Hindi Syntax: Annotating Dependency, Lexical Predicate-Argument Structure, and Phrase Structure

Martha Palmer
U. of Colorado
Boulder, CO 80309, USA
mpalmer@colorado.edu

Rajesh Bhatt
U. of Massachusetts
Amherst, MA 01003, USA
bhatt@linguist.umass.edu

Bhuvana Narasimhan
U. of Colorado
Boulder, CO 80309, USA
narasimb@colorado.edu

Owen Rambow
Columbia University
New York, NY 10115, USA
rambow@ccls.columbia.edu

Dipti Misra Sharma
Int'l Institute of Info. Technology
Hyderabad 500019, India
dipti@iiit.ac.in

Fei Xia
University of Washington
Seattle, WA 98195, USA
fxia@u.washington.edu

Abstract

This paper describes a treebanking project for Hindi/Urdu. We are annotating dependency syntax, lexical predicate-argument structure, and phrase structure syntax in a coordinated and partly automated manner. The paper focuses on choices in syntactic representation, and the stages we think are most appropriate for annotating differnt types of information.

1 Introduction

This paper concerns the creation of a Hindi/Urdu *multi-representational* and *multi-layered* treebank. "Multi-representational" means that we use both dependency and phrase structure for syntactic representation. "Multi-layered" means that we have different layers of representation: we represent both syntax and lexical predicate-argument structure. While multi-layered representations have become common, they are usually the effect of different projects at different points in time, each of which adds information to an existing resource; for instance, the English PropBank (Kingsbury et al., 2002) adds information to the existing Penn Treebank for English (Marcus et al., 1994). More coherent multi-layered approaches are also found in the Prague Dependency Treebank (Hajič et al., 2001), or in treebanks based on LFG (King et al., 2003) or HPSG (Oepen et al., 2002). Knowing the target layers in advance facilitates coordinated annotation design that can minimize the manual annotation effort while maximizing the linguistic information that is being represented. This paper will illustrate this process of coordinated annotation among several levels using examples.

Multi-representational treebanks are less common. While phrase structure treebanks are often converted to dependency in order to train dependency parsers (and occasionally *vice versa*), the process usually results in a treebank which itself is undocumented. The meaning of the dependency representation can only be inferred from the algorithm which was used to derive it from the phrase structure representation; there is no independent linguistic motivation and documentation of the dependency representation (as there is for the phrase-structure representation).

Having both dependency structure and phrase structure treebank for the same data enhances the utility of the resource. For example, in the development of parsers, it is becoming increasingly clear that the proper choice of representation of the syntax of a language is itself a question of parsing research. A prototypical example is the application of the Collins parser (Collins, 1997) to the Prague Dependency Treebank (Collins et al., 1999). The automatic mapping from dependency to phrase-structure was the major area of research in this effort, with important differences in parser performance resulting from changes in the mapping. Similarly, automatically changing the representation in a phrase structure treebank can also improve parsing results ((Klein and Manning, 2003) for English, (Levy and Manning, 2003) for Chinese, (Kulick et al., 2006) for Arabic). And finally, there is increasing interest in the use of dependency parses in NLP applications, as they are considered to be simpler structures which can be computed more rapidly and are closer to the kinds of semantic representations that applications can make immediate use of (e.g., (McDonald et al., 2005) and the CoNLL 2006 Shared Task). We will illustrate several syntactic analyses using dependency and phrase structure, and describe the process of producing these annotations.

We will also have two types of semantic role annotation: karaka (see Section 3.1) and PropBank (Section 3.2). The natural semantic role annotation for Hindi is, of course, karaka, with its strong basis in Paninian tradition. However, by mapping it to PropBank as well, we will facilitate alignment to parallel data with PropBank annotations, and support machine translation efforts based on parallel PropBanks (Choi et al., 2009).

2 Two Kinds of Syntactic Structure

Two different approaches to describing syntactic structure, dependency structure (DS) and phrase structure (PS), have in a sense divided the field in two, with parallel efforts on both sides. Formally, in a PS tree, all and only the leaf nodes are labeled with words from the sentence (or empty categories), while the interior nodes are labeled with nonterminal labels. In a dependency tree, all nodes are labeled with words from the sentence (or with empty categories that stand for words which are phonologically null for various reasons). Linguistically, a PS groups consecutive words hierarchically into phrases (or constituents), where each phrase is marked with a phrase marker. In a DS, syntactic dependency (i.e., the relation between a syntactic head and its arguments and adjuncts) is the primary syntactic relation represented. The notion of constituent is only derived.

