Grammar Extraction from Treebanks for Hindi and Telugu

Prasanth Kolachina, Sudheer Kolachina, Anil Kumar Singh, Samar Husain, Viswanatha Naidu,Rajeev Sangal and **Akshar Bharati**

Language Technologies Research Centre,
IIIT-Hyderabad, India
{prasanth_k, sudheer.kpg08, anil, vnaidu, samar}@research.iiit.ac.in,sangal@iiit.ac.in

Abstract

Grammars play an important role in many Natural Language Processing (NLP) applications. The traditional approach to creating grammars manually, besides being labor-intensive, has several limitations. With the availability of large scale syntactically annotated treebanks, it is now possible to automatically extract an approximate grammar of a language in any of the existing formalisms from a corresponding treebank. In this paper, we present a basic approach to extract grammars from dependency treebanks of two Indian languages, Hindi and Telugu. The process of grammar extraction requires a generalization mechanism. Towards this end, we explore an approach which relies on generalization of argument structure over the verbs based on their syntactic similarity. Such a generalization counters the effect of data sparseness in the treebanks. A grammar extracted using this system can not only expand already existing knowledge bases for NLP tasks such as parsing, but also aid in the creation of grammars for languages where none exist. Further, we show that the grammar extraction process can help in identifying annotation errors and thus aid in the task of the treebank validation.

1. Introduction

Large scale annotated resources such as syntactic treebanks, PropBank, FrameNet, VerbNet, etc. have been at the core of Natural Language Processing (NLP) research for quite some time. For a language like English for which these resources were first developed, they have proved to be indispensable in advancing the state-of-art for hosts of applications. Following the success of efforts like the Penn TreeBank (PTB) (Marcus et al., 1994), Prague dependency treebank (Hajicova, 1998), several attempts are underway to build such NLP resources for new languages. One such ongoing effort is to create a treebank for Hindi-Urdu (Bhatt et al., 2009; Palmer et al., 2009; Begum et al., 2008a). Begum et al. describe a dependency annotation scheme based on the Computational Paninian Grammar or CPG (Bharati et al., 1995). The treebank being developed using this annotation scheme currently contains around 2500 sentences. Despite its modest size, the Hindi treebank has helped improve considerably the accuracies for a variety of NLP applications, especially parsing (Bharati et al., 2008).

The role of grammars in the development of advanced NLP systems is well known. Traditionally, the task of creating a grammar for a language involved selecting a formalism and encoding the patterns in that language as rules, constraints etc. But with the availability of large scale syntactically annotated treebanks, it is now possible to automatically extract an approximate grammar of a language in any of the existing formalisms from a corresponding treebank, thus reducing human effort considerably. This method of extracting grammars from treebanks allows for creation and expansion of knowledge bases for parsing. Grammars extracted through this method can be used to evaluate the coverage of existing hand-crafted grammars. The extraction process itself can help detect annotation errors. Another major advantage of extracting grammars from treebank as compared to the traditional approach of handcrafting grammars is the availability of statistical information in the form of weights associated with the primitive elements in the grammar (Xia, 2001).

One of the important issues with any kind of annotated corpora is data sparseness. Sparseness of annotated data has a detrimental effect on the performance of natural language processing applications trained over such corpora. In the case of syntactic annotation, information about the argument structure of the verb is crucial for applications such as parsing. For instance, there exist differences among individual verbs in the number of their annotated instances based on the frequency of their occurrence. The number of annotated instances greatly varies from verb to verb. In fact, the sparse data also poses a challenge for grammar extraction from treebanks. One of the ways to overcome this limitation of sparse data in syntactic treebanks is through generalization of the argument structure across different verbs. Furthermore, generalization based on clustering can lead to creation of verb classes based on the similarity of argument structure. In this paper, we present a basic system to extract a dependency grammar in the CPG formalism from treebanks for two languages, Hindi and Telugu. Towards this end, we explore an approach which relies on generalization of argument structure over verbs based on the similarity of their syntactic contexts. A grammar extracted using this system can not only expand an already existing knowledge base for NLP tasks such as parsing, but also aid in the creation of a useful resource. Further, the grammar extraction process can help in identifying annotation errors and thus make the task of the treebank validation easier.

