Development of a Bengali parser by cross-lingual transfer from Hindi

Ayan Das, Agnivo Saha, Sudeshna Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur, WB, India
ayan.das@cse.iitkgp.ernet.in
agnivo.saha@gmail.com
sudeshna@cse.iitkgp.ernet.in

Abstract

In recent years there has been a lot of interest in cross-lingual parsing for developing treebanks for languages with small or no annotated treebanks. In this paper, we explore the development of a cross-lingual transfer parser from Hindi to Bengali using a Hindi parser and a Hindi-Bengali parallel corpus. A parser is trained and applied to the Hindi sentences of the parallel corpus and the parse trees are projected to construct probable parse trees of the corresponding Bengali sentences. Only about 14% of these trees are complete (transferred trees contain all the target sentence words) and they are used to construct a Bengali parser. We relax the criteria of completeness to consider well-formed trees (43% of the trees) leading to an improvement. We note that the words often do not have a one-to-one mapping in the two languages but considering sentences at the chunk-level results in better correspondence between the two languages. Based on this we present a method to use chunking as a preprocessing step and do the transfer on the chunk trees. We find that about 72% of the projected parse trees of Bengali are now well-formed. The resultant parser achieves significant improvement in both Unlabeled Attachment Score (UAS) as well as Labeled Attachment Score (LAS) over the baseline word-level transferred parser.

1 Introduction

Parsing is a very important component of natural language processing. Machine learning techniques have been applied to produce highly accurate parsers for natural languages given collections of annotated parse trees called treebanks. However, creating treebank for a language involves a great deal of manual effort and treebanks do not exist for a large number of the world's languages and good quality parser learning requires a large treebank.

In recent years there have been some interesting work on developing dependency parsers where in the absence of treebanks, cross-lingual parsing has been used to develop a parser in a Target Language (TL) taking advantage of an existing parser or a treebank in a different source language (SL). Some of these systems use a parallel corpus to improve the quality of transfer parsers along with some other resources.

Though Bengali is the seventh most spoken language in the world, resources available for NLP in Bengali are scant. A small treebank consisting of about 1300 parse trees was made available for the participants of ICON 2009 (http://www.icon2009.in/) tool contest on parsing in Bengali in which 150 sentences were used for testing. We wish to explore the efficacy of cross-lingual parser transfer in Indian languages by applying it on the Hindi-Bengali language pair. Though a lot of experiments in cross-lingual parsing have been carried out in European languages, no work has been reported in Indian language pairs.

Hindi and Bengali belong to the same family of Indo-Aryan languages and share certain basic syntactic similarities. Both have the SOV sentence structure. However, there are several differences in the morphological structure of the words and phrases between these two languages.

In this paper, we refer to a transferred tree as a "complete" tree if it is connected, projective, has root aligned to the root word of the source tree and contains all the words in the target sentence. If the tree

Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing,
pages 33–43, Osaka, Japan, December 11-17 2016.

has size greater than one, satisfies all other conditions of a complete tree but does not contain all the target sentence words then it is called a "well-formed" tree. Naturally, the number of well-formed trees is much larger than the number of complete trees and it has been found that a parser trained using the well-formed trees is slightly more accurate than the parsers trained using complete trees.

We also show that due to the high-level syntactic similarities between between the Hindi and Bengali, a phrase-level transfer results in more number of well-formed parse trees (72% of the projected parse trees) than by word-level transfer (43% of the projected parse trees). The increase in number of trees also helps in developing a better parser in TL.

Though it is challenging to develop a full parser for a language, developing a shallow parser or chunker is relatively straightforward and can be done using simple rule-based or statistical methods. We use the chunkers for Bengali and Hindi along with projection of Hindi parse trees to develop a Bengali parser by phrase-level transfer. The resulting parsers improves Unlabeled Attachment Score (UAS) from 67 to 80 and Labeled Attachment Score (LAS) from 47 to 62 compared to the word level parser.

2 Related work

In this section we present the major approaches to cross-lingual syntactic transfer proposed in the literature.

