Explicit Argument Identification for Discourse Parsing In Hindi: A Hybrid Pipeline

Rohit Jain and **Dipti Misra Sharma**
Language Technologies Research Center (LTRC), IIIT-H
Hyderabad,
Telangana, India

Abstract

Shallow discourse parsing enables us to study discourse as a coherent piece of information rather than a sequence of clauses, sentences and paragraphs. In this paper, we identify arguments of explicit discourse relations in Hindi. This is the first such work carried out for Hindi. Building upon previous work carried out on discourse connective identification in Hindi, we propose a hybrid pipeline which makes use of both sub-tree extraction and linear tagging approaches. We report state-of-the-art performance for this task.

1 Introduction

Units within a piece of text are not meant to be understood independently but understood by linking them with other units in the text. These units may be clauses, sentences or even complete paragraphs. Establishing relations between units present in a text allows the text to be semantically well structured and understandable. Understanding the internal structure of text and the identification of discourse relations is called discourse analysis.

A fully automated shallow discourse parser would greatly aid in discourse analysis and improve the performance of Text summarization and Question answering systems. Given a text, a shallow discourse parser would identify discourse relations, consisting of two spans of text exhibiting some kind of relationship between each other. Discourse relations whose presence is marked explicitly by discourse connectives are called Explicit discourse relations and those which are not are called Implicit discourse relations.

At present, complete shallow discourse parsers are only available for English (Lin et al., 2014; Wang and Lan, 2015; Ali and Bayer, 2015). The ongoing CoNLL 2016 shared task on Shallow Discourse Parsing has included Chinese as well. Work towards a complete shallow discourse parser in Hindi has also begun. Jain et al. (2016) reported state-of-the-art results for discourse connective identification in Hindi. Our work focuses on the next part towards a shallow discourse parser for Hindi i.e. argument identification for Explicit discourse relations.

In this paper, we discuss current approaches for this task and also propose a hybrid pipeline incorporating many of these approaches. We report high accuracies of 93.28% for Arg2 identification, 71.09% for Arg1 identification and 66.3% for Arg1-Arg2 identification.

The rest of the paper is organized as follows. Section 2 briefly introduces the Hindi Discourse Relations Bank(HDRB). Related work carried out in English is discussed in Section 3. In section 4, we describe in detail our approach to argument identification of Explicit discourse relations. Section 5 discusses the performance of the proposed pipeline and we conclude in Section 6.

2 Hindi Discourse Relations Bank(HDRB)

The Hindi Discourse Relation Bank(HDRB) was created broadly following the lines of Penn Discourse TreeBank(PDTB) (Miltsakaki et al., 2004; Prasad et al., 2008)'s lexically grounded approach along with a modified annotation workflow, additional grammatical categories for explicit connectives, semantically driven Arg1/Arg2 labelling and modified sense hierarchies.(Oza et al., 2009; Kolachina et al., 2012)

HDRB was annotated on a subset of the Hindi TreeBank (Begum et al., 2008) which includes part-of-speech, chunk and dependency parse tree annotations. HDRB contains 1865 sentences and a word count of 42K. Furthermore HDRB contains 650 explicit discourse relations and 1200 implicit discourse relations.

In HDRB, one of the arguments occurs after the discourse connective and the other occurs before the connective. Discourse relations not adhering to this rarely occur in the corpus. However, due to the semantic labelling of Arg1 and Arg2, Arg2 does not always occur after the connective. For example:

- चंदीगढ में बृहस्पतिवार की सुभह भारी वर्शा की वजह से इसके आसपास और निचले इलाकों में बाढ़ की स्थिति पैदा हो गई और कई सद्कों, विशेष कर दक्सीणी सेकटरों मेइ स्थिती काफी बिगड गई है ।

- *Heavy rains have occurred in Chandigarh* because of which **there is possibility of floods in nearby and lower areas and the condition of roads, especially in the southern sectors, has worsened.**

The relation sense is "Contingency cause relation", where the situation described in Arg2 (italicized) is the cause of the the situation described in Arg1 (bolded). Due to this fact, Arg2 occurs before Arg1. However, for the purpose of argument identification we refer to the argument occurring before the connective as Arg1 and the argument occurring after the connective as Arg2. We believe changing the labels later on during sense identification to be the simpler approach.

