack={*sung_have, songs, praise_in, land_of*} Current=*this*
3	Stack={*sung_have, songs, praise_in*} Current=*land_of* Output={*Many, Bengali, poets, this*}
3	Stack={*sung_have, songs, praise_in*} Output={*Many, Bengali, poets, this, land_of*}
1	Stack={*sung_have, songs*} Current=*praise_in*
3	Stack={*sung_have, songs*} Output={*Many, Bengali, poets, this, land_of, praise_in*}
1	Stack={*sung_have*} Current=*songs*
3	Stack={*sung_have*} Output={*Many, Bengali, poets, this, land_of, praise_in, songs*}
1	Stack={} Current=*sung_have* Output={*Many, Bengali, poets, this, land_of, praise_in, songs, sung_have*}
3	Stack={} Output={*Many, Bengali, poets, this, land_of, praise_in, songs, sung_have*}

Table 1: Re-order Algorithm execution sequence

parse of the input sentence is obtained and the dependency tree is pre-processed before being fed to reordering sub-system. We perform two types of pre-processing:

- All auxiliary verbs are removed from the tree and post-fixed to their respective main verb. Relations *aux* and *auxpass* are removed from the tree as well. So, in case of Sentence 1, relation *aux (sung-5, have-4),* is removed and "*have"* is attached to "*sung"* to form the combined unit "*sung_have"*.

- Similarly, prepositions are represented as *prep_xxx* dependency relations. The corresponding preposition is extracted and re-inserted appropriately with the parent or child. In Sentence 1, preposition "*in"* is extracted from "*prep_in"* and post-fixed to child "*praise"*. The modified tree is shown in figure 3.

- part words (*prt* relation - e.g. *shut down* - *prt(shut,down)*) are post-fixed to the parent to form a single word (*shut_down* in this case) and the *prt* relation is removed from the dependency tree.

5. Linearization

The modified dependency parse tree of the source sentence is fed to the reordering subsystem. In the reordering phase, the dependency parse tree of the source sentence is linearized as per the syntax of the target language. For graphical input like a dependency parse, syntax planning is simply a graph traversal problem. The re-ordering scheme is similar to that used in (Singh, 2007) for ordering the relations of an Interlingua called UNL (Uchida et. al., 1999). The traversal ordering decisions among dependency relations having a common parent are done based on two aspects: parent-child positioning and relation priority.

5.1 Parent-child Positioning

Some relations are such that the parent of these relations is ordered before the child and in some cases it is the other way round. Examples of the former type are *conj (conjunction), appos (apposition), advcl (adverbial clause), ccomp (clausal complement), rcmod (relative clause modifier).* For instance, in Sentence 1*,* one of the dependency relation is *prep_of (praise-8, land-11)*. In Hindi, *land* is ordered before *praise* i.e. the child is ordered before the parent for the relation *prep_of*, unlike English where the parent of *prep_of* relation is ordered before the child.

For each dependency relation we mark it as *parent-before child* or *child-before-parent*.

5.2 Prioritizing the Relations

When a parent has multiple children in the dependency parse tree, the children nodes of the parent need to be ordered. This is done by assigning a priority to each relation. Higher the priority of a relation, the corresponding child node (relata) is ordered leftmost as compared to other relatas. In dependency parse of Sentence 1, among the children of node *sung*, *nsubj* has higher priority than *dobj* and *prep_in* has a lower priority than *nsubj* but higher priority than *dobj*. As a result, child word *praise* of *prep_in* is ordered between the child word

poets of *nsubj* and child word *songs* of *dobj* in the reordered Sentence 1.r.

This pair wise priority among dependency relations can be represented in the form of a square matrix of all relations. A portion of it is shown in Table 2. An 'L' in the i^{th} row and j^{th} column means that i^{th} relation is to the *left* of the j^{th} relation in the re-ordered sentence. A '-' is present in case two relations should not be compared. It implies that the matrix is reverse symmetric.

	nsubj	dobj	Prep	amod	nn
nsubj	-	L	L	-	-
Dobj	R	-	R	-	-
Prep	R	L	-	L	L
amod	-	-	R	-	L
Nn	-	-	R	R	-

Table 2: Example Priority Matrix for pair-wise priority among a subset of dependency relations

Corpus		#sentences	#words (English)	#words (Hindi)
EILMT	Training	6500	130585	156010
	Develop.	467	9541	11419
	Testing	472	9529	11335

Table 3: EILMT datasets used for evaluation

5.3 Re-ordering Algorithm

The following algorithm does re-ordering by using the Parent-Child positioning rules and sorting on relation priorities. It is similar to that used in [Singh 2007], except that unlike UNL, dependency parse does not result in non-atomic nodes called scope node.