Direct Transfer - Delexicalized parsing Direct transfer method learns a parser in SL and applies it to TL. A direct transfer model cannot make use of lexical features or address difference in word order. Delexicalized parsing proposed by Zeman and Resnik (2008) involves supervised training of a parser model in on a SL treebank without using any lexical features and then applying the model directly to parse sentences in TL. This was applied to Danish-Swedish pair. Søgaard (2011) used a similar method for several different language pairs and found that performance varied widely (F1-score : 50%-75%) depending upon the similarity of the language pairs. Täckström et al. (2012) used cross-lingual word clusters obtained from a large unlabelled corpora as additional features in their delexicalized parser. Naseem et al. (2012) proposed a method for multilingual learning to languages that exhibit significant differences from existing resource-rich languages which selectively learns the features relevant for a target language and ties the model parameters accordingly. Täckström et al. (2013) improved performance of delexicalized parser by incorporating selective sharing of model parameters based on typological information.

Distributed Representation Distributed representation of words as dense vector can be used to capture cross-lingual lexical information and can be augmented with delexicalized parsers. Bilingual dictionaries may be used to transfer lexical features. Xiao and Guo (2014) learnt language-independent word representations to address cross-lingual dependency parsing. Duong et al. (2015) followed a similar approach where the vectors for both the languages are learnt using a skipgram-like method in which the system was trained to predict the POS tags of the context words instead of the words themselves.

Annotation projection Cross-lingual parser transfer by annotation projection use parallel data and project parse trees in SL to TL through word alignment. But most translations are not word-to-word and only partial alignments can be obtained in many cases. Hwa et al. (2005) proposed a set of projection heuristics that make it possible to project any dependency structure through given word alignments to a target language sentence. McDonald et al. (2011) proposed a method where a delexicalized direct transfer parser trained in SL was used to parse some TL sentences which were in turn used to seed a parser in TL. The target language parser so trained was used as a lexicalized parser in the space of the target language sentences. Ma and Xia (2014) built a dependency parser by maximizing the likelihood on parallel data and the confidence on unlabeled target language data.

Rasooli and Collins (2015) proposed a method to induce dependency parser in TL using a dependency parser in SL and a parallel corpus. The transferred trees that consist of a subset of the words in the target language sentence are expanded into full trees using a decoding technique. Lacroix et al. (2016) proposed a simple alignment scheme for cross-lingual annotation projection but their performance is lower than that of Rasooli and Collins (2015).

Treebank translation Tiedemann et al. (2014) and Tiedemann (2015) proposed methods for treebank translation. They used a SMT system to obtain the phrase tables and word alignment information from the parallel corpus and used some heuristics to translate the SL treebank to a treebank of TL. They have shown that direct projection works quite well for some languages and significantly outperforms the direct delexicalized transfer model.

Parsing in Hindi and Bengali language: Bharati and Sangal (1993) and Bharati et al. (2002) are some of the first notable works on parsing of Indian languages. Nivre (2005) and Nivre (2009) have developed supervised parsers for Indian languages such as Hindi and Bengali. Some of the work in Indian language parsing use a chunk as unit instead of a word. Bharati et al. (2009) and Bharati et al. (2009) have proposed a two-stage constraint-based approach where they first try to extract the intra-chunk dependencies and resolve the inter-chunk dependencies in the second stage. Ambati et al. (2010) used disjoint sets of dependency relations and performed the intra-chunk parsing and inter-chunk parsing separately. Some of the major works on parsing in Bengali language appeared in ICON 2009 (http://www.icon2009.in/). The highest UAS and LAS for Bengali were 90.32 and 84.29 respectively.

3 Objective

We aim is to explore cross-lingual transfer parser development for Indian languages. For most Indian languages very little annotated resources are available. No annotated treebank is available in the open source for Bengali, though a 1300 sentence treebank was made available to participants of ICON 2009 tool contest. We explore methods for transfer parsing from Hindi to Bengali due to our familiarity with the languages and the Bengali language resources available with us. However we expect that this will be indicative of the type of performance between other language pairs belonging to the same family. We use a Hindi dependency treebank and a parallel Hindi-Bengali corpus to build the Bengali dependency parser by annotation projection.

We explore methods for transfer from Hindi to Bengali. Fully transferred projective trees have been found to be most useful to train a parser in the target language (Lacroix et al., 2016). To increase the amount of training data we wish to explore relaxations of this requirement so that more transferred trees can be used without negatively impacting the quality. We also wish to explore the use of other linguistic resources to improve the quality of the transferred trees.