In the corpus, Arg1 can occur in the same sentence as the connective (**SS**) or in the sentence preceding that of the connective (**PS**) with proportions of 46% and 54% respectively, whereas Arg2 only occurs in the same sentence as the connective.

Arg1 can cover 1,2,3 or even more than 4 sentences with proportions of 89.2%, 5.4%, 2.6% and 2.8% respectively. As such in this paper, we only consider the sentence containing the connective and the sentence immediately preceding it for Arg1 identification.

3 Related Work

Argument identification for Hindi has not been explored before, therefore we discuss some of the approaches adopted for English.

Ghosh et al. (2011) proposed a linear tagging approach for argument identification using Conditional random fields and n-best results.

Lin et al. (2014) proposed a sub-tree extraction approach for argument identification. Firstly an argument position classifier was employed to decide the location of Arg1(PS/SS). In the case of PS, Arg1 was labelled as the entire preceding sentence. For tagging Arg1(SS) and Arg2, a argument node identifier was employed to decide which nodes were part of Arg1(SS) or Arg2. Next sub-tree extraction was used to extract Arg1(SS) and Arg2. However, since it is not necessary that arguments may be dominated entirely by a single node as pointed out by Dinesh et al. (2005), this method has inherent shortcomings.

Kong et al. (2014) proposed a constituent based approach where, similar to Lin, an argument identifier is employed to decide which constituents are Arg1 and Arg2. Previous sentence was considered as a special constituent to handle Arg1(PS). A constituent pruner was also employed to reduce the number of candidate constituents considered for Arg1 and Arg2. In addition, Integer Linear Programming(ILP) with language specific constraints, was employed to ensure the argument identifier made legitimate global predictions.

Approaches in English can be summed up as two sub-tasks: (1) Considering the possible constituents/nodes/words to be identified as Arg1 or Arg2 by use of subtree extraction (Lin et al., 2014), constituent pruning (Kong et al., 2014) or simple baseline (Ghosh et al., 2011) approaches. (2) Classification of selected constituents/nodes/words as Arg1/Arg2/None by use of CRF(Ghosh et al., 2011) or classifier(Lin et al., 2014; Kong et al., 2014) based approaches.

4 A Hybrid Pipeline to Argument Identification

We base our pipeline on the two sub tasks discussed in the previous section. We use a method similar to subtree extraction to extract possible candidates for Arg1/Arg2 and use CRF tagging to further refine the extent of the extracted arguments.

We approach the task of Arg1 and Arg2 identification seperately since tagging Arg1 is inherently more difficult. We first discuss Arg2 identification and then Arg1 identification. Features used are listed in Table 1.

4.1 Arg2 Identification

Doing a simple analysis on HDRB, we find that Arg2 largely lies in two positions in the dependency tree. Arg2 can either occur in the subtree of the connective node(Conn-SubTree) or in the subtree of the first verb group node occurring as parent to the connective node(Parent-VG-SubTree) as shown in Image 1.

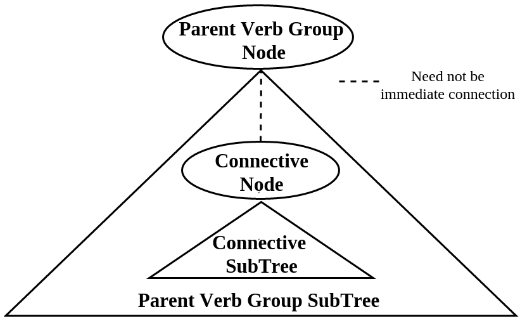

Figure 1: Arg2 Sub Tree Positions

To decide the position of Arg2, we make use of a classifier with *Conn-St*r, *Conn-Pos-Sentence*, *Is-Leaf-Node*, *VG-In-SubTree*, *VG-In-Parent-SubTree* and *Right-Word-Location* as features. Once we have the position of Arg2, all the nodes present in the subtree are extracted as Arg2. Henceforth, we refer to this step as ***SubTree-Extraction***.