2. Goals of the paper

The main goals of this paper are as follows:

1. To present a system that extracts grammars in the CPG formalism from the Hindi and Telugu treebanks

2. To use the extracted grammar to improve the coverage of an existing hand-crafted grammar for Hindi, which

is being used for parsing (Bharati et al., 2009a)

3. To generalize verb argument structure information over the extracted verb frames to address sparsity in the annotated corpora

4. To aid in the validation of treebanks by detecting different types of annotation errors using the extracted grammars

3. Related Work

In this section we briefly survey some of the work on grammar extraction, generalization using syntactic similarity. We also mention a few details about both the Indian language treebanks that we used. Syntactic alternation can be an important criterion while generalizing verbs. We briefly discuss how syntactic alternation in Hindi differs from English.

3.1. Grammar Extraction

The role of grammars in NLP is more extensive than is generally supposed. Xia (2000) points out that the task of treebanking for a language bears much similarity to the task of manually crafting a grammar. The treebank of a language contains an implicit grammar for that language. Statistical NLP systems trained over a treebank make use of this grammar implicit in the treebank. This is why grammar driven approaches and data driven or statistical approaches are not necessarily mutually exclusive. It is well known that the traditional approach of manually crafting a high quality, large coverage grammar takes tremendous human effort to build and maintain. In addition, the traditional approach does not provide for flexibility, consistency and generalization. To address these limitations of the traditional approach to grammar development, Xia (2001) presents two alternative approaches that generate grammars automatically, one from descriptions (LexOrg) and the other from treebanks (LexTract).

The LexTract system extracts explicit grammars in the TAG formalism from a treebank. It is not, however, limited to the TAG formalism as it can also extract CFGs from a treebank. Large scale treebanks such as the English Penn Treebank (PTB) are not based on existing grammars. Instead, they were manually annotated following the annotation guidelines. Since the process of creating annotation guidelines is similar to the process of building a grammar by hand, it can be assumed that an implicit grammar, hidden in the annotation guidelines, generates the structures in the treebank. This implicit grammar can be called a *treebank grammar*. As suggested by Xia, the task of grammar extraction using LexTract can be seen as the task of converting this implicit *treebank grammar* to an explicit TAG grammar. LexTract builds an LTAG grammar in two stages. First, it converts the annotated phrase structure trees in the PTB into LTAG derived trees. In the second stage, it decomposes these derived trees into a set of elementary trees which form the basic units of an LTAG grammar. It also extracts derivation trees which provide information about the order of operations necessary to build the corresponding derived trees. In her work, Xia has demonstrated the process for treebanks of three languages: English, Chinese and Korean. She also showed that grammars extracted using LexTract have several applications. They can be used as stand alone grammars for languages that do not have existing grammars. They can be used to enhance the coverage of already existing grammars. They can be used to compare grammars of different languages. The derivation trees extracted using LexTract can be used to train statistical parsers and taggers. LexTract can also help detect certain kinds of annotation errors and thereby, semi-automate the process of treebank validation. A major advantage of the LexTract approach to grammar development is that it can provide valuable statistical information in the form of weights associated with primitive elements.

The work we present in this paper is on the same lines as the LexTract approach to grammar development, but it is on a much smaller scale. It is meant to be the first step towards building a LexTract like system for extracting CPG grammars for Indian languages. Since we worked with dependency treebanks of Hindi and Telugu, we chose a dependency grammar formalism known as Computational Paninian grammar (CPG). In fact, the annotation guidelines followed to annotate the treebank are based on this grammar (Bharati et al., 2009b). As such, the grammar extraction process is much more straightforward than the one in LexTract. In the next section, we give a brief outline of the CPG formalism where we define the basic terminology and briefly discuss the components of a CPG grammar.