4 Resources used

For our experiments, we used the Hindi HDTB treebank (ltrc.iiit.ac.in/treebank_H2014/) and the UDEP treebank (http://universaldependencies.org/). The HDTB treebank consists of 18637 parse trees and the Hindi UDEP treebank consists of 15870 parse trees divided into training, development and testsets. In HDTB and UDEP treebanks, Anncorra (Sharma et al., 2007) and universal dependency (McDonald et al., 2013) tagsets are used to tag the parse trees respectively. For our experiments, we used the neural network based parser (Saha and Sarkar, 2016).

The initial Hindi and Bengali word embeddings were obtained by running word2vec (Mikolov et al., 2013) on Hindi Wikipedia dump corpus and FIRE 2011 (http://www.isical.ac.in/ clia/2011/) corpus respectively.

For chunking we used the chunker developed at our institute. For testing we used the testset of 150 parse trees annotated using tagset similar to Anncorra tagset. This set of Bengali trees is the testset of the Bengali treebank used in ICON2009 (http://www.icon2009.in/) contest to train parsers for various Indian languages. The original dataset contains partially labeled parse trees with only inter-chunk dependency relations and chunk information of each sentence. We completed each parse tree by manually tagging the intra-chunk dependencies using the chunk information. We used these full trees for our experiments.

5 Proposed Method

We explore cross-lingual parser transfer by annotation projection from Hindi to Bengali by making use of a Hindi-Bengali parallel corpus. We first developed a system that does word level annotation projection

as described below.

5.1 Word level annotation projection based transfer

We use word level annotation projection to project the dependencies of the parsed Hindi sentences via the aligned parallel corpus to create a Bengali treebank on which the Bengali parser can be trained.

Word alignment of parallel corpus The parallel corpora $C_{HB} = \{(h^{(i)}, b^{(i)})\}$, where $h^{(i)}$ is a Hindi sentence and $b^{(i)}$ is the corresponding Bengali sentence, contains m parallel sentence pairs. The sentences in the parallel data were aligned in both directions using the GIZA++ tool and combined using the intersection heuristic which selects only $1:1$ alignment links. The intersect heuristic was chosen to avoid aligning a word with multiple words which might result in the formation of cycles and multiple links in the parse trees during the transfer. It results in more accurate but less number of alignments resulting in non-alignment of some Bengali words.

Annotation projection The Hindi treebank (HTB) comprise of n trees $\{(h^{(i)}, \text{tree}(h^{(i)}))\}$ where $h^{(i)}$ is a Hindi sentence and tree($h^{(i)}$) is the corresponding parse tree. Algorithm 1 outlines the steps for training the Bengali parser by word-level annotation projection method. We used the following criteria to select

Algorithm 1: Training the Bengali parser by word-level annotation projection method

input : Hindi treebank HTB, Hindi-Bengali parallel corpus C_{HB}
output: Bengali parser trained using transferred Bengali treebank

Use GIZA++ alignment tool on C_{HB} to get word-aligned sentences. For $(h^{(i)}, b^{(i)})$ get the alignment $A^{(i)} = \{(x, y)\}$, where word $h_x^{(i)}$ is aligned to word $b_y^{(i)}$.
Train a parser using the HTB to get *hindiparser*
Initialize: Bengali treebank (BTB) = NULL
for *each Hindi sentence* $h^{(i)}$ *in* C_{HB} **do**
    Parse (h_i) using *hindiparser*.
    `/* Project tree(`$h^{(i)}$`) on` $b^{(i)}$ `using` $A^{(i)}$ `to get dep(`$b^{(i)}$`) */`
    dep($b^{(i)}$) = `Project` (tree($h^{(i)}$),$b^{(i)}$,$A^{(i)}$)
    `/* Check if dep(`$b^{(i)}$`) corresponds to a well-formed tree for` $b^{(i)}$ `*/`
    If there is exactly one ROOT **AND** dep($b^{(i)}$) forms a well-formed connected tree **AND** it is projective **AND** all words $\in b^{(i)}$ appear in $dep(b^{(i)})$
    Add dep($b^{(i)}$) to BTB
end
Train a parser using BTB to get a Bengali parser *benparser*
Procedure `Project` (*tree*($h^{(i)}$), $b^{(i)}$, $A^{(i)}$)
    dep($b^{(i)}$) = NULL
    for *each dependency (head, modifier) in tree*($h^{(i)}$) **do**
        if $\exists w_1$ *: (head,* w_1*)* $\in A(i)$ ***AND*** $\exists w_2$ *: (modifier,* w_2*)* $\in A(i)$ ***AND*** $w_1 \neq w_2$ **then**
            Add (w_1, w_2) to dep($b^{(i)}$)
        end
    end
    return dep($b^{(i)}$)

complete trees:

1. The ROOT of the target tree must be mapped to the ROOT of the source tree.
2. The transferred dependency set must form a connected projective tree.
3. Every word in the Bengali sentence appears in the tree.

We find that large number of trees were eliminated due to incomplete transfer because some of the Bengali words in these sentences did not get aligned to any Hindi word. We then relax the requirement of complete trees by removing the requirement of complete trees by replacing the criterion 3 by the criterion that size of tree must be greater than 1 and making the corresponding change in Algorithm 1 to obtain the well-formed trees.

Well-formed parse trees were obtained for $21,554$ Bengali sentences, out of which 7018 were complete when HDTB treebank was used to train the Hindi parser. The percentage of fully transferred trees largely depends upon the syntactic similarities of the languages which is evident from the fact that during English to German transfer, only 2.4% of the trees were fully transferred (Rasooli and Collins, 2015).

Rasooli and Collins (2015) have shown that the inclusion of partial and incomplete trees degrades performance of the parser. In English to German parsing, the German parser trained using 18000 full trees gave an accuracy of 85.8% and a parser model trained on 968000 transferred parse trees comprising of a mixture of full and partial trees gave an accuracy of 74%. They considered trees where a subset of words forms a projective tree or a span of k words appear as modifiers. However, we observed that inclusion of well-formed partial trees (according to our criteria) along with the fully transferred trees results in increase in UAS from 66% to 67.4%. The results are shown in Table 2.

5.2 Motivation for chunk-level transfer

We hypothesize that the number of transferred trees can be increased if we can address the problem of difference in phrase structure of the two languages. The example in Section 5.2 shows how the chunk-level transfer can address the problem on non-alignment of some words due to difference in phrase structure. Thus, chunk-level transfer may significantly increase the number of transferred well-formed trees. In Table 1, we show some examples of Hindi and Bengali phrases that bring out the difference in the structure of phrases in the two languages, which means that one to one mapping between words is often not possible.

English phrase	Hindi phrase	Bengali phrase
is eating	khA rAhA hAy (eat being is)	khAchchhe (eating)
died	mare (died)	mArA jay (death happened)
due to earthquake	bhukAmp ke dwArA (earthquake of by)	bhumikamper fale (earthquake-of result)

Table 1: Example phrases with English, Hindi and Bengali equivalents

English sentence (E_1): "Several people got stuck due to landslide on way to KedarnAth".
Hindi sentence (H_1): "KedArnAth ke rAste mein bhuskhalan ke kAran bahut se log fAnse"
Bengali sentence (B_1): "KedArnAther pathe dhaser kArane bahu lok Atke pade"

The following example illustrates the word-level and chunk-level transfer of the parse tree of H_1 to the parse tree of B_1. Both H_1 and B_1 have the same meaning as that of E_1.

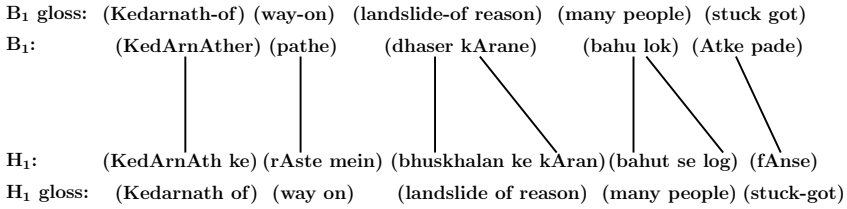

Figure 1: Word alignment between B_1 and H_1

Figure 2 shows the parse trees for B_1 and H_1, and the Bengali parse tree formed after transfer via word alignment. Note that the dependencies "Atke → pade" was not obtained in the projected tree since the

