A Domain-Restricted, Rule Based, English-Hindi Machine Translation System Based on Dependency Parsing

Pratik Desai, Amit Sangodkar, Om P. Damani
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
pratikdesai,amits,damani@cse.iitb.ac.in

Abstract

We present a domain-restricted rule based machine translation system based on dependency parsing. We replace the transfer phase of the classical analysis, transfer, and generation strategy with a syntax planning algorithm that directly linearizes the dependency parse of the source sentence as per the syntax of the target language. While we have built the system for English to Hindi translation, the approach can be generalized to other source languages too where a dependency parser is available.

1 Introduction

We present the design of a domain-restricted rule based machine translation system based on dependency parsing (de Marneffe et al., 2006). In contrast with the classical Analysis-Transfer-Generation model (Boitet, 2003), we combine the Transfer and Generation phases in a single Generation phase based on the Descending Transfer. Figure 1 shows how our approach contrasts with the traditional approaches. Domain restriction comes in the form of use of a domain-specific dictionary with semantic properties. Other than restricting the vocabulary, no other restriction on the lnaguage is assumed.

We uses dependency parse of the source sentence as an intermediate representation from which the target language sentence can be directly generated. In contrast with the existing transfer-based, example-based, and statistics based English-Hindi translation systems (Bharati et al., 1997; Sinha and Jain, 2003; Ananthakrishnan et al., 2006; Ramanathan et al., 2009; Chaudhury et al., 2010; Venkatapathy, 2010), the key contribution of this work is an architecture (Figure 2) and a syntax planning algorithm(Algorithm 2) that directly linearizes the dependency parse tree of the source sentence as per the syntax of the target language. Both the architecture and the syntax planning algorithm are adapted from (Singh et al., 2007). We present the results for English to Hindi translation and evaluate several dependency parsers for this task. Our approach is generalizable and can easily be adapted for translation from English to several South-Asian languages. While our word reordering rules are based on dependency relations, we also make use of the phrase structure parse to extract some of the information missing in a dependency parse.

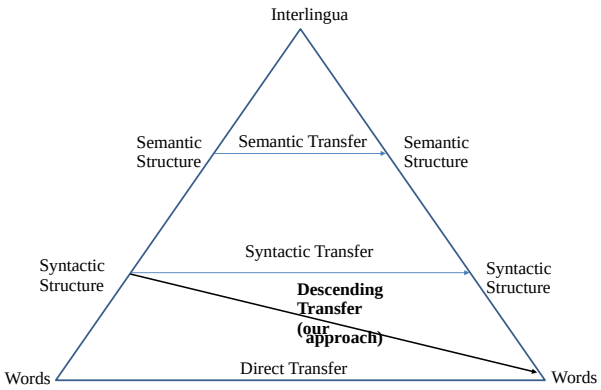

Figure 1: Situating our approach in the Vaquois Triangle

The rest of the paper is organized as follows. Section 2 gives an introduction to dependency parse representation used. Section 3 explains the architecture of our rule-based translation system using dependency parse, algorithms of various stages in language generation and the necessary resources. Section 4 presents the results of trying different dependency parsers and Section 5 contains the error analysis. Section 6 concludes the paper.

2 Dependency representation

Dependency parse represents semantic relations between words in a sentence. Dependencies are triplets containing name of the relation, parent and dependent like `relation(parent, child)`. They can be represented in the form of a graph with each edge representing the relation from a parent node to a child node. Our current system uses Stanford Dependencies Representation (de Marneffe et al., 2006) but our approach can be adapted for any dependency scheme like those used by Minipar (Lin, 1998) and Link parser (Sleator and Temperley, 1993). Consider the following example:

D S Sharma, R Sangal and J D Pawar. Proc. of the 11th Intl. Conference on Natural Language Processing, pages 177–185,
Goa, India. December 2014.