In a dependency representation, a node stands for itself, for the lexical category (or *preterminal*) spanning only the word itself (e.g., N), and for its maximal projection spanning the node and all words in the subtree it anchors (e.g., NP). Thus, the types of intermediate projections that are common in phrase structure, which cover only some of the dependents of a word (such as an N' as a projection between N and NP), do not directly correspond to anything in a dependency representation. Attachments at the different levels of projection are therefore not distinguished in a dependency tree. This has certain ramifications for annotation, though in practice very few syntactically relevant distinctions are made by choosing attachments at different levels of projection. The most prominent one involves scope in conjunctions, so that the two readings of *young men and women* can be distinguished (are the women young as well or not?). If a dependency representation chooses to represent conjunction by treating the conjunction as a dependent to the first conjunct, then the two readings do not receive different syntactic representations, unless a specific scope feature is introduced for the adjective. It is worth noting that most scope distinctions are to our knowledge never represented in syntactic treebanks, such as scope of negation, modals, and quantified NPs.

A key element of our approach is a commitment to automatic conversion from the manual dependency structure treebank to our phrase structure treebank. Bringing these two major threads together, with clear principles and general tools for mapping back and forth between them, makes the results of each side immediately available to the other, with a major boost in productivity for both. Conversion from PS to DS is well understood, and consists primarily of identifying the heads of the phrases and preserving their dependency relations as reflected in the phrase structure tree. Conversion going in the opposite direction, DS-to-PS, is usually considered more difficult, since it requires making explicit information that is not typically represented in the dependency structure (Xia and Palmer, 2001). We have chosen the more challening DS-to-PS direction since dependency is a natural choice for Hindi because of the rich Paninian tradition and the relatively free word order of Hindi. We also believe that annotating Hindi with DS is faster than annotating PS, and we can build on existing work (Sharma et al., 2006). Given this choice, we must ensure any additional information that phrase structure requires is available at the time of the conversion process (see Section 3 for more details). In addition, we plan to manually check a portion of the output of the automatic DS-to-PS conversion to ensure the high quality of the conversion.

In order to ensure successful conversion from DS to PS, we are simultaneously developing three sets of guidelines for Hindi: one for dependency structure, one for phrase structure, and one for PropBank (PB). While allowing DS and PS guidelines to be based on different, independently motivated principles (see Section 4), we have been going through a comprehensive list of constructions in Hindi as a team, carefully exploring any issues that might impact the conversion process, such as at which stage information needed by PS should be added. In particular, during the guideline design phase, we make sure that DS and PB contain sufficient information to ensure successful conversion; that is, if PS makes a distinction that neither DS nor PB make, and the distinction is not systematic enough and therefore cannot be automatically learned reliably by the conversion process, then the conversion would not be successful. Therefore these distinctions must be manually annotated. For instance, the distinction between unergative intransitive verbs and unaccusative intransitive verbs must be marked by either DS or PB for the conversion process to produce a PS that captures this distinction (see Section 4.2). Furthermore, we coordinate the guidelines for DS and PS with respect to the examples chosen to support the conversion process. These examples form a conversion test suite.

We have developed an automatic DS-to-PS conversion algorithm, which we discuss in Section 5.1. We test the algorithm on the test suite of examples taken from the guidelines. This ensures: 1) that the PS and DS guidelines are consistent; and 2) that sufficient information (e.g., a rich set of dependency types) is included in the input DS to allow high-quality conversion. The manual annotation has two stages: 1) we first manually create DS's for the sentences; and 2) then add additional lexical information, including many empty arguments, at the PropBank stage. The corresponding PS annotations will be generated automatically from the DS + PB annotation by applying our conversion algorithm.

3 Syntactic Annotation Choices: The Basics

This section provides more details about the annotation guidelines at the three different levels.

3.1 Dependency Structure Guidelines

The Paninian grammatical model (Bharati et al., 1995; Begum et al., 2008) has been chosen as the basis of our dependency structure analysis. The Paninian perspective views a sentence as a basis of communication and aims at interpreting meaning through its realization in a sentence. The model offers a syntactico-semantic level of linguistic knowledge with an especially transparent relationship between the syntax and the semantics. The sentence is treated as a series of modifier-modified relations which has a primary modified (generally the main verb). The appropriate syntactic cues (relation markers) help in identifying various relations. The relations are of two types: karaka and others. *Karakas* are the roles of various participants in an action (arguments). For a noun to hold a karaka relation with a verb, it is important that they (noun and verb) have a direct syntactic relation. Panini has spelled out six karakas. Relations other than karakas are captured using the relational concepts of the model (adjuncts). For example, relations such as purpose, reason, and possession are marked within this scheme. These argument labels are very similar in spirit to the verb-specific semantic role labels used by PropBank, which have already been successfully mapped to richer semantic role labels from VerbNet and FrameNet (Loper et al., 2007; Yi et al., 2007). This suggests that much of the task of PropBanking can be done via a deterministic karaka-PropBank mapping based on the dependency annotation.