Although Arg2 lies in either in "Conn-SubTree" or "Parent-VG-SubTree", it does not necessarily cover the entire subtree. Thus we need to refine the extent of Arg2 extracted from the *SubTree-Extraction* . We approach this as a linear tagging task, allowing us to capture the local dependency between nodes. We use *Conn-Rel-Pos*, *Node-Tag*, *Clause-End*, *Is-Conn* and *Part-of-Conn-Sub-Tree* as features. Henceforth, we refer to this step as ***Partial-SubTree***.

We find that Arg2 sometimes extends further up into the dependency tree. For example:

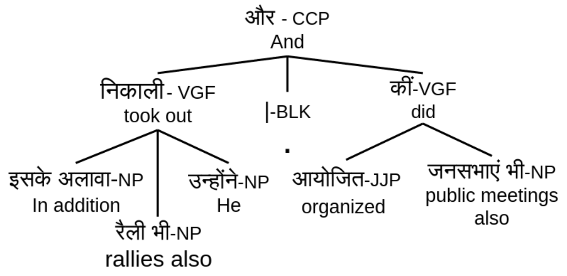

Figure 2: Arg2 Extended example dependency tree

- इसके अलावा , उन्होंने रैली भी निकाली और जनसभाएं भी आयोजित कीं ।
- In addition, he also took out rallies and also organized public meetings.

इसके अलावा (In addition)'s Arg2 lies in "Parent-VG-SubTree". However, the presence of *and* indicates some kind of correlation between "he also took out rallies" and "also organized rallies". This correlation is also indicated in the dependency tree where both VG groups are children of *and*. To handle these and other similar cases we employ a classifier to decide whether extension beyond the current node is necessary. The current node is either connective node or the parent VG node of the connective node depending upon the decision made in the *SubTree-Extraction* step. We use *Conn-Str*, *Node-Tag* of current node, *Node-Tag* of parent of current node, *Conn-Rel-Pos* for parent of current node as features for this step. Henceforth we refer to this step as ***SubTree-Extension***.

SubTree-Extraction, *Partial-SubTree* and *SubTree - Extention* complete the pipeline for Arg 2 identification.

4.2 Arg1 Identification

Following the approach adopted for English (Kong et al., 2014; Lin et al., 2014; Wang and Lan, 2015), we approach Arg1 as two distinct problems: Arg1(SS) and Arg1(PS) identification. We employ a classifier to decide the position of Arg1. We use *Conn-Str*, *Node-Tag* of connective, *Conn-Pos-Sentence*, *Chunk-Before-Conn* as features for this step. Henceforth we refer to this step as ***Arg1 Pos Identification***.

The position of Arg1(SS) in the dependency tree, similar to Arg2, shows strong correlation with the position of the connective in the dependency