3.2. Generalization Based on Syntactic Similarity

The problem of sparse data in Propbank has been previously addressed using syntactic similarity based generalization of semantic roles across verbs (Gordon and Swanson, 2007). We try to address the data sparseness problem by generalizing over argument structure across syntactically similar verbs to arrive at an automatic verb classification. Gordon and Swanson (2007) define syntactic similarity for phrase structure trees using the notion of a parse tree path (Gildea and Jurafsky, 2002). Gildea and Jurafsky define a parse tree path as ‘the path from the target word through the parse tree to the constituent in question, represented as a string of parse tree non-terminals linked by symbols indicating upward and downward movement through the tree’. This parse tree path feature is used to represent the syntactic relationships between a predicate and its arguments in a parse tree. The syntactic context of a verb is extracted as the set of all possible parse tree paths from the parse trees of sentences containing that verb. The syntactic context of a verb is then converted into a feature vector representation. The syntactic similarity between two verbs is calculated using different distance measures such as Euclidean distance, Chi-square statistic, cosine similarity etc. In our work, we present an analogous measure of syntactic similarity for the dependency structures in the Indian Language (IL) Treebanks, which is described in section 5. We characterize the syntactic context of a verb using a *karaka* frame representation. The notion of *karakas* is explained in the next section.

3.3. Syntactic Alternations in Hindi

Syntactic alternations of a verb have been claimed to reflect its underlying semantics properties. Levin's classification of English verbs (Levin, 1993) based on this assumption demonstrates how syntactic alternation behavior of a verb can be correlated to its semantic properties thereby leading to a semantic classification. There have also been several attempts at automatically identifying distinct clusters of verbs that behave similarly using clustering algorithms. These empirically-derived clusters were then compared against Levin's classification (Merlo and Stevenson, 2001).

The following are some linguistic aspects of verb alternation behavior that we encountered in Hindi:

- In Hindi, the inchoative-transitive alternation pattern cannot be considered an alternation of the same verb stem. The verb stems in such constructions, although morphologically related, are mostly distinct. This is illustrated in the examples below:

```
Inchoative:
darawAzA        KulA
door-3PSg-Nom   open
'The door opened.'

Transitive:
Atifa-ne        darawAzA    KolA
Atif-3PSg-Erg  door-3PSg   open
'Atif opened the door.'
```

- Similarly, the diathesis alternation pattern discussed by Levin is not exhibited by Hindi verbs.

- Since Hindi is a morphologically rich, free-word order language, the alternations are not with respect to the position of the constituent as is the case in English. In Hindi, alternations are with respect to the case-endings (or the post-positions) of the nouns, which are called *vibhaktis* in CPG.

- Post-positions or *vibhaktis* alternation is determined by the form that the verb stem takes in a particular construction. In other words, the arguments of a verb are realized using different case-endings or *vibhaktis* based on the tense, aspect and modality (TAM) features of the verb. This is illustrated in the examples below:

```
abhaya          rotI  KatA hE
Abhay-Nom-3PSgM bread eat-pres.simp.-3PSgM
'Abhay eats bread.'

abhaya-ne rotI        KAyI
Abhay-Erg bread-3PSgF eat-past.simp.-3PSgF
'Abhay ate bread.'

abhaya-ne rotI-ko   KAyA
Abhay-Erg bread-Acc eat-past.simp.-default
'Abhay ate bread.'
```

In the above sentences, the nominal *vibhaktis* (case-endings or post-positions) change according to the TAM and agreement features of the verb. This co-variation of *vibhaktis* with verb's inflectional features is true not only of finite verb forms but also of non-finite verb forms. All this information is exploited in the CPG formalism in a systematic way, as discussed in the next section.

3.4. Indian Language Treebanks

In this sub-section, we give a very brief overview of the treebanks used in our work. We worked with treebanks of two Indian languages, Hindi and Telugu. The treebanks for Hindi and Telugu contain 2403 and 1226 sentences respectively. The development of these treebanks is an ongoing effort. The Hindi treebank is part of a multi-level resource development project (Bhatt et al., 2009). Some of the salient features of the annotation process employed in the development of these treebanks are as follows:

- The syntactic structure of sentences is based on the dependency representation scheme.

- Dependency relations in the Hindi treebank are annotated on top of a manually POS-tagged and chunked corpus. In the Telugu treebank, the POS-tagging and chunking was not performed manually.

- Dependency relations are defined between chunk heads.

- The dependency tagset used to annotate dependency relations is based on the CPG formalism which we discuss in section 4.