3.2 PropBank Guidelines

PropBanking involves a semantic layer of annotation that adds predicate argument structures to syntactic representations (Palmer et al., 2005; Babko-Malaya, 2005). The semantic roles of predicates are described at two levels of granularity. For the purposes of annotating the treebanked structures, the arguments of the verbs are labeled using a small set of numbered arguments, e.g. "Arg0", "Arg1", "Arg2", etc. In the frame files that are associated with each verb, the numbered argument labels for each verb are associated with fine-grained verb-specific descriptions. For instance, the verbs *see* and *eat* both have two semantic roles, an "Arg0" and an "Arg1". But the verb-specific roles associated with the two arguments are quite different. In the case of the verb *see*, "Arg0" is specified as the viewer, and "Arg1" as the thing viewed. In the case of the verb *eat*, "Arg0" is the eater and "Arg1" is the meal. An analogy can be drawn to Dowty's Prototypical Agent (Arg0) and Prototypical Patient (Arg1) (Dowty, 1991). Additionally, verb modifiers are annotated using functional tags such as ArgM-LOC, ArgM-TMP, ArgM-MNR to represent locative, manner, and temporal adjuncts respectively. The PropBank Framesets for Hindi will include mappings to the karaka, k1, k2, etc., many of which can be made deterministically. Finally, the Hindi PropBank (PB) also annotates (or adds) null elements (empty arguments such as dropped subjects and objects) in the syntactic representation using the information about lexical subcategorization provided in the frame files. The null elements provide critical information that the conversion process uses to produce the PS structure.

3.3 Phrase Structure Guidelines

The Phrase Structure guidelines are inspired by the Principles-and-Parameters methodology, as instantiated by the theoretical developments starting with Government and Binding Theory (Chomsky, 1981). A binary branching representation is consistently assumed. We make three theoretical commitments/design considerations that underlie the guidelines. The first is that any minimal clause distinguishes at most two positions structurally (the core arguments). These positions can be identified as the specifier of VP and the complement of V. With a transitive predicate, these positions are occupied by distinct NPs while with an unaccusative or passive, the same NP occupies both positions. All other NPs are represented as adjuncts. Secondly, we represent any displacement of core arguments from their canonical positions, irrespective of whether it crosses a clause boundary, via traces. The displacement of other arguments is only represented if it crosses a clause boundary. A third assumption that inspires the phrase structure decisions is the idea that syntactic relationships such as agreement and case always require c-command but do not necessarily require a [specifier, head] configuration. Within these constraints, we always choose the simplest structure compatible with the word order. We work with a very limited set of category labels (NP, AP, AdvP, VP, CP) assuming that finer distinctions between different kinds of verbal functional heads can be made via features.

3.4 Ensuring Consistency

In the next section we discuss in detail several examples, many of which could have quite different analyses in DS and PS. We show how we are coordinating the syntactic analysis between the two different frameworks to facilitate the conversion. We also discuss additional information, such as traces of displaced arguments and other null elements, that is only required by PS. We describe at what point in the annotation process the null elements are added, either manually during DS or PB, or automatically during conversion.

4 Hindi Syntax

In this section, we discuss a few common constructions in Hindi and show how they are represented in DS, PB, and PS. We point out theoretical issues that could potentially cause difficulties for the conversion process, and how we have chosen to address them.

4.1 Basic Clause Structure

In any Hindi clause, there are at most two arguments which can agree with the verb. Instead of talking about "subject" and "object", we will call these arguments the "first argument" and the "second argument", respectively, because they do not behave exactly like, say, English subject and object. The higher argument can be nominative (null case marker/postposition) or dative (*ko*; in this case there is an obligative meaning), and for transitive verbs, ergative (*ne*). The lower argument can be nominative or dative. The realization of case on these two arguments is not lexically dependent. The verb agrees (in gender and number) with a nominative argument (first or second). If both the first and the second argument are nominative, then agreement is with the first; if neither argument is nominative, then the verb shows default masculine singular agreement. Here are some transitive examples showing different case markings (and thus different agreement on the verb).[1]

(1) Transitive verb with no case markers:

Atif	kitaab	paRh-egaa
Atif.M	book.F	read-Fut.3MSg

'Atif will read (a/the) book.'

(2) Transitive verb with the first argument in ergative:

Atif	ne	kitaab	paRh-ii
Atif.M	Erg	book.F	read-Pfv.FSg

'Atif read (a/the) book.'

(3) Transitive verb with the first argument in ergative and the second argument with *ko* marker:

[1]The meanings of grammatical markers used in the examples are as follows: 1,2,3 (1/2/3 person), Acc (Accusative), Dat (Dative), Erg (Ergative), F (Feminine), Fut (Future), Gen (Genitive), Inf (Infinitive), M (Masculine), Pl (Plural), Pfv (Perfective), Prog (Progressive), Prs (Present), and Sg (Singular).