Initialization: Put the root node on the Stack.

Begin-Algo

While the stack is non-empty:

1. Pop the top node from the stack and make it *Current.*
2. If the *Current* node is not marked
   - 2.1 Mark the node and divide the children of current node into *Before-Current* and *After-Current* groups based on the parent-child positioning rules of the relations.
   - 2.2 Topological Sort each group in ascending order based on the pair wise priorities of the relations.
   - 2.3 Push them on the stack in *sorted-after-current, current, sorted-before-current* order.
3. Else if the node is marked, output the *Current* node.

End-Algo

Table 1 shows the sequence of execution of the algorithm for sentence 1. The final re-ordering of Sentence 1 with function words is:

Sentence 1.r (Reordered) *Many Bengali poets this land of praise in songs sung have.*

It is similar to the syntax of the corresponding Hindi sentence:

कई बंगाली कवियों ने ईस महान भूमि की प्रशंसा के गीत गाये है

"Kai kaviyon ne is mahaan bhoomi ki prashansa ke geet gaaye hai".

6. Performance Evaluation

We have evaluated the performance of our system for English to Hindi statistical machine translation.

6.1 Dataset Used

We have used the publicly available EILMT datasets provided during SMT Tools contest, International Conference on Natural Language Processing (ICON) 2008. The details of the datasets are as given in Table 3.

6.2 Experimental Setting

We have used the Moses toolkit (Koehn et. Al. 2007) as the statistical machine translation system. For Hindi language model, trigram model is used. The SRILM toolkit is used for language modelling. The Baseline system constitutes only pure Moses without any kind of pre or post processing. The SMT system is trained on the training corpus. The development corpus is used to set weights to the language model, distortion model and the phrase translation model. This process is also called tuning. For tuning, the minimum error rate training (mert) is used. Finally, the decoding is performed using Moses decoder. In case of reordering, a similar procedure is followed. The only difference is that now, the English corpus (training, tuning, and testing) is reordered before the SMT step. The English sentences are reordered based on the dependency parse of the sentence.

Initially the training corpus is cleaned. Sentences with length greater than 40, empty lines, and redundant spaces were removed. Then a 6-gram language model is learnt. After this the entire Moses tool is trained using train-factored-phrase model with alignment option as grow-diag-final and reordering option as msd-bidirectional-fe. This training is done using the original sentences.

After the training process, the tool is tuned with tuning scripts provided with Moses. Using the tuned system,

translation of development data and testing data is done. Maximum phrase length used for the decoding is seven. In the next step, the process is repeated with re-ordered English sentences instead of the original English sentences.

6.3 Experimental Results

Our results are summarized in the Table 4.

Corpus	Baseline		Re-ordered	
	Devlop.	Test	Devlop.	Test
EILMT	0.149	0.145	0.175	0.160

Table 4: BLEU scores

The BLEU score has improved from 0.145 to 0.175 using development data and the score on test data also shows improvement from 0.145 to 0.160 for EILMT data set. These results in improvement in the translation output of Moses system, using the re-ordered source language sentences, over the baseline system.

6.3.1 Sample Output

Some sample output translations from SMT system are shown. E is the English sentence, RE is the reordered English sentence, B is the hindi translation using the Baseline, R is the hindi translation using the re-ordered technique with RE as input to the SMT system.

E: these lovely pavilions were constructed by the Mughal emperor shah jahan.
RE: these lovely pavilions the Mughal emperor shah jahan by constructed were

B: ये मनोहर दर्शक दीर्घाएं द्वारा निर्मित किया गया , मुगल सम्राट , शाह विशेषित है .
R: ये मनोहर दर्शक दीर्घाएं , मुगल सम्राट , शाह जहां द्वारा बनवायी गयी थीं .

Figure 4: An example showing positive impact of reordering

E: the village has a number of interesting antique shops and cafes.
RE: the village interesting antique shops and cafes of a number has

Figure 5: An example showing negative impact of reordering

B: गांव में अनेक रूचिकर प्राचीन दुकानों और रेस्त्राओं है .
R: गांव रूचिकर , दुकानों और cafes .

6.4 Error Analysis

The improvement over baseline is lower than expected. The main source of error is the parse errors of the Dependency Parser wherein the relation among words is captured incorrectly. The small size of the parallel corpus is also the reason for low accuracy.

6.4.1 Small Corpus Size

The training corpus contains only around 6500 sentences. The problem of data sparsity is reflected in the output. So, there are several missing words which are not translated. Also, low frequency of words results in incorrect phrase table entries causing incorrect phrase translation during decoding.

6.4.2 Parsing Errors

In case the Stanford parser is unable to identify a specific relation for a particular dependency, it qualifies that relation generically as a *dep* (dependent) relation. For example, consider the output of Stanford parser for the following sentence: *the hill station is very charming in winter when the rains have stopped*. The parser gives a dependency *dep(winter, stopped)* where the correct relation is *rcmod*.