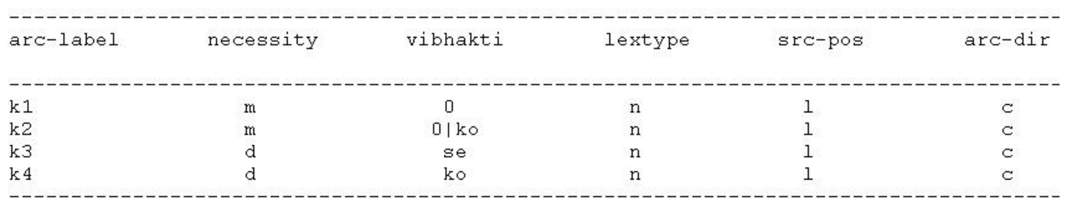
4. Computational Paninian Grammar

In this section, we give a brief overview of the Computational Paninian Grammar (CPG) formalism. We only outline details relevant to our goal of grammar extraction. See Bharati et al. (1995) for a detailed discussion of the CPG formalism and the Paninian theory on which it is based. In subsection 4.1, we introduce the basic terminology necessary for an overview of this formalism.

4.1. Terminology

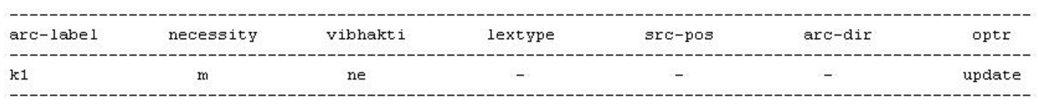
- The notion of *karaka* relations is central to Paninian Grammar. *Karaka* relations are syntactico-semantic relations between the verbs and other related constituents in a sentence. Each of the participants in an activity denoted by a verbal root is assigned a distinct *karaka*. There are six different types of *karaka* relations in the Paninian grammar as listed below:

    1. k1: *karta*, participant central to the action denoted by the verb
    2. k2: *karma*, participant central to the result of the action denoted by the verb
    3. k3: *karana*, instrument essential for the action to take place
    4. k4: *sampradana*, beneficiary/recipient of the action



arc-label	necessity	vibhakti	lextype	src-pos	arc-dir
k1	m	0	n	l	c
k2	m	0|ko	n	l	c
k3	d	se	n	l	c
k4	d	ko	n	l	c

Figure 1: Basic demand frame for the verb 'de' (to give)



arc-label	necessity	vibhakti	lextype	src-pos	arc-dir	optr
k1	m	ne	-	-	-	update

Figure 2: 'yA' transformation frame for transitive verb

5. k5: *apadana*, participant which remains stationary (or is the reference point) in an action involving separation/movement
6. k7: *adhikarana*, real or conceptual space/time[1]

For example, in the following example sentence:

```
samIrA-ne  abhaya-ko  phUla     diyA
Samira-Erg Abhay-Dat flower-Acc give.past
-----------------------------------.3PSgM
'Samira gave a flower to Abhay.'
```

Samira is the *karta* (k1), the flower is the *karma* (k2) and Abhay is the *sampradana* (k4). Similarly, in the following example:

```
Atifa ne kueM se  pAnI      nikAlA
Atif-Erg well-Abl water-Acc draw.3PSgM
'Atif drew water from the well.'
```

Atif is the *karta* (k1), well is the *apadaana* (k5) and water is the *karma* (k2).

In addition to these *karaka* relations, there are some additional relations in the Paninian scheme such as *tadarthya* (or purpose)[2].

- The notion of *vibhakti* relates to the notion of local word groups based on case ending, preposition and post-position markers. For a nominal word group, *vibhakti* is the post-position (also known as *parsarg*) occurring after the noun. Similarly, in the case of verbal word group, a head verb may be followed by auxiliary verbs which may remain as separate words or may combine with the head verb. This information following the head verb (in other words, verb stem) is collectively called the *vibhakti* of the verb. The *vibhakti* of a verb contains information about the tense, aspect and modality (TAM) and also Agreement, which are features assigned to the verb in a syntactic construction. Therefore, it can also be referred to as the TAM marker of the verb.

  In the previous example sentence, the nouns 'Atifa' and 'kuAM' have the *vibhaktis* '-ne' and '-se' respectively. The *vibhakti* of the verb 'nikAla' is 'yA' which is also its TAM label.

  Nominal *vibhaktis* have also been found to be important syntactic cues for identification of semantic role in the CPG scheme (Bharati et al., 2008).