Atif	ne	kitaab	ko	paRh-aa
Atif.M	Erg	book.F	Dat	read-Pfv

'Atif read the book.'

4.1.1 Dependency

The two main arguments of the transitive verb in the DS attach to the verb with dependency types k1 and k2. Figure 1 shows the DS for Ex (3). For the sake of simplicity, in DS the case marker is conjoined to the preceding noun (as indicated by the underscore); the noun and its case marker form a word group, which is represented as a single node in the DS. The other two transitive verb examples have different case markers for k1 and k2, but the dependency types remain the same; therefore, the DS's for them are identical to the DS in Figure 1, barring the different case markers and the inflected form of the verb.

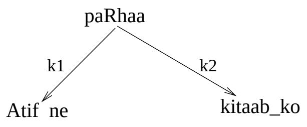

Figure 1: DS for the transitive example in (3)

4.1.2 PropBank

The arguments of the verb in the transitive structure receive semantic role labels specified in the frame file associated with the verb. The annotation and the frame for Ex (3) are below; those for Ex (1) and (2) are identical except for the different case markers and the inflected form of the verb. Note that the Arg0 and the Arg1 line up exactly with k1 and k2 for all three examples.

(3-PB) Semantic labeling for the transitive example in (3):

```
rel: paRh-aa
Arg0: Atif ne
Arg1: kitaab ko

Roleset id: paRh.01 ('to read')
  Arg0: reader
  Arg1: what is read
```

4.1.3 Phrase Structure

The first and second arguments are structurally distinguished using node labels (V, VP-Pred, VP), and thus no functional tags are needed to identify these two arguments (just like in the English PTB the syntactic object is not marked with functional tags). Unlike DS, each word has its own node in the phrase structure representation. This can be seen in the PS trees in Figures 2-4, which differ primarily in the presence or absence of postpositional markers. In contrast, postpositions do not create additional structure in the DS representation, where a noun and its postposition are put together and correspond to a single node. DS and PS differ in this aspect because words in a word group in DS are not always siblings in PS. For instance, while auxiliaries and main verbs form a verb group in DS (see Figure 5), they attach to different levels in PS (see Figure 6).

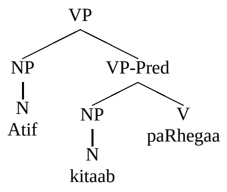

Figure 2: PS for the transitive example in Ex (1)

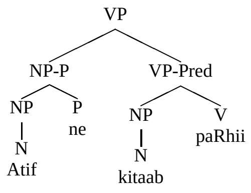

Figure 3: PS for the transitive example in Ex (2)

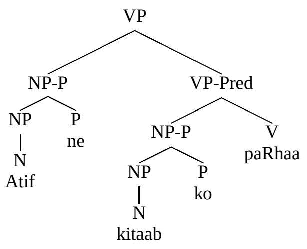

Figure 4: PS for the transitive example in Ex (3)

4.2 Unaccusative Verbs

Unaccusative verbs are intransitive verbs (verbs with only one argument) in which, roughly speaking, the first argument has semantic properties usually associated with lower arguments (such as the second argument). In the examples immediately below, the first object is the object undergoing *opening*, not the agent bringing about the *opening* (as is usually the case with first arguments). Ergative case is not an option with unac-

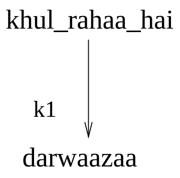

Figure 5: DS for the unaccusative example in Ex (4)

cusatives, but nominative and obligational dative cases are possible.

(4) Unaccusatives: nominative (zero marker)

darwaazaa	khul	rahaa	hai
door.M	open	Prog.MSg	be.Prs.Sg

'The door is opening.'

(5) Unaccusatives: dative

darwaaze	ko	khul-naa	hai
door.M	Dat	open-Inf	be.Prs

'The door has to open.'

In fact, the issue of unaccusativity in Hindi is far more complex and a more detailed discussion is outside the scope of the paper. In this paper, we accept the generalization about unaccusativity as presented here, which correctly characterizes a significant set of verbs.

4.2.1 Dependency

The DS treats unaccusatives like other intransitive verbs and the first argument of the verb is annotated as *k1*. Like the word group for a noun and its case marker, in the DS we also group auxiliary verbs and main verbs into a verb group. Figure 5 shows the DS for Ex (4), and the DS for Ex (5) is similar.