Feature Name	Feature Description	Used In
Conn-Str	Connective String	A2 (SE1,SE2), A1(PI) A1-SS(PI,SE1,SE2)
Conn-Pos	Connective part-of-speech tag	
Node-Tag	Chunk tag of the node	A2 (PS,SE2), A1(PI) A1-SS(SE1,PS,SE2)
Conn-Pos Sentence	Connective position in the sentence (Start/Middle)	A2(SE1), A1(PI), A1-SS(SE1)
Is-Leaf-Node	Connective node is a leaf node in the dependency tree	A2(SE1), A1-SS(SE1)
VG-In SubTree	Presence of VG node in sub tree of node	A2(SE1), A1-SS(SE1)
VG-In-Parent SubTree	Presence of VG node in parent of node	A2(SE1), A1-SS(SE1)
Right-Word Location	Location of word immediately after connective in the dependency tree w.r.t connective node	A2(SE1), A1-SS(SE1)
Conn-Rel Pos	Position of chunk w.r.t connective in sentence. (Before/After)	A2(PS,SE2), A1(PS,SE2)
Clause-End	Indicates presence of clause boundary	A2(PS), A1(PS)
Is-Conn	Node is part of a discourse connective or not	A2(PS), A1-SS(PS)
Part-Conn SubTree	Indicates whether node is part of discourse connective subtree, other than the connective in question	A2(PS) A1-SS(PS)
Chunk-Before Conn	Number of chunks before discourse connective	A1(P1)
Arg2-Pos	Position of Arg2 in dependency tree	A1-SS(SE1)
Conn-Two Clause	Indicates the presence of two verb groups as children to connective node. Captures possible coordination of two verb groups by connective	A1-SS(SE1)
Verb-Group	Verb group string & POS tag sequence	A1-PS(VSL)
Verb-Group Compact	Verb group string and POS tag sequence consisting of main main and its corresponding auxiliary verbs	A1-PS(VSL)
Verb-Root Inflection	Root and Inflection of main and auxiliary verbs	A1-PS(VSL)

A1:Arg1,A2:Arg2,A1-SS:Arg1 Same Sentence, A1-PS: Arg1 Previous Sentence

SE1:SubTree Extraction, PS:Partial SubTree, SE2:SubTree Extension, PI:Position Identifier, VSL: VG SubTree Labelling

Table 1: List of features used for Argument Identification

tree. In addition to *Conn-SubTree* and *Parent-VG-SubTree*, Arg1(SS) also lies in the subtree of the first verb group node occurring as parent to *Parent-VG* (pParent-VG-SubTree). This happens when Arg2 lies in the *Parent-VG-SubTree*.

To identify Arg1(SS), we use the same pipeline used for Arg2 identification, with certain differences in choice of features. *SubTree-Extraction* uses *Conn-Str*, *Is-Leaf-Node*, *Arg2-Pos*, *Node-Tag* of parent node of connective, *Node-Tag* of parent of parent node of connective, *Conn-Two-Clause* as features. Both *Patial-SubTree* and *SubTree-Extension* use the same set of features used for Arg2 identification.

SubTree-Extraction, *Partial-SubTree* and *SubTree - Extention* complete the pipeline for Arg 1 (SS) identification.

A similar pipeline for Arg1(PS) identification cannot be used, since both Arg2 and Arg1(SS) showed a strong correlation to the connective node in the dependency tree. No such anchor node exists for Arg1(PS).

We divide the dependency tree of previous sentence into smaller verb group subtrees(VG SubTree). We consider each of them as candidates to be labelled as Arg1(SS). In the case of nested verb group sub trees, we treat them as two separate verb group subtrees ensuring no overlap of nodes between them. We refer to this step as **VG-SubTree-**