[1]In the tagset used, k7p represents spatial location, k7t represents temporal location and k7/k7v represents conceptual location.

[2]The complete tagset can be found at `http://ltrc.iiit.ac.in/MachineTrans/research/tb/dep-tagset.pdf`

4.2. Components of CPG: Demand Frames and Transformation Frames

A key aspect of Paninian grammar (CPG) is that the verb group containing a finite verb is the most important word group (equivalent to the notion of a 'head') of a sentence. For other word groups in the sentence dependent on this head, the *vibhakti* information of the word group is used to map it to an appropriate *karaka* relation. This *karaka-vibhakti* mapping is dependent on the main verb and its TAM label. This mapping is represented by two templates: default *karaka* chart (also known as basic demand frames) and *karaka* chart transformation (also known as transformation frame). The default demand frame defines the mapping for a verb or a class of verbs with respect to a basic reference TAM label. It specifies the *karaka* relations selected by the verb along with the *vibhaktis* allowed by the basic TAM label. The basic reference TAM label in CPG is chosen to be 'tA hE' which is equivalent to Present Indefinite/Simple Present. For any other TAM label of that verb or verb class, a transformation rule is defined that can be applied to the default demand frame to obtain the appropriate *karaka-vibhakti* mapping for that TAM combination. The transformation rules can affect the default demand frames in three ways, each defined as an operation in CPG:

1. Insert: A new *karaka* relation is inserted into the demand frame along with its *vibhakti* mapping
2. Delete: An existing *karaka* relation is deleted from the default demand frame
3. Update: A *karaka-vibhakti* mapping entry in the default demand frame is updated by modifying the *vibhakti* information according to the new TAM label

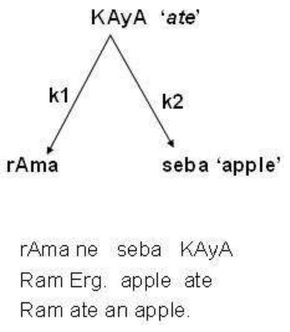


Figure 3: Dependency structure containing 'KA' (to eat)

The default demand frame and a transformation frame for the ditransitive verb 'de' (to give) and the TAM label 'yA' are shown below as an example.

5. Syntactic Similarity for Dependency Structures

A key concept in much of the previous work on semantic role labeling is the notion of a parse tree path (Gildea and Jurafsky, 2002). The parse tree path representation is based on PTB-style phrase structure (PS) trees. Gordon and Swanson (2007) define syntactic similarity between words using the parse tree path representations of their syntactic contexts.

In the case of dependency parse structures, however, the notion of a parse tree path is not required. This is because dependency structures are based on the idea that the syntactic structure of a sentence consists of binary asymmetrical modifier-modified relations between the words of that sentence. Therefore, in a dependency structure, the information about the syntactic relationship between a predicate and its arguments is trivially represented as a parent-child relationship between a head (modified) and its dependents (modifiers). For any predicate in a dependency structure, an argument frame representation which contains information about the categorial type, semantic role (in the case of labeled dependency) and other kinds of information about each of the arguments can be extracted readily from the tree. Such an argument frame representation in a dependency structure would be equivalent to the parse tree path representation for PSTs. We refer to these argument frame representations of predicates extracted from a dependency parse tree as *karaka* frame representations. Figure 3 shows the dependency structure of a sentence containing the verb 'KA' (to eat). The *karaka* frames extracted for the verb 'KA' in this parse tree are [NP k1 ne] and [NP k2 0]. These extracted *karaka* frames characterize the syntactic context of the verb 'KA' (to eat) for this sentence. The set of all *karaka* frames extracted from each sentence of a given verb characterizes the possible syntactic context of that verb. The next step is to represent this syntactic context as a feature vector. This is done simply by tabulating the frequencies of each distinct *karaka* frame into a feature vector representation. The resulting feature vectors are normalized by dividing the frequencies by the number of instances of that particular verb stem.

The syntactic similarity between two verbs is calculated as the distance between their feature vector representations using a variety of distance metrics such as Euclidean distance, Manhattan distance and cosine similarity.

6. Extracting a CPG grammar from the Treebank

In this section, we describe the various steps through which CPG grammars for Hindi and Telugu, were extracted from the treebanks.