4.2.2 PropBank

PropBank assigns the label "Arg0" for the first argument of intransitive unergative verbs and "Arg1" for unaccusative verbs. The verb *khul* ('open') is unaccusative by a number of diagnostics, as reflected in the frame file for this verb. Below is the PB annotation for Ex (4), and the analysis for the unaccusative verb in Ex (5) is similar.

(4-PB) Semantic labeling for the unaccusative example in (4):

```
rel: khul rahaa hai
Arg1: darwaazaa
```

```
Roleset id: khul.01 ('to open')
  Arg1: the thing opening
```

The unusual mismatch between the *karaka* label and the PB label constitutes explicit marking of the unaccusative verb, and therefore provides necessary guidance that allows the conversion process to correctly produce the PS given below.

4.2.3 Phrase Structure

For PS, we insert a special trace in the second position (*CASE*), representing the fact that semantically, the constituent in first position comes from somewhere below. This can be interpreted as follows: the constituent gets its semantic (lexical-semantic) interpretation in the second position, then moves, and behaves syntactically like any other first position constituent. Our analysis follows the standard analysis of unaccusativity as articulated in (Burzio, 1986).

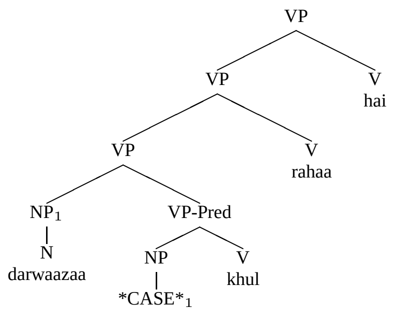

Figure 6: PS for the unaccusative example in (4)

4.3 Support Verb Constructions

Hindi-Urdu productively allows eventive noun phrases to combine with a number of verbs (most prominently *ho* 'be' and *kar* 'do') to form transitive and intransitive verbal structures. There are two kinds of constructions: in one, an argument appears with genitive case (suggesting case marking happens within the NP), while in the other, all arguments have verbal case marking. Note that some nouns allow only the genitive, others do not allow the genitive at all, and some allow both constructions. Note that there are also support verb constructions with dative arguments, which we do not have space to discuss.

Type 1: the argument of the eventive noun has genitive case. The verb agrees with the eventive noun.

(6) NP + be:

gehenoN	kii	kal	chorii	huii
jewels.MPl	Gen.F	yesterday	theft.F	be.Pfv.FSg

‘The jewels were stolen yesterday’

(7) NP + do:

Atif	ne	gehenoN	kii	kal	chorii	kii
Atif	Erg	jewels.MPl	Gen.F	yesterday	theft.F	do.Pfv.FSg

‘Atif stole the jewels yesterday.’

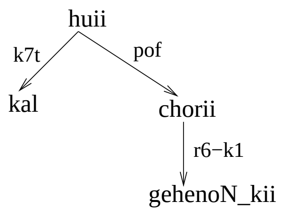

Figure 7: DS for the type 1 example in Ex (6)

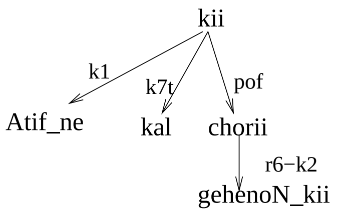

Figure 8: DS for the type 1 example in Ex (7)

Type 2: the argument of the eventive noun has verbal case marking. In this case, the verb agrees with the argument of the eventive noun if it is nominative.

(8) NP + *be*: no genitive, no external argument:

geheneN	kal	chorii	ho	gaye
jewels.MPl	yesterday	theft.F	be	GO.Pfv.MPl

‘Yesterday the jewels got stolen.’

(9) NP + *kar*: no genitive, external argument:

Atif	ne	kal	geheneN	chorii	kiye
Atif.M	Erg	yesterday	jewels.MPl	theft.F	do.Pfv.MPl

‘Atif stole the jewels yesterday.’

4.3.1 Dependency

The DS interprets the event nominal *chorii* and the verbalizer (support verb) *huii/kii* as a unit, represented by the *pof* label. This unit inherits its arguments from the event nominal. Support verbs of Type 1 and Type 2 are handled differently in the DS. For Type 1, *gehenoN* depends on *chorii* because the genitive case indicates that *syntactically* the former is closely related to the latter; As a result, the DS attaches *gehenoN* to *chorii* (the event nominal) with a r6-k1 label (Figure 7) or r6-k2 label (Figure 8). In Type 2, there are no genitive case markers to link *geheneN* and *chorii* syntactically, so both *geheneN* and *chorii* are attached to the verb *ho_gaye/kiye* as shown in Figures 9-10. The *pof* label for *chorii* indicates that *chorii* and *ho_gaye/kiye* together represent the predicate relation of the clause.