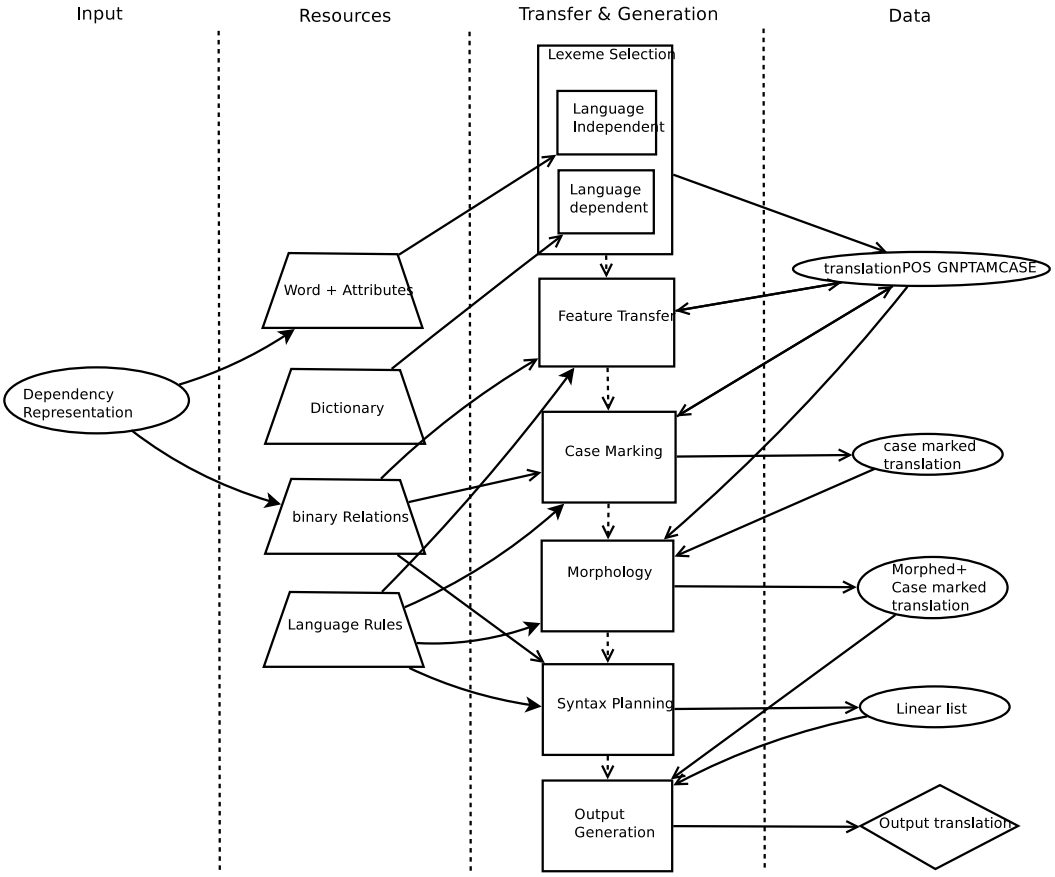

Figure 2: Generation Architecture

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

The dependency parse for the Sentence 1 given by the Stanford Parser is:

```
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)
```

The parse can be represented in a tree form as shown in Figure 3. The words in the parse tree are numbered to distinguish between different occurrences of the same word in a sentence.

In Figure 3, edge labels like `nsubj` and `dobj` are dependency relations relating two words in the sentence. The first word is called a head/parent/governor and the second word is called a child/dependent. For the dependency `nsubj(sung-5, poets-3)`, *sung* is the head and *poets* is the dependent and the two are related by `nsubj` (subject) relation.

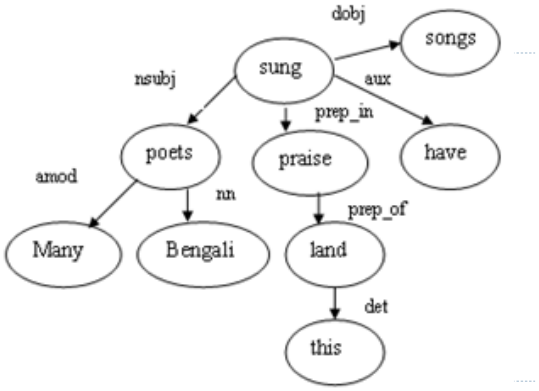

Figure 3: Dependency Tree for Sentence 1

2.1 Pre-processing the Dependency Tree

In our system, a dependency parse of the input sentence is obtained and the dependency tree is pre-processed before being fed to the generation sub-system. Following types of pre-processing are performed:

- All auxiliary verbs are removed from the tree and post-fixed to their respective main verbs. Relations *aux* and *auxpass* are removed from the tree as well. For example, in case of Sentence 1, rela-

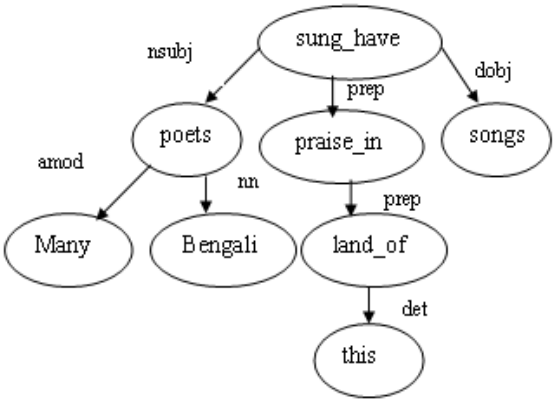

Figure 4: Modified Dependency Tree for Sentence 1

tion *aux (sung-5, have-4)*, is removed and *have* is attached to *sung* to form the combined unit *sung_have*.