1. The verb nodes were identified from each sentence in the treebank. For each verb node in the dependency structure of a sentence, the subtree rooted at that node is extracted. *Karaka* frames for this subtree are extracted as shown in the previous section. A *karaka* frame corresponding to each child of the verb node is obtained. An extracted *karaka* frame contains the following information about the dependent (modifier): category type, relation type, post-position. The information about the verb stem is also included in the frame. In other words, the *karaka* frames extracted for a verb node are lexicalized. At the end of this step, we have a sets of *karaka* frames corresponding to each verb instance in the treebank. We call these frames 'verb instance frames'.

2. A list of distinct verb stems based on their surface form is compiled.

3. For each distinct verb stem in the verb stem list, we take all its instances in the treebank, along with the corresponding verb instance frames extracted in 1. Two instance frames for a particular verb stem can differ from one another in one of the following ways:

    - The same *karakas* are realized in different instance frames by NPs marked with different post-positions
    - Some *karakas* which are present in some instance frames are absent in the others
    - The same *karakas* are realized in the differing instance frames by different categories of the chunks
    - Two instance frames differ in most or all of the *karakas*.

Case 1 reflects the well known phenomenon of *vibhakti* or post-position alternation for a verb trigerred by difference in the TAM marker (section 3.3.). Case 2 relates to the distinction between mandatory and non-mandatory dependents of a predicate. Case 3 corresponds to the differences in the category type of an argument for the same verb. Case 4 can be attributed solely to verb ambiguity and is the most difficult problem to address.

4. The *karaka* frames extracted for each verb from the treebank are converted into a feature vector representing the verb's syntactic context. As described in the previous section, the feature vector of a verb contains the frequencies of each distinct *karaka* frame extracted for that verb normalized by the number of occurrences of that verb. In section 8., we show how these normalized *karaka* frame frequencies can be useful for detecting annotation errors. The syntactic similarity between two verbs extracted from the treebanks is estimated as a distance between their feature vectors. At the end of this step, a similarity database containing the pair-wise syntactic similarity values for all the extracted verbs is obtained.

5. The final step in the grammar extraction process is to build Basic Demand Frames and Transformation frames (see section 4.2.) using the *karaka* frames for each extracted verb. In other words, the basic demand frame and all possible transformation frames for each verb, need to be inferred from the *karaka* frames. As can be seen from Fig 1, the basic demand frame contains information about the various participants that the verb demands and their mandatory/optional status. In order to obtain the distinction between mandatory and optional participants, we introduce a distinction between the core versus non-core *karaka* labels, which is similar to the distinction made in Framenet (Baker et al., 1998). We took the following labels in the Paninian scheme to be core: *k1*, *k1s*, *k2*, *k2p*, *k4* and *k4a* (section 4.). It must be noted that such a distinction is not defined within the CPG framework and we introduce the distinction to simplify the process of extraction. In fact, the distinction between mandatory and optional demands of a verb can be inferred from the *karaka* frames. However, this inference might not be reliable due to possibility of annotation errors in the treebanks. In the case of transformation frames, the sparseness of the annotated corpora is a problem already pointed out. Verbs differ in the number of their annotated instances. The number of transformations present for a verb will vary according to the number of its annotated instances. In the treebanks we used, there were a large number of verbs with a single instance. To counter this problem, in the case of verbs with very few annotated instances, we use the *karaka* frames of similar verbs in addition to those of the verbs to infer the transformation frames.

Language	Hindi	Telugu
Sentences	2403	1226
Verb Types	1238	391
Verb Tokens	5051	1616
Tokens per Type	4.07	4.13
Verbs with Single Instances	799	199
Verbs with Multiple Instances	439	192
Complex Predicates	934	122