4.3.2 PropBank

Annotation of support verb constructions is especially challenging for PropBanking, since some of the arguments are clearly determined by the nominal predicate rather than the verbal predicate. We prefer to think of them as complex predicates, whose argument structure can only be determined by considering the influence of both the verbal and nominal predicate. Our current approach is to identify the verb as appearing in a support verb construction and do minimal annotation before passing it along for nominal predicate argument annotation. In this paper we present the final argument labels that would be arrived at after the two-pass annotation.

In Ex (8) and (9), the object nominal *chorii* (’theft’) can be viewed as constituting an argument of the verb as well as part of a complex predicate that includes the verb. This situation is reinforced by selecting the verb frame file associated with the “support verb” senses of the verbs *ho* (’be’) and *kar* (’do’). The object nominal is a pred-

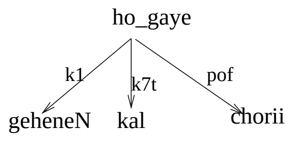

Figure 9: DS for the type 2 example in Ex (8)

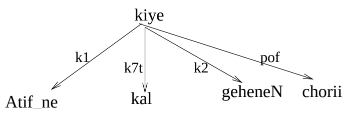

Figure 10: DS for the type 2 example in Ex (9)

icate that itself takes its own arguments and has a frame file of its own. The final annotation combines the arguments from the support verb and the noun to create a single annotation for the phrase as a whole, as shown in (8-PB) and (9-PB). The constructions with genitive case-marking shown in Ex (6) and (7) receive a similar analysis. The exact form of the frame files for the support verbs and object nominals are still under discussion and are not included here.

(8-PB) Semantic labeling for the example in (8):

```
rel: chorii ho gaye
Arg1: geheneN
ArgM-TMP: kal
```

(9-PB) Semantic labeling for the example in (9):

```
rel: chorii kiye
Arg0: Atif ne
Arg1: geheneN
ArgM-TMP: kal
```

4.3.3 Phrase Structure

In both Type 1 and Type 2, the eventive NP headed by *chorii* 'theft' takes *geheneN* 'jewels' as its internal argument. Type 1 and Type 2 differ with respect to whether the eventive NP is a proper argument (first or second) of the verb. In Type 1, the eventive NP is a proper argument as can be seen by the fact that it triggers agreement. Its internal argument receives genitive case within the NP and any further movement of the internal argument is an optional scrambling movement as indicated by the *SCR* trace. The PS's for Ex (6) and (7) are shown in Figures 11 and 12, respectively.

The analysis for Type 2 is quite different as shown in Figures 13-14: the eventive NP headed by *chorii* 'theft' is not in the second position; rather, it is a sister to the verb, in a non-case and non-agreement position. Instead *geheneN* 'jewels', the internal argument of *chorii* 'theft', moves to the second position out of the eventive NP. This movement is for case reasons as indicated by the *CASE* trace and is obligatory.

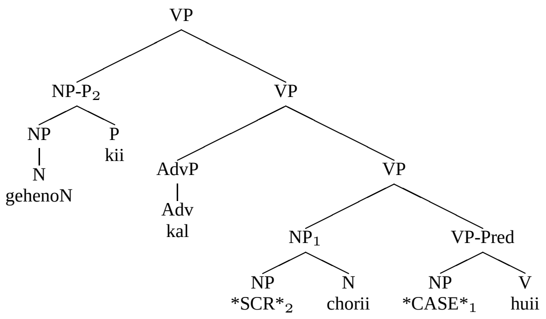

Figure 11: PS for the support verb construction example in (6)

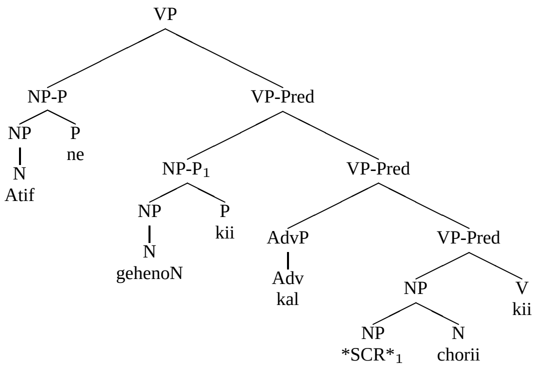

Figure 12: PS for the support verb construction example in (7)

5 A Note on the Overall Process

The first step in our pipeline is the manual annotation of DS. Traditionally, DS annotation attaches arguments to the relevant predicating expression without worrying about canonical word order or missing predicates/arguments. We follow the tradition with the exception of inserting missing predicates (e.g., in the gapping construction) to DS so that their arguments can attach to them. We briefly considered asking the DS tree-