- In Stanford dependency representation, prepositions are represented as *prep_xxx* dependency relations. During the pre-processing, prepositions are extracted from corresponding relations and reinserted appropriately with the parent or the child word. In Sentence 1, preposition *in* and *of* are extracted from *prep_in* and *prep_of* and post-fixed to children *praise* and *land* respectively.

- part words (*prt* relation - e.g. shut down - *prt(shut,down))* are post-fixed to the parent word to form a single word (*shut_down* in this case) and the *prt* relation is removed from the dependency tree. Similarly `nn` relation is removed and both child and parent nodes are combined to form a single node if both are proper nouns.

- For adverbial clauses, *mark* and *complm* relations are replaced with corresponding attributes. For example, in the sentence, `Forces engaged in fighting because insurgents attacked`, the dependency relations `advcl(enagaged, attacked)` and `mark(engaged, because)` are collapsed to a single relation `advcl(enagaged, attacked)` and the attribute `because` is added to parent `engaged`. This attribute helps in adding the correct function word *kyunki* during the case marking phase.

- *det* relation is removed and attribute `def` or `indef` is added to the parent node.

The modified tree for Sentence 1 is shown in Figure 4. Note that the preprocessing steps in our system are different from those in (Venkatapathy, 2010). In particular, our syntax planning algorithm (given in Section 3.5) does not require us to break cyclic dependencies.

3 Generation Architecture

Having explained the concept of dependency parsing, we now explain the the different subsystems of the generation system of Figure 2.

3.1 Lexeme selection and feature extraction

The dependency parse of the input sentence is a graphical data structure with the nodes representing the concepts and the arcs representing the dependency relations. The first stage of the translation process is the selection of the target word corresponding to each source word. Each word is looked up in the domain specific dictionary, and the corresponding lexeme is obtained. Since we are using a domain-specific dictionary, lexeme selection module is trivial in that most of the time there is only one target word for a source word. In the absence of a domain-specific dictionary, we will need a better lexeme selection module that incorporates word sense disambiguation.

We use three tools for feature extraction: Morpha (Minnen and Pearce, 2001), RelEx (Richardson et al., 2006), and Function Tagger (Blaheta and Charniak, 2000). All semantic and morphological properties of source words are not extracted by these tools and hence we assume the availability of a dictionary with various semantic and morphological attributes for each target word. Consider the following example:

Sentence 2 *This association gives training for emu-keeping and also supplies the birds.*

यह संघ इमूपालन के लिए प्रषिक्षण देता है और पक्षियों को भी उपलब्ध कराता है

`yah sangh emu-paalan ke liye prashikshan detaa hai aur pakshiyon ko bhi uplabdh karata hai`

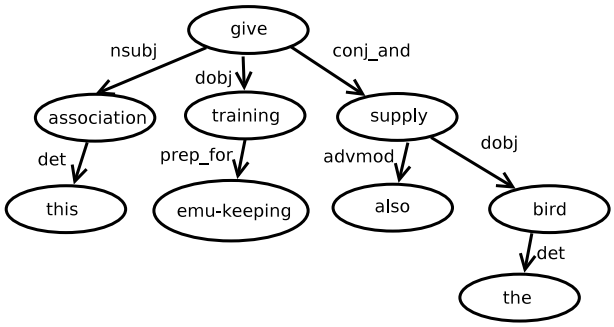

Figure 5: Dependency Tree for Sentence 2

The dependency parse for Sentence 2 is shown in Figure 5. Table 1 shows parts of the desired output of the lexeme selection and feature extraction stage for Sentence 2.

3.2 Feature transfer

This is one of the most important stage of the generation process. Attributes of the nodes are transferred

Word	Translation	Attributes/Features
give	द्	present,Verb,...
training	प्रषिक्षण	Noun,Male,Event,...
supply	उपलब्ध करा	present,Verb,Conjunct,...