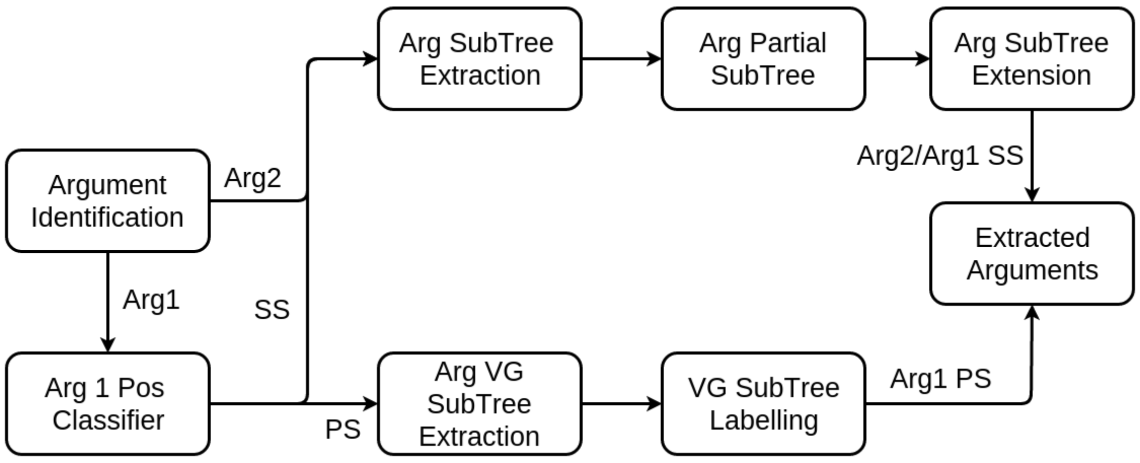

Figure 3: Argument Identication Pipeline

Extaction.

We make use of a classifier to decide whether each VG SubTree candidate is part of Arg1(PS) or not. We use *Verb-Group*, *Verb-Group-Compact*, *Verb-Root-Inflection* as features. All the nodes present in the VG SubTrees labelled as Arg1(PS) are extracted to form Arg1(PS). We refer to this step as **VG-SubTree-Labelling**.

VG-SubTree-Extraction and *VG-SubTree-Labelling* complete the pipeline for Arg1 (PS) identification. The entire pipeline for argument identification is shown below in Image 3.

5 Results

Firstly, we discuss the experimental setup, baselines and performance metrics we have considered to put the performance of our pipeline in perspective. Later on, we discuss, in detail, the performance of Arg2 and Arg1 identification pipelines.

5.1 Experimental Setup

Maximum Entropy (Fan et al., 2008) for classifier based steps and Conditional Random Fields (Lafferty et al., 2001) for linear tagging based steps were our choice of algorithms. L2 regularized Stochastic Gradient Descent (SGD) was used while training the CRF model and LibLinear solver (Fan et al., 2008) with L2 penalties was used to train the Maximum Entropy model. Maximum Entropy was implemented using Nltk toolkit[1] and Sklearn[2] whereas Conditional Random Fields was implemented using a CRFsuite[3](Okazaki, 2007).We used 5-fold cross validation to arrive at the results.

5.2 Baseline and Performance metrics

As discussed in Section 2, Arg2 is the argument occurring after the connective and Arg1 is the argument occurring before the connective. Therefore Arg2 baseline is computed by labelling Arg2 as the text span between the connective and the beginning of the next sentence. Similarly Arg1(SS) baseline is computed by labelling Arg1(SS) as the text span between the connective and the end the of the previous sentence. Arg1(PS) baseline is computed by labelling the entire previous sentence as Arg1(PS).

Ghosh et al. (2011), kong et al. (2014) and Lin et al. (2014) have reported performance using exact match metric. In addition to reporting performance using exact match metric, we introduce a new metric for measuring performance- Partial match:

$$\frac{\mid ArgResult \cup ArgGold \mid - 0.5 * \mid ARgResult \cap ArgGold \mid}{\mid argGold \mid}$$

Where ArgResult is the argument labelled by the system, ArgGold is the argument labelled in the corpus. Partial match scores between 0-1 and incorporates a penalty for each missed or erroneously labelled node/chunk. Partial match is thus a more lenient scoring metric than exact match, however the

[1]http://www.nltk.org/
[2]http://scikit-learn.org/stable/
[3]http://www.chokkan.org/software/crfsuite/

penalty ensures the leniency is limited. Partial match allows us to measure minor performance improvements that are not captured by exact match metric.

5.3 Arg2 Results

We report Arg2 identification results in Table 2

Step	Exact	Partial
Baseline	63.2	77.95
SubTree-Extraction	58.28	69.10
Partial-SubTree	91.56	92.88
SubTree-Extension	93.28	95.37

Table 2: Arg2 identification results

We report a baseline score of 63.2 and 77.95 for exact and partial matches respectively. *SubTree-Extraction* does not reach the performance of the baseline with scores of 58.28 for exact match and 69.10 for partial match. With an increase of 33.28 for exact match and 23.78 for partial match, *Partial-SubTree* step results in the largest performance gains. *SubTree-Extension* further improves performance by 1.72 and 2.49 for exact and partial respectively. For Arg2 identification, we report a final score of 93.28 for exact match and 95.37 for partial match.