In some cases, the parser labels a dependency incorrectly. For example, consider the following sentence: *The jeep safari refreshes and revitalizes*. The parser wrongly interprets *safari* as a verb (POS=VBZ) and *refreshes*, *revitalizes* as nouns (POS=NNS) because of which all the dependencies are garbled. For instance, we get *nsubj(safari,jeep)* instead of *amod(safari,jeep)*.

To be fair, our corpus is also a peculiar corpus with many constructs that are specific to Indian English and maybe the parser needs to be trained with such a corpus before it can be expected to give good performance.

6.4.3 Other Reordering Errors

We have made many simplifying assumptions which do not always hold true in practice:

- **Fixed Priority**: Certain pair of relations do not follow a definite priority order in all cases. Since, we have assigned a definite priority order for all pairs of relations, the reordering can go wrong in these cases. Similarly, the relation precedence can also differ based on the situation. For example, *bay of bengal* and *hundreds of years* have a similar dependency parse representation. However, the order of the words as per Hindi syntax is different. The former, *bengal of bay(बंगाल की खाड़ी)*, shows a child-before-parent precedence and the latter, *hundreds of years (सैकड़ो साल)*, shows a parent-before-child precedence.
- **Ordering Relative Clauses**: Relative clauses follow a Parent-before-child precedence. For example, the sentence *An insect called stem-borer has affected watermelon.*is reordered as *insect, stem-borer called, watermelon affected has*. The relative clause, *called as stem-borer* is ordered after its parent, *insect*. Although the translation is acceptable, there is a loss of fluency. A more fluent translation would

correspond to reordering *stem-borer called insect watermelon affected has*.

Also, cases where the relative clause is better placed towards the end of the sentence or attached to the word that it is associated with, need to be determined.

- **Adverbial Clauses and Markers**: Adverbial clauses are typically ordered after the verb that they qualify. Hence, *advcl* relation obeys a parent-before-child precedence. Now, consider conditionals, such as, *if you come, then I shall eat,* where the dependency relation is *advcl(eat,come)*. The desired reordering as per Hindi syntax is *if you come, then I eat shall (*यदि तुम आओगे तो मैं खाऊँगा*)* which indicates the child *come* is ordered before parent *eat*. If we follow a parent-before-child precedence here, the reordered output would be, *then I eat shall, if you come* (तो मैं खाऊँगा यदि तुम आओगे)

  Similarly, markers (words connecting clauses like *because, since, before* etc.) are generally ordered before the parent (child-before-parent precedence). Now consider the sentence *he lived in delhi before he moved to mumbai*. Here the dependency relations of interest are *advcl(live,move)* and *mark(move,before)*.

  The desired reordering for this sentence is *he mumbai to moved before he delhi in lived (*वह मुंबई आने से पहले दिल्ली में रेहता था*)*. Here, both the exceptions hold i.e. the child of *advcl* relation *move* has to be ordered before the parent *live* and the marker *before* has to be ordered after the clause containing *move*. Such cases are not handled well in our system. With our existing relation precedence rule, reordered sentence would be, *he delhi in lived before he mumbai to moved (*वह दिल्ली में रेहता था से पहले मुंबई आने*)* Essentially this shows that the priority rules have to be lexicalized, if we want to handle all corner cases.
- **Clausal Complement relation:** The relation *ccomp* occurs in two cases: (i) *He says that you like swimming* - *ccomp(say,like*), (ii) *situated near the sea, mumbai is a nice place* - *ccomp(place,situated).*

  In (i), as per Hindi syntax, *he says that you swimming like*, *says* should be ordered before *like* - a parent-before-child precedence holds. However, in (ii), although, *parent-before-child precedence* gives an arrangement that is acceptable, the construction of the sentence is such that it consists of two separate parts whose order should be maintained in the target language, which gives an arrangement as *the sea near situated mumbai a nice place is (*समंदर के निकट स्थित मुंबई एक अच्छी जगह है*)*. So *situated* should be ordered before *place* which indicates that a child-before-parent precedence holds in this specific case.
- **Inconsistency of relative clause, conjunction, neg, cop**: These relations are applicable to both type of parents: verbs as well as nouns. In verb-parent case, they follow a order of parent-*neg-cop-partmod-conj* whereas for noun-parent, the order is parent-*rcmod-conj -neg -cop*.

  To some extent, this inconsistency can be resolved by exploiting the phrase structure parse provided the phrase structure parse denotes the relative clause or conjunctive clause as a sentential part, and the main clause (along with *neg,cop* children) as a separate sentential part. Then we can group sentential parts together, so that these parts are ordered as a single entity and the components of these parts are not mixed with other parts during the reordering process.