Table 1: parts of the output of Lexeme Selection & Feature Extraction stage for Sentence 2

to each other in this stage. Most of the transferred attributes come from the nouns. The attributes are transferred to adjectives and verbs as follows:

- **Transferring features from nouns to adjectives** Adjectives need to take the number and gender information from the noun that it qualifies. This can be done by using the `amod` relation which has noun as parent node and adjective as child node. For example, `good boy` and `good girl` are translated to अच्छा लडका(*accha ladka*) and अच्छी लडकी(*acchi ladki*) respectively. Here, whether `good` gets translated to अच्छा(*accha*) or अच्छी(*acchi*) is dependent on the gender attribute that it gets from the noun *boy* or *girl*.

- **Transferring features from nouns to verbs** For verbs, gender and number information is obtained from the subject in case of active voice and object in the case of passive voice. For example, in Sentence 2 the gender attribute `M` is transferred from *association* to *give*.

A multi-phase algorithm is employed for transitive feature transfer. For example, for the dependency tree in Figure 5, after gender attribute `M` is transferred from *association* to *give*, in next phase, attribute `M` is transferred from *give* to *supply*.

For feature transfer, rules with the following format are used:

POS:Relation:with [p]arent or [c]hild:p+ve attr:p-ve attr:c+ve attr:c-ve attr:attr to be transferred

Here +ve/-ve attributes are those attributes which should/should not be possessed by the parent or child for the transfer to take place. For the same relation, the order of the rules application is important. For example, consider the following sequence of rules:

```
V:nsubj:c:null:null:null:null:morph
V:dobj:c:null:null:topic:null:passive
```

In the above rules, the first one checks whether a node with POS=V (verb) has the `nsubj` (subject) relation, and the second one checks for the `dobj` (object) relation. These rules are to be applied in the given order. The first rule in the above example refers to transfer of morphological attributes without any conditions. The second rule says that if the child node is the *topic* of the sentence, and has `passive` attribute, then transfer the `passive` attribute (and only this attribute) to the parent `verb`. Morphological attributes are not transferred in this case.

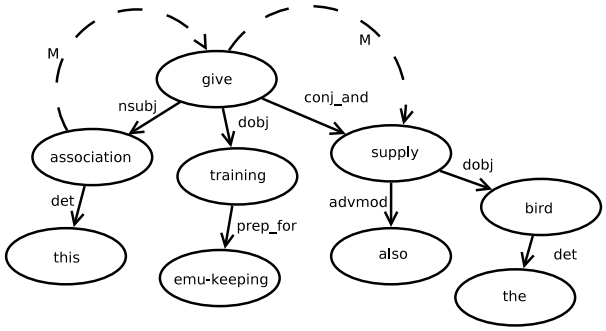

Figure 6: Feature Transfer in Dependency Parse

For Sentence 2, feature transfer is shown in Figure 6. As per the rules, child of `nsubj` relation (*association*) transfers its morphological features, 'M'(masculine) for instance, to the parent verb (*give*). Similarly, child of conjunction relation (*supply*), gets features ('M') from its parent (*give*), if the child is not governed by a subject, which is the case here.

3.3 Case marking

In this step, target language function words such as prepositions, conjunctions, clause markers e.t.c. are identified. Consider the following sentence

Sentence 3 *Ram ate rice with a spoon.*
राम ने चम्मच से खाना खाया
`ram ne chammach se khanna khaaya`

In the corresponding Hindi sentence, से*/se* is added to indicate the relation that चम्मच(*chammach/spoon*) has with the verb खाया(*khaaya/eat*) and ने*/ne* is added to indicate the relation between *Ram* and खाया(*khaaya/eat*). Case marker to be inserted is decided depending on the relation between two nodes and the lexical information present in dictionary entry of both child and parent node of the relation, and also some attributes that the parent node and child node should satisfy and some attributes that should not be satisfied. Therefore, an exhaustive list of rules is needed. Also, the rules must be ordered so that the most restrictive rule for a relation should be checked first and subsequently, lesser restrictive rules are checked.

The case (function words) depends on the dependency relation and the attributes of the two nodes involved in the relation. The case marking rules have the following format:

POS:p pre:p post:c pre:c post:p+ve attr:p-ve attr:c+ve attr:c-ve attr

`Pre/Post` are the case markers to be applied before/after the node (pre/post-positioning). +ve/-ve attributes are the attributes which a node should or should not have.

