
Two semantic features make all the difference in parsing accuracy

**Akshar Bharati, Samar Husain, Bharat Ambati,
Sambhav Jain, Dipti M Sharma and Rajeev Sangal**

Language Technologies Research Center

IIT-Hyderabad

Outline

- Motivation
- Hindi Dependency Treebank
- Dependency parsers
- Experiments
- General observations
- Conclusion

Motivation

- Urgent need of a broad coverage Hindi parser
 - Parser required for almost all Natural Language applications
 - Availability of a Hindi Treebank annotated with dependency relations
 - Results can indirectly check the consistency of treebank annotation
-

Hindi Dependency Treebank

- Hindi is a verb final language with free word order and rich case marking.
- Experiments have been performed on a subset of Hyderabad Dependency Treebank (Begum et al., 2008) for Hindi.

- No. of sentences : 1800
 - Average length : 19.85 (words/sentence)
 - Unique tokens : 6585
-

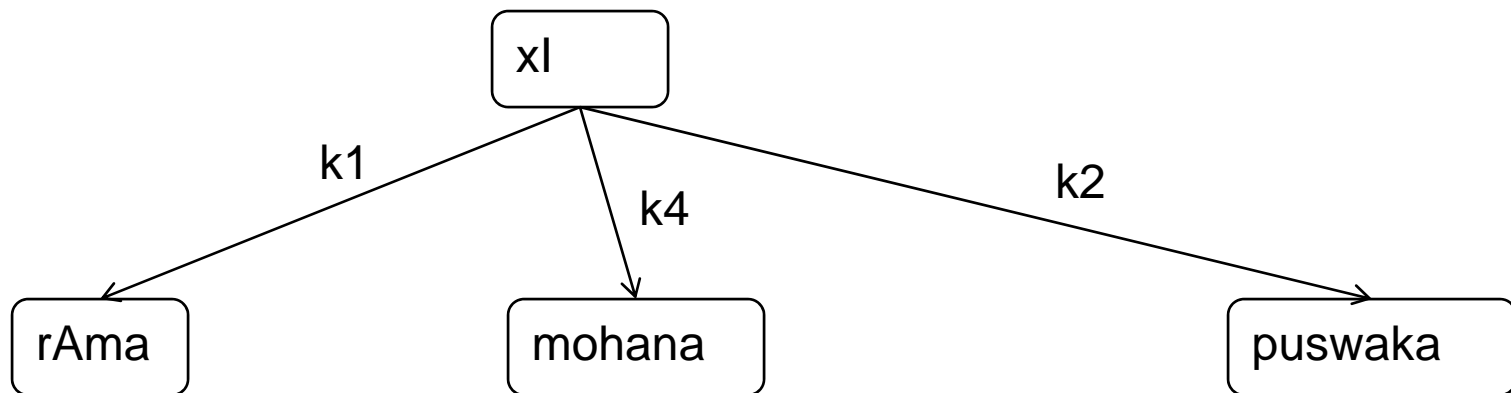
Example

rAma ne mohana ko puswaka xI |

(**rAma** ne) (mohana ko) (**puswaka**) (**xI**) |

'Ram' 'ERG' 'Mohan' 'DAT' 'book' 'gave'

'Ram gave a book to Mohan'



Dependency Parsers

- Grammar-driven parsers
 - Parsing as a constraint-satisfaction problem
 - Data-driven parsers
 - Use corpus for building probabilistic models
 - Parsers Explored (Data-driven)
 - MaltParser
 - MSTParser
-

MaltParser

- Version 1.0.1:
 - Parsing algorithms:
 - Nivre (2003) (arc-eager, arc-standard)
 - Covington (2001) (projective, non-projective)
 - Learning algorithms:
 - MBL (TIMBL)
 - SVM (LIBSVM)
 - Feature models:
 - Combinations of part-of-speech features, dependency type features and lexical features
-

MST Parser

- Version 0.4b:
 - Parsing algorithms:
 - Chu-Liu-Edmonds (1967) (non-projective)
 - Eisner (1996) (projective)
 - Learning algorithms:
 - Online Large Margin Learning
 - Feature models:
 - Combinations of part-of-speech features, dependency type features and lexical features
-