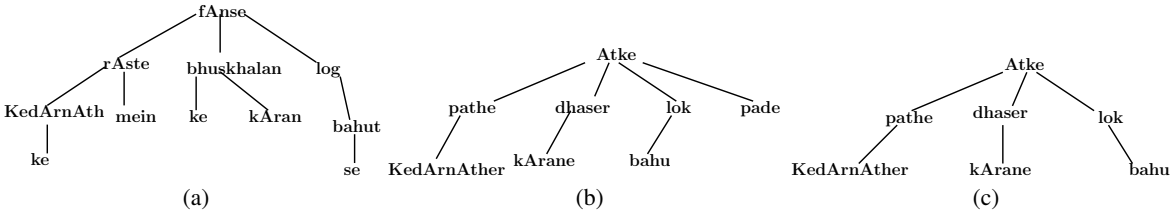

Figure 2: (a) Parse tree of H_1 (b) Parse tree of B_1 (c) Transferred Bengali word-level tree.

words "pade" was not aligned to any Hindi word. However, this problem can be eliminated by chunk-level transfer as shown below. Figure 3 shows the chunk alignment of B_1 and H_1. Each parenthesized

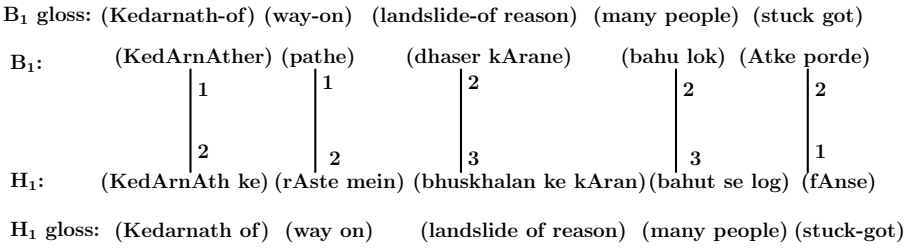

Figure 3: Chunk-level mapping between B_1 and H_1

set of words indicates a chunk. The numbers corresponding to each chunk indicate the number of words in each chunk. Figure 4a and Figure 4b show the chunk-level trees which contain only the inter-chunk

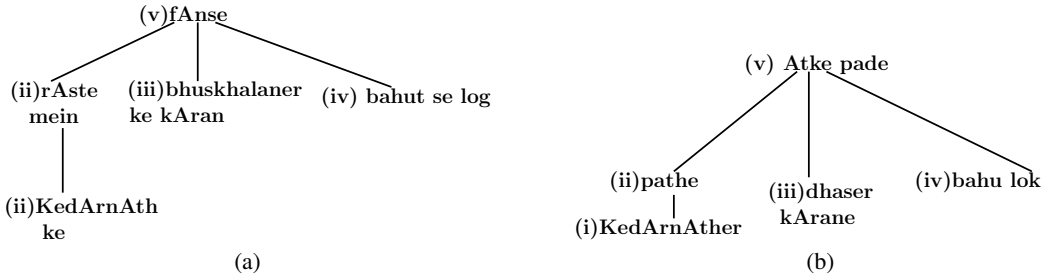

Figure 4: (a) Chunk-level parse tree of H_1 (b) Chunk-level parse tree of B_1

links. Figure 5a shows the chunk-head tree of B_1 obtained by chunk level transfer. Figure 5b shows the

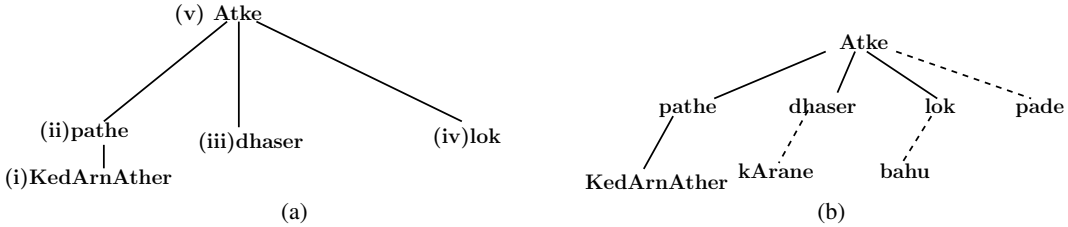

Figure 5: (a) Bengali chunk head parse tree before expansion (b) The same tree after expansion

expanded chunk-head tree containing all the words. This shows that chunk-level transfer may alleviate the problem arising due to non-alignment of some words.