Example: For Sentence 2, function words (case markers) that are identified with dependency relations are given in Table 2.

Parent Node	Child Node	Relation	Case
give(present,V,M,sg)	*supply*(present,V,M,CJNCT)	conj_and	और
training(N,M,sg)	*emu-keeping*(N,M,oblique,sg)	prep_for	के लिए
supply(present,V,M,CJNCT)	*bird*(pl,def,N,M,oblique)	dobj	को

Table 2: Case Marking for Sentence 2

3.4 Morphology Generation

In this stage, words are inflected depending on the surrounding words. In the earlier stages, transitive feature transfer between words has already been done. Now all the inflection related information is available locally and all nouns, verbs, pronouns, and adjectives undergo inflections. Nouns and adjectives are inflected based on gender, number, and case marker information. For example, लडका(*ladka/boy*) in plural form becomes लडके(*ladke/boys*). In addition if it has a case marker like पे(*pe*), it becomes लडको(*ladko/on boys*). This case marker information is identified by the oblique attribute added to the noun *boy* during the case marking stage.

Verbs inflect depending on gender, number, person, tense, aspect and mood attributes and the voice in which it is used(active or passive). For example, *Vinod plays cricket* translates to विनोद क्रिकेट खेलता है(*Vinod kriket khelta hai*) and *Children play cricket* translates to बच्चे क्रिकेट खेलते है(*Bacche kriket khelte hai*). Here the different inflections of the verb खेल(*khel/play*) are due to the number information obtained in feature transfer stage from the qualifying noun *Vinod* or *children*.

Pronouns inflect purely based on case markers. For example, मे and case marker को are combined as मुझे.

Algorithm 1 is the Morphology generation algorithm. The procedure is recursive and the initial argument is the root of the dependency graph. Several morphology generation rules are required for each part of speech because the inflections differ dependning on the attributes.

Algorithm 1 Morphology(current)

1: **if** *current* is not marked **then**
2: mark *current*
3: **for** each unmarked parent p_i of *current* **do**
4: Morphology(p_i)
5: $pos \leftarrow$ POS of *current*
6: $attrs \leftarrow$ attribute set of *current*
7: **for** each rule r_i for pos **do**
8: $ruleAttrs \leftarrow$ attribute set of r_i
9: $match(r_i) = \frac{|attrs \cap ruleAttrs|}{|ruleAttrs|}$
10: $r = argmax(match)$
11: inflect *current* as per rules associated with r
12: **for** each child c_i of *current* **do**
13: Morphology(c_i)

Resources

Morphology of a word depends solely on the attributes of the word. The attributes obtained through dictionary look-up and during feature transfer (if any), decide the morphology on the word. For morphology of nouns, verbs and adjectives, the rule format is:

PenDel:UltDel:PenIns:UltIns:Attrs

In Hindi and many other South-Asian languages, morphological inflections apply to the last character and/or the last but one character of the word. In the above rules, <PenDel> is the character to be deleted from the penultimate position in the root word, while <UltDel> is the character to be deleted from the ultimate position. Similarly, <PenIns> and <UltIns> refer to the characters to be inserted in the penultimate and ultimate positions of the root word. <Attrs> is the attribute list. Since there are only small number of pronoun forms, the result of the combination of pronouns with case markers are pre-computed and stored. For example, pronoun मैं(*mein*), with case marker के(*ke*) gives morphed word as मेरे(*mere*).

Example: In Sentence 2, *bird* has attributes *pl*(plural) and *oblique*(the word takes case marker), which results in the morphed word पक्षियों(*pakshiyon*). Morphology example for several words are given in table 3.

Node	Attributes	Morphed Word
give	present,V,M,sg	देता है
supply	present,V,CJNCT,M	उपलब्ध कराता है
bird	pl,def,N,ANIMT,M,oblique	पक्षियों

Table 3: Results for morphology processing of Sentence 2

3.5 Syntax planning

This stage rearranges the words (nodes in the dependency tree) as per the syntax of the target language. It is this stage that determines the fluency of the obtained translation. For example, in the sentence I like apples, the words have to be reordered as I apples like, to finally get the translation मै सेब पसन्द करता हूं. Syntax planning algorithm works directly on the dependency parse of the sentence and is based on two parameters:

1. Parent-child precedence within a relation: Depending on the relation involved, parent has to be ordered before or after the child. Dependencies like conj(conjunction), appos(appos), advcl(adverbial clause) follow parent

before child order while nsubj, dobj follow child before parent order. For example, the dependencies for the sentence I like apples are:

dobj(like,apples)

nsubj(like,I)

For both these dependencies the child has to be ordered before parent to get I apples like.