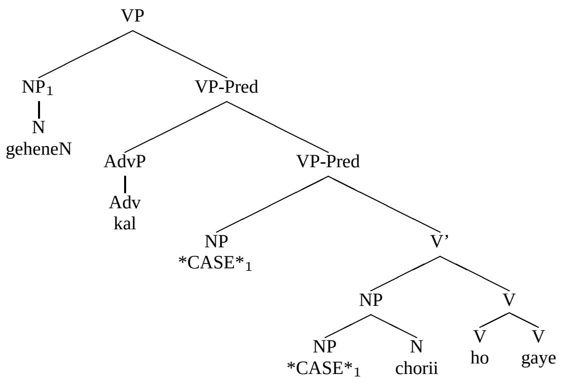

Figure 13: PS for the Type 2 example in (8)

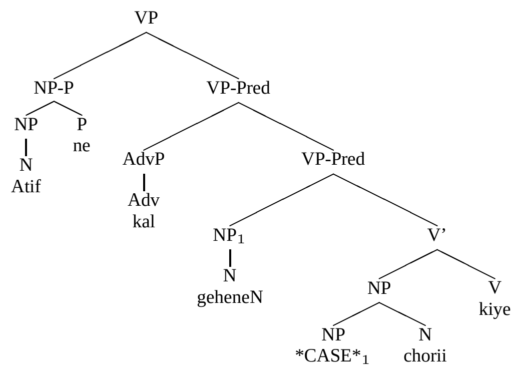

Figure 14: PS for the Type 2 example in (9)

bankers to mark missing arguments during the annotation process, but this proved to be a major additional burden for these annotators. Since the main focus of the PropBank annotation is the lexical predicate-argument structure, it was decided that this would be a more appropriate stage for taking into account missing arguments. Displaced arguments are marked in PS only; the conversion process will insert traces and coindex them with the antecedents automatically. This process is possible because the displaced arguments can be easily detected by looking at the word order.

We have developed a table that covers a wide range of empty categories, and specifies at what stage in the process they will be added. Some are added manually during the DS annotation if their addition creates a node that is essential to the DS tree structure. Many more are now being added during the PropBanking annotation. Therefore, the conversion process will take as input the original DS structure augmented with the PB argument labels and empty arguments. The close correspondence between the Paninian karaka and the PropBank labels will simplify the labeling aspect of the PB annotation, allowing additional time for the empty arguments. Finally, traces for displaced arguments can be added automatically during the conversion process.

Our goal is to manually treebank 400K words of naturally occurring Hindi newswire, and, using similar guidelines, 200K words of naturally occurring Urdu newswire, with DS structure. The assumption is that a high precision transliteration process will allow us to effectively merge the two treebanks for both languages. This data will then be PropBanked, and finally converted to PS. All of the levels of annotation as well as the original text will be made freely available. A long term goal is a larger treebank, which will allow us to add other genres and create a more balanced corpus. We have already successfully trained a parser on a pilot version of the DS treebank (Bharati et al., 2008), and will retrain it as soon as a new 100K-word treebank is available.

We use standard inter-tagger agreement (ITA) as a first step towards evaluating the consistency and accuracy of our annotation. The subsequent layers provide additional quality control and often higlight inconsistencies or inaccuracies. The conversion process is the most stringent test of the annotation, since it requires perfect formatting and consistency for the conversion to succeed.

5.1 Conversion

The DS-to-PS conversion process (assuming DS plus PB) has three steps. First, for each (DS, PS) pair appearing in the conversion test suite, we run a consistency checking algorithm to determine whether the DS and the PS are consistent. The inconsistent cases are studied manually and if the inconsistency cannot be resolved by changing the analyses used in the guidelines, a new DS that is consistent with the PS is proposed. We call this new dependency structure ”DS$_{cons}$” (”cons” for ”consistency”; DS$_{cons}$ is the same as DS for the consistent cases). Because the DS and PS guidelines are carefully coordinated, we expect the inconsistent cases to be rare and well-motivated. Second, conversion rules are extracted automatically from these (DS$_{cons}$, PS) pairs. Last, given a new DS, a PS is created by applying conversion rules. Note that non-projective DS's will be converted to projective DS$_{cons}$. A preliminary study on the English Penn Treebank showed promising results. Error analyses indicated that most conversion errors were caused by ambiguous DS patterns in the conversion rules. This implies that including sufficient information in the input DS (plus PB) could reduce ambiguity, significantly improving the performance of the conversion algorithm. The details of the conversion algorithm, definition of consistency, and the experimental results are described in (Xia et al., 2009).