Data

- Training set : 1178 Sentences
 - Development set : 352 Sentences
 - Test set : 363 Sentences

 - Input format
 - CONLL format

 - Input Data
 - Chunk heads

 - Tagset
 - Core tagset (24 tags), a subset of complete tagset (38 tags)
-

Example

rAma ne mohana ko puswaka xl |

(**rAma** ne) (mohana ko) (**puswaka**) (**xl**) |

'Ram' 'ERG' 'Mohan' 'DAT' 'book' 'gave'

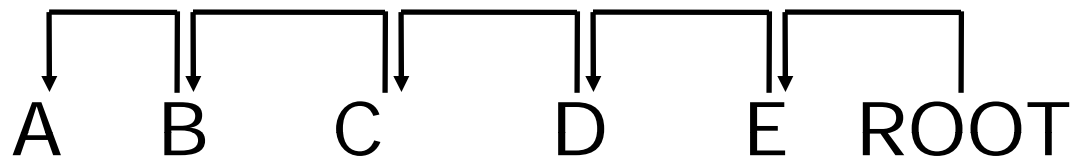
'Ram gave a book to Mohan'

CONLL format:

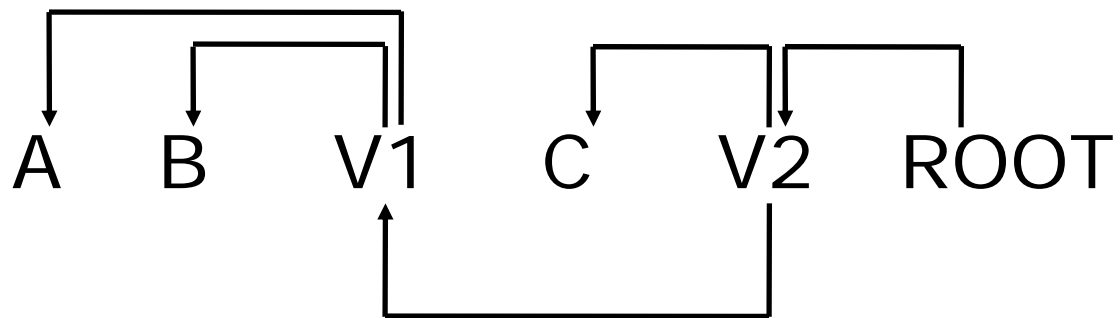
ID	FORM	LEMMA	CPOSTAG	POSTAG	FEATS	HEAD	DEPREL	PHEAD	PDEPREL
1	rAma	rAma	NP	NNP	–	4	K1	–	–
2	mohana	mohana	NP	NNP	–	4	K4	–	–
3	puswaka	puswaka	NP	NN	–	4	K2	–	–
4	xl	xe	VGF	VM	–	0	main	–	–

Baselines

□ Baseline1



▸ □ Baseline2



◀

Results

	Unlabeled Attachment*
Baseline1	46.56%
Baseline2	60.89%

*Correct head-dependent pair, labels on the arc not considered

Experiments

- We begin with two basic hypotheses which we test while tuning the parser.
- For morphologically rich, free word order languages
 - a) High performance can be achieved using vibhakti
 - b) Subject, Object (k1, k2) can be learnt with high accuracy using vibhakti

vibhakti: Generic term for preposition, post-position and suffix

eg: '*ne*' in '*rAma ne*'

Experiments (cont...)

- Experiments-I: Tuning the parser
 - Parsing Algorithms
 - Model Parameters
 - Morpho-syntactic features (FEATS)
 - Feature set
-

Tuning the parser: Parsing Algorithms

□ Malt

■ Nivre

- Arc eager
- Arc standard

■ Covington

- Projective
- Non-projective

□ MST

- Chu-Liu-Edmonds (non-projective)
 - Eisner (projective)
-

Tuning the parser: Model Parameters

□ MST

- Different training-k (k highest scoring trees)
 - K=1 to k<=20
 - K=5 gave the best results.
- Order
 - 1 and 2
 - Order 2 gave better results than order 1

□ Malt

- Tuning SVM model was difficult.
 - Tried various parameters but could not find any pattern
 - CONLL shared task 2007 settings used by same parser for various languages.
 - Turkish settings performed better than others
-