2. Priority across relations: For a node with multiple children, left to right ordering of children nodes is done based on the priority given to their corresponding relations with the parent. For example, nsubj has a higher priority than dobj. Taking the previous example, I will be ordered before apples for the parent like which gives us the required word order I apples like.

Resources

For resolving parent-child precedence, all dependency relations are marked as parent-before-child or child-before-parent. For resolving the relation priority, a pair wise relation priority matrix is used. The order of the matrix equals the number of relations present. A part of the matrix for dependency relations is shown in the Table 4.

An 'L' in the i^{th} row and j^{th} column means that the child of i^{th} relation is ordered before the child of the j^{th} relation in the final translation. From Table 4, it can be seen that nsubj (subject) has higher priority than dobj (object) and prep (preposition) has a lower priority than nsubj but higher priority than dobj, so prep's child word is ordered between those of nsubj and dobj.

	nsubj	dobj	prep	amod	nn
nsubj	-	L	L	-	-
dobj	R	-	R	-	-
prep	R	L	-	L	L
amod	-	-	R	-	L
nn	-	-	R	R	-

Table 4: Relation Priority

The details of our syntax planning algorithm are given in Algorithm 2.

Example: For Sentence 2, syntax planning details are shown in Table 5. The output list at the end of the algorithm, gives the final ordering of the words in the target language syntax. For Sentence 2, the final output list is:

(this, association, emu-keeping, training, give, bird, also, supply)

This word order is as per the expected translation and the final output sentence generated is:

यह संघ इमूपालन के लिए प्रषिक्षण देता है और पक्षियों को भी उपलब्ध कराता है

yah sangh emu-paalan ke liye prashikshan detaa hai aur pakshiyon ko bhi uplabdh karata hai

Algorithm 2 Syntax Planning

Require: 1. root node placed on Stack, and, 2. all nodes unmarked

1: **while** Stack is not empty **do**
2: *current* ← POP node from Stack
3: **if** *current* is not marked **then**
4: mark *current*
5: separate unmarked relatas of *current* into $beforeCurrent$ and $afterCurrent$ lists {depending on parent-child precedence, single and multiple rules }
6: sort $beforeCurrent$ and $afterCurrent$ {depending on relation priority rules}
7: push nodes on the Stack in the order, sorted $afterCurrent$, *current*, sorted $beforeCurrent$ respectively
8: **else**
9: Output *current* node

This completes the description of our system.

4 Exploring other parsers

Parsing is the first and a very important stage of the rule based machine translation system discussed here. A wrong parse is bound to give a wrong translation and hence severely affects accuracy. It is important to have the most accurate dependency parser for the translation system for improving translation accuracy. The current system uses Stanford Parser for dependency parsing. A performance study of dependency parsers accuracies has been presented in (Cer et al., 2010). As per their results, the two parsers that rank higher than Stanford parser are CJ reranking parser (McClosky et al., 2006) and Berkley parser (Petrov and Klein, 2007). Both of these parsers can generate Stanford Typed Dependencies. Note that there are other dependencies schemes such as those used by Link parser (Sleator and Temperley, 1993) and Minipar (Lin, 1998). In (Popel et al., 2011), several dependency parsers were compared for a English-to-Czech dependency-based statistical translation system. Use of these other parsers will require rewriting of the rules in our system since they deploy different dependency schemes. Hence we restrict our comparison to CJ and Berkeley parsers since they employ Stanford Dependencies. The main difference between these three parsers is in their phrase structure parsing algorithm.

Stanford parser is an unlexicalized PCFG parser. CJ reranking parser consists of two components - a coarse-to-fine generative parser and a reranker for the parses generated from the parser. Berkeley parser uses a unlexicalized parsing with hierarchically split PCFG. All three use the same methodology for generating Stanford Dependencies from the phrase structure tree.