7 Conclusions and Future Work

We have demonstrated that reordering the source sentence as per the syntax of the target language has the potential of improving the performance of a statistical machine translation system. However, our study is limited by the performance of the dependency parser and the size of the parallel corpus. In future, this study can be extended in a number of ways:

- By employing different dependency parsers
- By training dependency parsers with a corpus of Indian English sentences as opposed to American or British English
- Using a larger corpus
- Working with different language pairs
- Using other evaluation metrics. Also, instead of doing just end-to-end evaluation, quality of reordering alone can be determined.
- Automatic determination of priority relations from a large reordered corpus.

8 Acknowledgements

This work was supported in part by the TCS sponsored project Laboratory for Intelligent Internet Research. We also wish to thank the English to Indian Languages Machine Translation (EILMT) Consortium funded by the Government of India for making the relevant datasets available.

9 References

R. Ananthakrishnan, P. Bhattacharyya, M. Sasikumar and R. M. Shah (2007). Some Issues in Automatic Evaluation of English-Hindi MT: More Blues for BLEU, *Int. Conference on Natural Language Processing (ICON)* 2007.

A. Birch and M. Osborne (2010). LRscore for evaluating lexical and reordering quality in MT. In *ACL-2010 Workshop on Statistical Machine Translation (WMT)*.

C. Boitet, (2003). « Automated Translation », *Revue française de linguistique appliquée*, 2003/2 Vol. VIII,

p. 99-121.

M. Collins, P. Koehn, and I. Ku˘cerov´a. (2005). Clause restructuring for statistical machine translation. In Proceedings of *Association for Computational Linguistics Conference (ACL)* 2005.

D. Genzel. (2010). Automatically learning source-side reordering rules for large scale machine translation. In Proceedings of the *International Conference on Computational Linguistics. (COLING)* 2010.

N. Habash. (2007). Syntactic preprocessing for statistical machine translation. *Machine Translation Summit XI*, 2007.

H. Isozaki, K. Sudoh, H. Tsukada, and K. Duh. (2010). Head finalization: A simple reordering rule for SOV languages. In *ACL-2010 Workshop on Statistical Machine Translation (WMT)*.

J. Katz-Brown, S. Petrov, R. McDonald, D. Talbot, F. Och, H. Ichikawa, M. Seno and H. Kazawa (2011). Training a Parser for Machine Translation Reordering. In Proceedings of the *Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

Philipp Koehn et. al. (2007) Moses: Open Source Toolkit for Statistical Machine Translation, In Proceedings of *Association for Computational Linguistics Conference ACL 2007*, demonstration session.

A. Lavie and M. Denkowski. (2009). The Meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3).

D. Lin. (1998). Dependency-based evaluation of MINIPAR. In Workshop on the Evaluation of Parsing Systems, Granada, Spain.

Marie-Catherine de Marneffe, Bill MacCart ney and Christopher D. Manning, (2006) Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of Lexical Resource Evaluation Conference (LREC)*. 2006.

Stanford Dependencies Manual (2008), Available at http://nlp.stanford.edu/software/dependencies_manual.pdf.

J. Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4).

K. Papineni, S. Roukos, T. Ward, and W. Zhu. (2002). BLEU: a method for automatic evaluation of machine translation. In Proceedings of *Association for Computational Linguistics Conference (ACL)* 2002.

Daniel D. Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In Third *International Workshop on Parsing Technologies.*

S. Singh, M. Dalal, V. Vachhani, P. Bhattacharyya, O. P. Damani (2007). Hindi Generation from Interlingua (UNL), *Machine Translation Summit XI*, 2007.

D. Talbot, H. Kazawa, H. Ichikawa, J. Katz-Brown, M. Seno, and F. Och. (2011). A lightweight evaluation framework for machine translation reordering. In EMNLP-2011 *Workshop on Statistical Machine Translation (WMT)*.

H. Uchida, M. Zhu, M. et al.. (1999). Universal Networking Language: A gift for a millennium. The United Nations University, Tokyo, Japan.

C. Wang. (2007). Chinese syntactic reordering for statistical machine translation. In Proceedings of the *Empirical Methods in Natural Language Processing (EMNLP),* 2007.

F. Xia and M. McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In Proceedings of the *International Conference on Computational Linguistics. (COLING)* 2004.

P. Xu, J. Kang, M. Ringgaard, and F. Och. 2009. Using a dependency parser to improve SMT for subject-objectverb languages. In Proceedings of the *North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)* 2009.

Y. Zhang and S. Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition based dependency parsing. In Proceedings of the *Empirical Methods in Natural Language Processing (EMNLP),* 2008.