Results

		UA (UC)	LA (LC)	L
MST	Default	<i>83.19 (40.50)</i>	<i>59.25 (8.54)</i>	<i>62.26</i>
	Non-Projective Algorithm, k-5	83.94 (43.53)	60.72 (8.81)	63.33
Malt	Default	<i>84.44 (44.63)</i>	<i>59.10 (9.09)</i>	<i>61.22</i>
	Arc-eager Turkish SVM settings	85.02 (42.98)	59.03 (9.09)	60.97

UA – Unlabeled Attachment

UC – Unlabeled Complete

LA – Labeled Attachment

LC – Labeled Complete

L – Labeled Accuracy

Tuning the parser: Morpho-syntactic features (FEATS)

- **F1** – no feature (default)
 - **F2** – TAM (Tense, Aspect and Modality) labels for verbs and postpositions for nouns
 - **F3** – TAM class for verbs and postpositions for nouns.
-

Example

rAma ne mohana ko puswaka xl |

(**rAma** ne) (mohana ko) (**puswaka**) (**xl**) |

'Ram' 'ERG' 'Mohan' 'DAT' 'book' 'gave'

'Ram gave a book to Mohan'

CONLL format:

ID	FORM	LEMMA	CPOSTAG	POSTAG	FEATS	HEAD	DEPREL	PHEAD	PDEPREL
1	rAma	rAma	NP	NNP	ne	4	K1	–	–
2	mohana	mohana	NP	NNP	ko	4	K4	–	–
3	puswaka	puswaka	NP	NN	0	4	K2	–	–
4	xl	xe	VERB	VM	yA	0	main	–	–

Tuning the parser: Feature set

	MALTParser	MSTParser
Default	FORM POSTAG DEPREL	Basic Uni-gram Features Parent FORM/POSTAG Child FORM/POSTAG Basic Bi-gram Features FORM/POSTAG of parent + FORM/POSTAG of child Basic Uni-gram Features + DEPREL
Extended	FEATS	Basic Uni-gram Features + parent FEATS Basic Uni-gram Features + child FEATS
Conjoined		parent FEATS + DEPREL child FEATS + DEPREL

Results

		UA (UC)	LA (LC)	L
MST	Default	<i>83.94 (43.53)</i>	<i>60.72 (8.81)</i>	<i>63.33</i>
	Extended Features	<i>88.71 (53.72)</i>	<i>64.27 (9.09)</i>	<i>66.67</i>
	Conjoined Features	<i>88.67 (55.10)</i>	<i>69.64 (14.60)</i>	<i>72.62</i>
Malt	Default	<i>85.02 (42.98)</i>	<i>59.03 (9.09)</i>	<i>60.97</i>
	Extended Features	<i>87.56 (54.55)</i>	<i>67.99 (14.88)</i>	<i>70.39</i>

Observations

- Using vibhakti as a feature helps enormously
 - Almost 10% jump in LA and 5% jump in UA
- Very low performance for subject, object
 - Around 50% of data do not have explicit vibhakti.

		k1	k2
MST	P	74.49	53.33
	R	75.35	63.38
Malt	P	75.53	53.01
	R	75.82	61.82

Experiments-II: New Hypothesis

- *Subject-Object confusion cannot arise in a language for human speakers*
 - Exploring right devices which humans use to disambiguate subject-object
 - GNP (Gender, Number, Person) information
 - Minimal semantics
-

GNP-1

- GNP for each lexical item using morphological analyzer
- Appended it to FEATS column of F3.
- **F4**: F3 + GNP

		UA (UC)	LA (LC)	L
MST	F3	88.67 (55.10)	69.64 (14.60)	72.62
	F4	88.03 (52.89)	69.10 (14.60)	72.40
Malt	F3	87.56 (54.55)	67.99 (14.88)	70.39
	F4	86.92 (51.79)	67.17 (16.25)	69.82

GNP-1 (cont...)