Step	State
1	Stack={give}
2	current={give} Stack={} output={}
5	before-current={training,association} after-current={supply}
6	sorted-before-current={association,training} after-current={supply}
7	Stack={supply,give,training,association}
2	current={association} Stack={supply,give,training}
5	before-current={this} after-current={}
7	Stack={supply,give,training,association,this}
2	current={this} Stack={supply,give,training,association}
9	output={this}
2	current={association} Stack={supply,give,training} output={this}
9	output={this, association}
2	current={training} Stack={supply,give} output={this, association}
5	before-current={emu-keeping} after-current={}
7	Stack={supply,give,training,emu-keeping}
2	current={emu-keeping} Stack={supply,give,training}
9	output={this, association, emu-keeping}
2	current={training} Stack={supply,give} output={this, association, emu-keeping}
9	output={this, association, emu-keeping, training}
2	current={give} Stack={supply}
	output={this, association, emu-keeping, training}
9	output={this, association, emu-keeping, training, give}
2	current={supply} Stack={} output={this, association, emu-keeping, training, give}
5	before-current={also,bird} after-current={}
6	sorted-before-current={bird,also}
7	Stack={supply,also,bird}
2	current={bird} Stack={supply,also}
9	output={this, association, emu-keeping, training, give, bird}
2	current={also} Stack={supply} output={this, association, emu-keeping, training, give, bird}
9	output={this, association, emu-keeping, training, give, bird, also}
2	current={supply} Stack={} output={this, association, emu-keeping, training, give, bird, also}
9	output={this, association, emu-keeping, training, give, bird, also, supply}
1	current={null} Stack={}
	output={this, association, emu-keeping, training, give, bird, also, supply}

Table 5: Syntax planning steps during the execution of Algorithm 2 for Sentence 2

4.1 Experimental Evaluation and Analysis

We use an 100 English sentences (1403 words) from agricultural domain for evaluation. The gold-standard Hindi translation contains 1231 words. These 100 sentences were chosen from 10 different discussion threads in an agricultural question-answer corpus. The average sentence length in words was 14. The input sentences were translated using Stanford typed dependencies from the three parsers. The output was evaluated against reference translations using BLEU (Papineni et al., 2002) score which ranges from 0 to 1. Despite the known limitations (Callison-Burch et al., 2006; Ananthakrishnan et al., 2007) we use BLEU, since it is the most popular translation evaluation metric. For completeness, we also evaluate *Google Translate* on the same test set.

Results in Table 6 show that for the task at hand, Stanford parser performs best and it is the only one that performs better than *Google Translate*. Note that we have used a domain-specific dictionary which also gives semantic properties of words. This plays an important part in improving the translation quality. In the absence of a domain-specific dictionary, we will need a better lexeme selection module that incorporates word sense disambiguation.

Based on the error analysis given next, we have identified several possible improvements to our system. In future, we plan to strengthen the rule base to the extent possible. We also need to strengthen the lexeme selection module by incorporating word sense disambiguation and multi-word expression identification modules. We have assumed the existence of a dictionary with various semantic attributes of target words. Automatic construction of such dictionaries from various parallel and monolingual sources is needed. We also plan to explore semantic parsers like Swirl (Surdeanu and Turmo, 2005) and Senna (Collobert et al., 2011) for determing source word attributes.

	Berkeley	**CJ reranking**	**Stanford**	**Google Translate**
BLEU	0.23	0.18	**0.27**	0.25

Table 6: Translation quality for different parsers and *Google Translate*

5 Error analysis

There are two types of errors in our system - those related to the parsers, and those related to the rest of the generation system. For parsers under considerations, dependency parse is derived from the phrase structure parse, hence parsing errors themselves can be divided into two categories:

- **Errors in the phrase structure parse** For example, in *What are its advantages?* the Stanford parser labels the phrase *are its advantages* accurately with `SQ` and gets the correct dependency `attr(are-2, what-1)`. Berkeley parser labels it inaccurately as SINV and hence generates the incorrect dependency `dep(are-2, what-1)`. CJ reranker gives wrong `POS` tag for *are* and hence generates `nsubj` instead of `attr`.
- **Errors in deriving the dependency parse** In the sentence *Also, remove all the rotten fruits and tips and destroy them*, all three parsers derive accurate phrase structure parses. But, CJ and Berkeley did not label *tip* as an `NP` and therefore they could not add label `predet` to the dependency between *fruit* and *all*.

Main errors in the rest of the system are:

1. **Limited rule base**: The current rule base of the system does not have enough coverage.
2. **Inaccurate phrase translations**: Dependency parsers give `prt`(*part of*) relation between words of some of the multi-word expressions. For rest of the phrases, system performs word by word translation. For example, phrases like *as soon as possible* end up getting translated word by word.
3. **Missing word properties**: Currently we depend on the parser and the dictionary for providing semantic properties of the words. Many semantic properties of the words are missing in the current system.