5.3 Chunk-based annotation projection method

In this section we discuss the method for creating a Bengali transfer parser by our approach of chunking based cross-lingual parser transfer using annotation projection. The method is described in Algorithm 2. In step 7 of Algorithm 2 we map Hindi chunk-level trees to Bengali chunk-level trees. Note that the basic algorithm for converting the chunk-level Hindi trees to chunk-level Bengali trees using the chunk

Algorithm 2: Bengali chunk-level parser by chunk-level annotation projection method

input : Hindi treebank, word alignment of Hindi-Bengali parallel corpus
output: Bengali chunk-level parser

1 **Chunk Alignment**: Obtain chunk-alignment of the Hindi-Bengali parallel sentences (C_{HB}) from the corresponding word alignment by Procedure `ChunkAlign` $\forall$ sentence pairs $(h^{(i)}, b^{(i)}) \in C_{HB}$
2 Convert the parse trees $\{ht^{(i)}\}$ of the Hindi sentences in C_{HB} to Hindi chunk-level parse trees $\{hct^{(i)}\}$ by collapsing the chunks using the following heuristics applied to each dependency.
3 **begin**
4 If both head and modifier are chunk head, replace them by the corresponding chunk identifiers.
5 If head and modifier belongs to same chunk, ignore the dependency.
6 **end**
7 Transfer Hindi chunk-level parse trees $\{hct^{(i)}\}$ to Bengali chunk-level parse trees $\{bct^{(i)}\}$ using chunk alignment obtained in Step 1.
8 Replace the Bengali chunk identifiers in each Bengali chunk tree by the corresponding chunk heads for all trees in $\{bct^{(i)}\}$.
9 Train the Bengali parser using chunk-head trees in $\{bct^{(i)}\}$ to get the Bengali chunk-level parser.
10 **Procedure** `ChunkAlign` (*Sentence pair* (h, b)*, set of Hindi chunks* $(hcset)$ *and set of Bengali chunks* $(bcset)$*, word alignment* $\mathbf{a}^w = \{(x, y)\}$)
11 **for** *each Hindi chunk* hc_i *in hcset* **do**
12 Initialize: 1. Set of Bengali chunks to which hc_i is aligned $map(hc_i)$ = {}
13 2. Chunk alignment ($\mathbf{a}^c$) of (h, b)
14 **for** *each word* w_h *in* hc_i **do**
15 **if** w_h *aligned to a Bengali word* (w_b) *i.e.* $(w_h, w_b) \in \mathbf{a}^w$ **then**
16 Add the Bengali chunk (bc) containing w_b to $map(hc_i)$
17 **end**
18 **end**
19 **if** *all words in* hc_i *aligned to words in a single Bengali chunk* (bc_j) **then**
20 Add (hc_i, bc_j) to $\mathbf{a}^c$
21 **else if** *words in* hc_i *are aligned to multiple Bengali chunks* **then**
22 Find the chunk head $head(hc_i)$
23 **if** $head(hc_i)$ *aligned to a Bengali chunk* bc **then**
24 Add (hc_i, bc) to $\mathbf{a}^c$
25 **else**
26 No map for hc_i
27 **end**
28 **end**
29 return $\mathbf{a}^c$

alignment is same as in word-level transfer (Algorithm 1) except that chunk-level transfer uses the chunk alignment instead of the word alignment and the chunk-level trees are transferred instead of the word-level trees. From the chunk level trees we obtain the chunk-head trees by replacing the chunk identifiers with the corresponding chunk heads in step 8 of Algorithm 2. In step 9 of Algorithm 2 the chunk-head trees are used to train a chunk-level parser.

The final parser comprises of two parts, a) a chunk-level parser and b) a chunk expander. The chunk-expander uses a set of rules for intra-chunk expansion. For expanding the chunks we used the rules proposed by Kosaraju et al. (2012) as well as some additional rules. At first, the chunk-level parser is used parse the chunk-head test trees and then the chunk-expander is used to complete the intra-chunk dependency relations.