Table 1: Verb statistics summary

Verb V1	verbs similar to V1	Cosine similarity
kaha (to say)	batA (to tell)	0.981
	doharA (to repeat)	0.802
	pUCa (to ask)	0.738
batA (to tell)	kaha (to say)	0.981
	doharA (to repeat)	0.799
	pUCa (to ask)	0.732
de (to give)	Beja (to send)	0.872
	sunA (cause to listen)	0.865
	sOMpa (to hand over)	0.855
mila (to meet)	A (to come)	0.795
	bIta (to elapse)	0.773
	Dala (to mould)	0.773
le (to take)	hatA (to remove)	0.950
	deKa (to see)	0.911
	sulaJA (to simplify)	0.887
A (to come)	pahuMca (to reach/arrive)	0.905
	cala (to walk)	0.903
	ruka (to stop)	0.885
banA (to make)	uTA (to lift)	0.903
	baDA (to increase)	0.897
	Coda (to leave)	0.852
jA (to go)	pahuMca (to reach/arrive)	0.886
	A (to come)	0.788
	cala (to walk)	0.685
deKa (to see)	le (to take)	0.911
	hatA (to remove)	0.855
	raKa (to keep)	0.845

Table 2: Similar Verbs for High Frequency Verbs in Hindi

7. Results

We applied the steps described in the previous section for both Hindi and Telugu treebanks. The overall summary of the verb statistics is presented in Table 1. As can be seen from the table, the number of verbs with single instance is quite high in both the treebanks. This statistic indicates the gravity of the data sparseness issue in these corpora. The number of complex predicates is also quite high in the case of Hindi. We excluded sentences with complex predicates during the grammar extraction process. Basic demand frames were inferred for 284 verbs in Hindi and 384 verbs in Telugu. In the case of transformation (TAM) frames, 81 frames were obtained for Hindi. However, due to the complex *vibhakti* alternation phenomenon in Hindi, these frames cannot be treated as definitive and require manual verification. In the case of Telugu, the process of inferring transformation (TAM) frames is comparatively simpler. Besides, the verb inflectional paradigm is comparatively

Type of error	Frequency
POS-tagging	0
Chunking	1
Morphological	3
Argument Structure	22

Table 3: Error statistics

smaller in Telugu. A total of 18 transformation (TAM) frames were extracted from the Telugu treebank.

While inferring transformation (TAM) frames for Hindi, in the case of verbs with single instance, annotated instances of similar verbs were also used for inference. In Table 2, We list 9 high frequency verbs in Hindi extracted from the treebank. For each of these verbs, we list the 3 most similar verbs obtained after applying our method for estimating syntactic similarity. We also provide the cosine similarity value for each pair. It is interesting to note that the similairty that exists among the verbs in each set listed in Table 2 is of different types. The sets of similar verbs corresponding to the first three verbs ('to say', 'to tell' and 'to give') in the table reflect a semantic similarity and can be assigned a common underlying semantic property. It is interesting that the similarity of syntactic context in the case of these verbs correlates with a semantic similarity. This is on the same lines as Levin's proposal for verb classification (Levin, 1993). However, not all sets of verbs exhibit this level of similarity. Verb sets corresponding to the last four verbs in Table 2 ('to come', 'to make', 'to go' and 'to see') are similar only with respect to their surface syntactic properties. There were also erroneous sets such as the set for the Hindi verb 'to take'.

8. Detecting Annotation Errors

The treebanks that we worked with are relatively recent developments and their validation process is still at a very early stage. The grammar extraction process that we follow can help semi-automate this process of treebank validation. The statistical information associated with each of the extracted *karaka* frames for a particular verb is helpful for detecting treebanking errors. *Karaka* frames with very low frequency are identified as containing one the following four main types of annotation errors:

- POS-tagging errors where a word is marked with the wrong POS tag. These errors are detected at various stages during the extraction.
- Chunking errors where a chunk is marked with the wrong chunk tag.
- Errors during morphological manual annotation or automatic analysis: Errors of this type include errors in identification of verb stems, TAM labels and *vibhaktis* (case–endings or post-positions) of nouns.
- Argument structure annotation errors: Errors of this type are most crucial and are difficult to detect during manual validation as the error frequency is very low.