6 Conclusion

We have presented our approach to producing Hindi and Urdu treebanks that are both multi-representational, i.e., DS and PS for the same data, and multi-layered, i.e., with layers for both syntactic structure and semantic structure (both karaka and PB). This approach requires careful coordination of the guidelines for DS, PS and PB. Current versions of these guidelines can all be found on our project Wiki.[2]

Since our plan is to manually annotate DS and PB and then automatically produce PS from DS + PB, we have had to carefully (1) coordinate syntactic analyses between the guidelines, and (2) consider any information that PS might require which DS would not traditionally supply. We have chosen to add many explicit empty arguments desired by PS at the PropBank annotation stage, and will therefore take the manual annotation of both DS and PB as input to our conversion process. The conversion process will itself be able to deterministically add traces for displaced arguments. In this paper we have provided details of the representations of several different types of syntactic constructions to illustrate the impact of our choices.

Acknowledgments This work is supported by NSF grants CNS-0751089, CNS-0751171, CNS-0751202, and CNS-0751213.

References

O. Babko-Malaya. 2005. Propbank 1 annotation guidelines.

Rafiya Begum, Samar Husain, Arun Dhwaj, Dipti Misra Sharma, Lakshmi Bai, and Rajeev Sangal. 2008. Dependency annotation scheme for indian languages. In *Proceedings of IJCNLP-2008*, Hyderabad, India.

Akshar Bharati, Vineet Chaitanya, and Rajeev Sangal. 1995. *Natural Language Processing – A Paninian Perspective*. Prentice-Hall of India.

Akshar Bharati, Samar Husain, Bharat Ambati, Sambhav Jain, Dipti M Sharma, and Rajeev Sangal. 2008. Two semantic features make all the difference in parsing accuracy. In *Proceedings of ICON-2008*, CDAC Pune, India.

Luigi Burzio. 1986. *Italian syntax: a government-binding approach*. Studies in natural language and linguistic theory. Kluwer, Dordrecht.

Jinho Choi, Martha Palmer, and Nianwen Xue. 2009. Using parallel propbanks to enhance word-alignments. In *Proceedings of the Third Linguistic Annotation Workshop*.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.

Michael Collins, Jan Hajič, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for czech. In *Proceedings of ACL-1999*.

Michael Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of ACL-1997*.

David Dowty. 1991. Thematic proto-roles and argument selection. *Language*, 67(3):547–619.

J. Hajič, E. Hajicova, M. Holub, P. Pajas, P. Sgall, B. Vidova-Hladka, and V. Reznickova. 2001. The Current Status of the Prague Dependency Treebank. In *Lecture Notes in Artificial Intelligence (LNAI)*, volume 2166, pages 11–20.

T. H. King, R. Crouch, S. Riezler, M. Dalrymple, and R. Kaplan. 2003. The PARC700 Dependency Bank. In *Proc. of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-2003)*, Budapest, Hungary.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the Penn TreeBank. In *Proceedings of HLT-2002*, San Diego, CA.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of ACL-2003*.

Seth Kulick, Ryan Gabbard, and Mitch Marcus. 2006. Parsing the Arabic Treebank: Analysis and Improvements. In *Proceedings of the 5th Conference on Treebanks and Linguistics Theories (TLT-2006)*, pages 31–32.

R. Levy and C. D. Manning. 2003. Is it Harder to Parse Chinese, or the Chinese Treebank? In *Proceedings of ACL-2003*, Sapparo, Japan.

Edward Loper, Szu-Ting Yi, and Martha Palmer. 2007. Combining Lexical Resources: Mapping between PropBank and VerbNet. In *Proc. of the 7th International Workshop on Computational Linguistics*, Tilburg, the Netherlands.

M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the ARPA Human Language Technology Workshop*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT/EMNLP 2005*.

S. Oepen, K. Toutanova, S. M. Shieber, C. D. Manning, D. Flickinger, and T. Brants. 2002. The LinGO Redwoods Treebank: Motivation and Preliminary Applications. In *Proc. of COLING*, Taipei, Taiwan.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Dipti Misra Sharma, Rajeev Sangal, Lakshmi Bai, and Rafiya Begam. 2006. Anncorra: Treebanks for indian languages (version - 1.9). Technical report, Language Technologies Research Center IIIT, Hyderabad, India.

Fei Xia and Martha Palmer. 2001. Converting Dependency Structures to Phrase Structures. In *Proceedings of HLT-2001*, San Diego, CA.

Fei Xia, Owen Rambow, Rajesh Bhatt, Martha Palmer, and Dipti Misra Sharma. 2009. Towards a multi-representational treebank. In *The 7th International Workshop on Treebanks and Linguistic Theories (TLT-7)*, Groningen, Netherlands.

Szu-Ting Yi, Edwarwd Loper, and Martha Palmer. 2007. Can semantic roles generalize across genres? In *Proceedings of HLT/NAACL-2007*.

[2]https://verbs.colorado.edu/hindi_wiki/index.php/Main_Page