   We use RelEx and Function Tagger for obtaining word attributes. In case of Relex, all the semantic attributes are not obtained for many words due to a incomplete and small rule base, while Function tagger provides a very limited set of attributes. Other semantic parsers like Swirl (Surdeanu and Turmo, 2005) and Senna (Collobert et al., 2011) need to be explored for this. Also, for Function tagger and Relex, better mapping of the attributes given by them to the attributes needed in the rule base(eg. `LOC` given by Function tagger is mapped to `PLACE` currently) needs to be done.
4. **Lack of sense disambiguation**: The current system does not take into account the sense of the word while selecting its translation from the dictionary.
5. **Imperfect syntax planning**: The syntax planning algorithm assumes fixed parent-child precedences. However, natural languages abound in exceptions. For example, `nsubj` generally has a higher priority than `advcl`. But for translating the sentence *the accident happened before he fell unconscious*, the priority of `advcl` should be greater than `nsubj`. The priority rules need to be extended and lexicalized.

6 Conclusions

In this work, we have shown that it is feasible to make a domain-retricted rule based machine translation system based on dependency parsing by directly linearizing the dependency parse tree of the source sentence as per the syntax of the target language. We experimented with three different parsers and found that the Stanford parser is the best among the three parsers for the task at hand. We have also identified a number of issues for improvement in our system.

References

R. Ananthakrishnan, M. Kavitha, J Hegde Jayprasad, Ritesh Shah Chandra Shekhar, and Sasikumar M. Sawani Bade. 2006. Matra: A practical approach to fully-automatic indicative english-hindi machine translation. In *Symposium on Modeling and Shallow Parsing of Indian Languages (MSPIL'06)*.

R. Ananthakrishnan, Pushpak Bhattacharyya, M. Sasikumar, and Ritesh M. Shah. 2007. Some issues in automatic evaluation of english-hindi mt: More blues for bleu. In *Intl. Conf. on Natural Language Processing (ICON)*.

A. Bharati, V. Chaitanya, A. P. Kulkarni, and R. Sangal. 1997. Anusaaraka: Machine translation in stages. *A Quarterly in Artificial Intelligence, NCST, Bombay (renamed as CDAC, Mumbai)*.

Don Blaheta and Eugene Charniak. 2000. Assigning function tags to parsed text. In *1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

C. Boitet. 2003. Revue franaise de linguistique applique. *Automated Translation* , VIII:99–121.

C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Re-evaluating the role of bleu in machine translation research. In *EACL*.

Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *7th International Conference on Language Resources and Evaluation (LREC 2010)*.

S. Chaudhury, A. Rao, and D. M. Sharma. 2010. Anusaaraka: An expert system based machine translation system. In *Natural Language Processing and Knowledge Engineering (NLP-KE)*.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *5th International Conference on Language Resources and Evaluation (LREC 2006)*.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Workshop on the Evaluation of Parsing Systems*.

D. McClosky, E. Charniak, and M. Johnson. 2006. Effective self-training for parsing. In *HLT/NAACL*.

J. Carroll Minnen, G. and D. Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *40th Annual meeting of the Association for Computational Linguistics (ACL)*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *NAACL*.

Martin Popel, David Mareek, and Nathan Green. 2011. Influence of parser choice on dependency-based mt. In *EMNLP 6th Workshop on Statistical Machine Translation*.

A. Ramanathan, H. Choudhary, A. Ghosh, and P. Bhattacharyya. 2009. Case markers and morphology: Addressing the crux of the fluency problem in english-hindi smt. In *ACL-IJCNLP 2009*.

R. Richardson, B. Goertzel, H. Pinto, and E. A. Fox. 2006. Automatic creation and translation of concept maps for computer science-related theses and dissertations. In *Second Int. Conference on Concept Mapping*.

Smriti Singh, Mrugank Dalal, Vishal Vachhani, Pushpak Bhattacharyya, and Om P. Damani. 2007. Hindi generation from interlingua (unl). In *MT Summit XI*.

R. Sinha and A. Jain. 2003. Anglahindi: an english to hindi machine-aided translation system. In *MT Summit IX*.

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

M. Surdeanu and J. Turmo. 2005. Semantic role labeling using complete syntactic analysis. In *9th Conference on Computational Natural Language Learning (CoNLL)*.

Sriram Venkatapathy. 2010. *Statistical Models Suited for Machine Translation from English to Indian Languages*. Ph.D. thesis, Centre for Language Technologies Research Centre, IIIT, Hyderabad, India.