5.4 Experimental results

We performed the experiments separately using two different treebanks, HDTB and UDEP. We did not mix the two treebanks because they use different dependency relation tagset and a substantial number of sentences are common between the two treebanks. We report only the unlabeled attachment score (UAS) for our experiments when the Hindi parser used to parse the Hindi sentences of the parallel sentences was trained using the UDEP treebank because the Hindi treebank (UDEP) is tagged with Universal Dependency tagset which is different from that of the Bengali testset of 150 parse trees. We report both UAS and LAS for our experiments when the HDTB treebank was used because the tagset used in ICON and HDTB have some similarity.

Table 2 summarizes the results of the word-level and chunk-level transfer parser for the two treebanks. We observe that the number of well-formed trees obtained by chunk-level transfer have increased significantly over word-level transfer. The drop in number of complete trees in chunk-level transfer is due to the disagreement of the chunker outputs of the two languages.

It is seen that considering well-formed trees along with complete trees results in slight improvement in result and the chunk-level annotation projection method performs significantly better than the word-level annotation projection-based method for both the datasets used to train the initial Hindi parser.

Treebank used for training Hindi parser	**Method**	**Complete trees**			**Well-formed trees**		
		Number of trees	**UAS**	**LAS**	**Number of trees**	**UAS**	**LAS**
HDTB	Word-level transfer	7018	65.7	44.7	21554	67.4	47.2
	Chunk-level transfer	6679	79.3	60.1	36196	**80.6**	**62.1**
UDEP	Word-level transfer	7882	60.2	-	26827	61.0	-
	Chunk-level transfer	7061	79.1	-	37323	**79.4**	-

Table 2: Comparison of UAS and LAS of chunk-level transfer parser with word-level transfer parser when Hindi parser trained using HDTB and UDEP treebanks.

Table 3: Comparison of errors for the most frequent dependency tags. The entries of column 3 to 6 indicates the number of dependencies bearing the corresponding tags in the gold data that actually appear in the parsed trees and the accuracy (in %). Rows 2-10 (k1 to k7t) are inter-chunk dependencies and Rows 11-15 (rsym to lwg__neg) are intra-chunk dependencies

	Actual Count of dependency relations	Word-level transfer (UD)	Chunk-level transfer followed by expansion (UD)	Word-level transfer (HDTB)	Chunk-level transfer followed by expansion (HDTB)
k1 (doer/agent/subject)	166	122 (73.5)	128 (77.1)	119 (71.7)	129 (77.7)
main (root)	150	84 (56.4)	104 (69.8)	101 (67.3)	108 (72.5)
k2 (object)	131	98 (74.8)	102 (77.9)	98 (74.8)	103 (78.6)
vmod (Verb modifier)	111	68 (61.3)	74 (66.7)	83 (74.8)	87 (78.4)
r6 (possessive)	82	49 (59.8)	45 (54.9)	51 (62.2)	38 (46.3)
pof (part of)	59	54 (91.5)	58 (98.3)	57 (96.6)	59 (100)
k7p (Location in place)	50	32 (64.0)	41 (82.0)	33 (66.0)	37 (74.0)
ccof (conjunction of)	47	2 (4.25)	2 (4.26)	15 (31.9)	14 (29.8)
k7t (Location in time)	40	26 (65.0)	26 (65.0)	25 (62.5)	29 (72.5)
rsym (punctuation)	249	119 (47.8)	241 (98.4)	154 (61.8)	242 (98.8)
nmod__adj (adjectival noun modifier)	79	74 (93.7)	79 (100)	76 (96.2)	79 (100)
lwg__vaux (auxiliary verb)	54	43 (79.6)	54 (100)	52 (96.3)	54 (100)
lwg__rp (particle)	23	4 (17.4)	19 (82.6)	8 (34.8)	21 (91.3)
lwg__neg (negation)	22	6 (27.3)	21 (95.4)	3 (13.6)	22 (100)

Rasooli and Collins (2015) incrementally increased the number of full trees by completing the partial

trees using a trained arc-eager parser model. The accuracy of the English to German transfer parser model increased from 70.6% to 74.32% as completed full parse trees were incrementally added to the set. Compared to the above result our method results in an increase in UAS from 67.4 to 80.6 and 61.0 to 79.4 for HDTB and UDEP respectively.