As an example, for the intransitive verb 'jA', we show how argument structure annotation errors can be identified from the extracted frames:

```
jA
  k1   NP          0       0.347826
  k1   NULL__NP    0       0.065217
-----------------------------------
  k1s  JJP         0       0.021739
  k2   NP          ko      0.065217
```

In the above example, the relative frequency of k1s and k2 is below the threshold for error identification, whereas the total relative frequency of k1 is above that threshold. Thus, in this case, k1s and k2 are identified as annotation errors.
In Table 3, we show the relative frequencies of these four types of errors over all the extracted instances of a sample of 25 randomly selected verbs. The table shows that the number of argument structure annotation errors is much higher than other types of errors. This is not surprising given the complexity of the dependency annotation task as compared to tasks such as POS-tagging, chunking and morph analysis. Further, the argument structure errors that were discovered in the course of grammar extraction were cases of genuine confusibility even for trained annotators. This shows that such instances can be incorporated in the annotation guidelines to reduce annotation errors in the future.
Apart from the above, we also discovered a small percentage (0.3) of sentences in which no verb was found during the extraction process. The number of errors was much larger in the case of Telugu treebank as it has not yet been subjected to any kind of validation.

9. Conclusions and Future Work

We present a system that can extract grammars in the CPG formalism from dependency treebanks for Hindi and Telugu. We discuss the various issues involved in the extraction process.
In order to address the issue of data sparseness, we explore a generalization approach based on syntactic similarity of verbs. We define the notion of syntactic similarity of verbs in a dependency representation using the *karaka* frame representation. The definition is relevant for dependency representation in any formalism. Applying this syntactic similarity to the verbs extracted from the treebanks, we obtain the pair-wise similarities over the entire set of verbs. Using this similarity database, the verbs can be clustered and an unsupervised verb classification can be obtained. The resulting verb clusters can be compared against earlier works on theoretical classification of Hindi verbs into verb classes (Begum et al., 2008b) which is one of our immediate future works.
We also show how statistical information obtained during the extraction process can enable detection of different kinds of annotation errors. A detailed study of how to incorporate the information provided by the grammar extraction process into a treebank validation system is also part of our future work.
The system that we present in this paper is still under development.

10. References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Montreal, Quebec, Canada. Association for Computational Linguistics.

R. Begum, S. Husain, A. Dhwaj, D.M. Sharma, L. Bai, and R. Sangal. 2008a. Dependency annotation scheme for Indian languages. *Proceedings of International Joint Conference on Natural Language Processing*.

Rafiya Begum, Samar Husain, Lakshmi Bai, and Dipti Misra Sharma. 2008b. Developing Verb Frames for Hindi. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Morocco.

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1995. Natural Language Perspective: A Paninian Perspective.

Akshar Bharati, Samar Husain, Bharat Ambati, Sambhav Jain, Dipti Misra Sharma, and Rajeev Sangal. 2008. Two semantic features make all the difference in parsing accuracy. In *Proceedings of the 6th International Conference on Natural Language Processing (ICON-08), CDAC Pune, India*.

Akshar Bharati, Samar Husain, Dipti Misra Sharma, and Rajeev Sangal. 2009a. Two stage constraint based hybrid approach to free word order language dependency parsing. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 77–80, Paris, France.

Akshar Bharati, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begum, and Rajeev Sangal. 2009b. Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank.

Rajesh Bhatt, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Sharma, and Fei Xia. 2009. A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop, held in conjunction with ACL-IJCNLP*, Singapore.

D. Gildea and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

A. Gordon and R. Swanson. 2007. Generalizing semantic role annotations across syntactically similar verbs. In *Association for Computational Linguistics*, volume 45, page 192.

E. Hajicova. 1998. Prague Dependency Treebank: From Analytic to Tectogrammatical Annotation. In *Proceedings of TSD'98*.

Beth Levin. 1993. *English Verb Classes and Alternations*. The University of Chicago Press.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

P. Merlo and S. Stevenson. 2001. Automatic Verb Classification based on Statistical Distribution of Argument Structure. *Computational Linguistics*, 27(3):373–408.

M. Palmer, O. Rambow, R. Bhatt, D.M. Sharma, B. Narasimhan, and F. Xia. 2009. Hindi Syntax: Annotating Dependency, Lexical Predicate-Argument Structure, and Phrase Structure. In *Proceedings of ICON-2009: 7th International Conference on Natural Language Processing, Hyderabad*.

Fei Xia, Martha Palmer, and Aravind Joshi. 2000. A uniform method of grammar extraction and its applications. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora*, pages 53–62, Hong Kong.

Fei Xia. 2001. *Automatic Grammar Generation from Two Different Perspectives*. Ph.D. thesis, University of Pennsylvania.