5.4 Arg1 Results

Coming to arg1 identification, we report a high accuracy of 99.1 % for *Arg1 Pos Identification* step. This is similar to the performance reported by Lin et al. (2014). We find that *Conn-Pos-Sentence* is sufficient to decide between Arg1(PS) and Arg1(SS). Other features used result in minor improvements.

Step	Exact	Partial
Baseline	43.38	71.57
SubTree-Extraction	2.05	22.63
Partial-SubTree	70.05	79.56
SubTree-Extension	71.18	80.12

Table 3: Arg1(SS) identification results

We report Arg1(SS) results in Table 3. For Arg1(SS), we report a baseline score of 43.38 and 71.57 for exact and partial matches respectively. *SubTree-Extraction* performs poorly with a score of 2.05 for exact match and 22.63 for partial match. Similar to Arg2, we find that *Partial-Subtree* results in a large increase in performance of 68 for exact match and 56.93 for partial match. *SubTree-Extension* yields minor improvements of 1.13 and 0.56 for exact and partial respectively. For Arg1(SS) we report a final score of 70.84 for exact match and 80.12 for partial match.

Step	Exact	Partial
Baseline	71.01	72.38
System	38.55	62.07

Table 4: Arg1(PS) identification results

For Arg1(PS) we report a baseline of 71.05 and 72.38 for exact and partial matches respectively. We find that our system does not exceed the baseline scores with 38.55 for exact match and 62.07 for partial match. We believe more work is needed to successfully extract Arg1(PS).

We thus report an accuracy of 93.28% for Arg2 identification, 71.09% for Arg1 identification and 66.3% for Arg1-Arg2 identification.

6 Conclusion

In this paper, we focus on argument identification for explicit discourse relations in Hindi. In particular we propose a hybrid pipeline using both subtree extraction and linear tagging approaches. This is the first such work carried out in Hindi.

References

Evgeny A Stepanov Giuseppe Riccardi Ali and Orkan Bayer. 2015. The unitn discourse parser in conll 2015 shared task: Token-level sequence labeling with argument-specific models. *CoNLL 2015*, page 25.

Rafiya Begum, Samar Husain, Arun Dhwaj, Dipti Misra Sharma, Lakshmi Bai, and Rajeev Sangal. 2008. Dependency annotation scheme for indian languages. In *IJCNLP*, pages 721–726. Citeseer.

Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2005. Attribution and the (non-) alignment of syntactic and discourse arguments of connectives. In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*, pages 29–36. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Sucheta Ghosh, Richard Johansson, and Sara Tonelli. 2011. Shallow discourse parsing with conditional random fields. In *In Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011*. Citeseer.

Rohit Jain, Himanshu Sharma, and Dipti Misra Sharma. 2016. Using lexical and dependency features to disambiguate discourse connectives in hindi. In *LREC*.

Sudheer Kolachina, Rashmi Prasad, Dipti Misra Sharma, and Aravind K Joshi. 2012. Evaluation of discourse relation annotation in the hindi discourse relation bank. In *LREC*, pages 823–828.

Fang Kong, Hwee Tou Ng, and Guodong Zhou. 2014. A constituent-based approach to argument labeling with joint inference in discourse parsing. In *EMNLP*, pages 68–77.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(02):151–184.

Eleni Miltsakaki, Rashmi Prasad, Aravind K Joshi, and Bonnie L Webber. 2004. The penn discourse treebank. In *LREC*.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Umangi Oza, Rashmi Prasad, Sudheer Kolachina, Dipti Misra Sharma, and Aravind Joshi. 2009. The hindi discourse relation bank. In *Proceedings of the third linguistic annotation workshop*, pages 158–161. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Jianxiang Wang and Man Lan. 2015. A refined end-to-end discourse parser. *CoNLL 2015*, page 17.