6 Error analysis

We analyzed the errors in dependency relations of the parse trees obtained by parsing the test sentences based on the number of dependency relations in the gold data that actually appear in the trees parsed by our parser. Table 3 summarizes the accuracies of the most frequent inter-chunk and intra-chunk dependency tags. We observe that the parser trained using the HDTB treebank identifies the "conjunct of" dependencies more accurately than the parser trained using UDEP treebank due to difference in annotation scheme of Anncorra and UDEP. However, the overall performance of the transferred parsers on the "ccof" relations is poor. We need to investigate further on this issue. The possessive/genitive (r6) dependencies are better identified by word-level transferred parser. For the proper identification of possessive/genitive relations the inflectional informations are essential which can be obtained from the modifiers. In case of chunk-level transfer, we are using embeddings and features of the chunk-head only, which may not be sufficient to capture the necessary information. We also observe that the rule-based expansion of chunks helps to identify the intra-chunk relations more accurately than by word-level transfer.

From the data we observed that disagreement between the Hindi and Bengali chunkers, disagreement between Hindi chunker and parser outputs and error in word alignment are some of the major sources of error resulting in multiple links, cycles, partial trees and non-projectivity. We shall give a detailed discussion of the errors in an extended version of the paper.

7 Conclusion

This work is a basic exercise on the development of a Bengali parser without using any Bengali treebank. We have shown that a Bengali parser of fair accuracy can be developed by cross-lingual transfer from Hindi language using a Hindi treebank and a Hindi-Bengali parallel corpus. We have also shown that chunk-level transfer parser outperforms the word-level transfer parser in terms of both UAS and LAS and it increases the number of transferred well-formed trees on two different datasets.

References

Bharat Ram Ambati, Samar Husain, Sambhav Jain, Dipti Misra Sharma, and Rajeev Sangal. 2010. Two methods to incorporate 'local morphosyntactic' features in hindi dependency parsing. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 22–30, Los Angeles, CA, USA, June. Association for Computational Linguistics.

Akshar Bharati and Rajeev Sangal. 1993. Parsing free word order languages in the paninian framework. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*, ACL '93, pages 105–111, Stroudsburg, PA, USA. Association for Computational Linguistics.

Akshar Bharati, Rajeev Sangal, and T Papi Reddy. 2002. A constraint based parser using integer programming. *Proc. of ICON*.

Akshar Bharati, Samar Husain, Meher Vijay, Kalyan Deepak, Dipti Misra Sharma, and Rajeev Sangal. 2009. Constraint based hybrid approach to parsing indian languages. In *Proceedings of the 23rd PACLIC*, pages 614–621, Hong Kong, December. City University of Hong Kong.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 113–122, Beijing, China, July. Association for Computational Linguistics.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:11–311.

Prudhvi Kosaraju, Bharat Ram Ambati, Samar Husain, Dipti Misra Sharma, and Rajeev Sangal. 2012. Intra-chunk dependency annotation : Expanding hindi inter-chunk annotated treebank. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 49–56, Jeju, Republic of Korea, July. Association for Computational Linguistics.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly easy cross-lingual transfer for transition-based dependency parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California, June. Association for Computational Linguistics.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1337–1348, Baltimore, Maryland, June. Association for Computational Linguistics.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 62–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria, August. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 629–637, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre. 2005. Dependency grammar and dependency parsing. Technical report, Vxj University.

Joakim Nivre. 2009. Parsing indian languages with MaltParser. In *Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing*, pages 12–18.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338, Lisbon, Portugal, September. Association for Computational Linguistics.

Agnivo Saha and Sudeshna Sarkar. 2016. Enhancing neural network based dependency parsing using morphological information for hindi. In *17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, Konya, Turkey, April. Springer.

D.M. Sharma, Sangal R., L. Bai, R. Begam, and K. Ramakrishnamacharyulu. 2007. Anncorra : Treebanks for indian languages, annotation guidelines (manuscript).

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 477–487, Stroudsburg, PA, USA. Association for Computational Linguistics.

Oscar Täckström, Ryan T. McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 1061–1071.

Jörg Tiedemann, Željko Agić, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 130–140, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jörg Tiedemann. 2015. Improving the cross-lingual projection of syntactic dependencies. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 191–199, Vilnius, Lithuania, May. Linköping University Electronic Press, Sweden.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

D. Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. *NLP for Less Privileged Languages*, pages 35 – 35, 2008///.