- GNP is important for agreement
 - Does help in indentifying relations
 - But agreement in Hindi is not straightforward.
 - Eg: verb agrees with object if subject has a post-position, it might sometime take the default GNP
 - Machine could not learn the selective agreement patterns
 - k1,k2 are worst hit by this feature
-

GNP-2

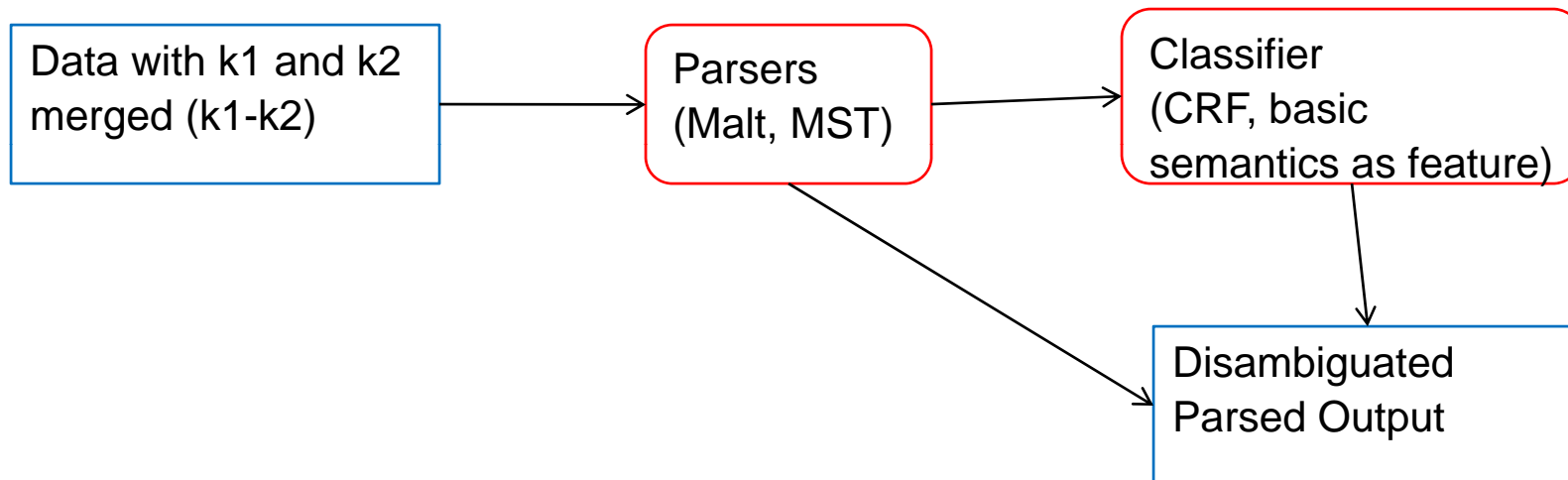
- To prove the importance of this feature in disambiguation
 - We provide the agreement feature explicitly by marking each nodes which agrees with the verb (F5)

		UA (UC)	LA (LC)	L
MST	F3	<i>88.67 (55.10)</i>	<i>69.64 (14.60)</i>	<i>72.62</i>
	F5	88.67 (55.65)	70.93 (16.80)	73.98
Malt	F3	<i>87.56 (54.55)</i>	<i>67.99 (14.88)</i>	<i>70.39</i>
	F5	87.63 (55.10)	69.32 (18.18)	71.86

Minimal Semantics

- *Two basic semantic features can disambiguate majority of subject-object confusion*
 - The semantic features are
 - human-nonhuman
 - animate-inanimate
-

Minimal Semantics



Minimal Semantics: Results

		UA (UC)	LA (LC)	L
MST	F3	<i>88.67</i> <i>(55.10)</i>	<i>69.64</i> <i>(14.60)</i>	<i>72.62</i>
	FM1	89.03 (56.20)	69.93 (14.05)	72.83
Malt	F3	<i>87.56</i> <i>(54.55)</i>	<i>67.99</i> <i>(14.88)</i>	<i>70.39</i>
	FM1	87.56 (53.44)	69.28 (14.88)	71.94

Minimal Semantics: Results

		F3		F3M1	
		k1	k2	k1	K2
MST	P	74.49	53.33	80.79	48.47
	R	75.35	63.38	74.28	65.71
Malt	P	75.53	53.01	76.46	48.98
	R	75.82	61.82	80.42	62.60

- Captures argument structure of the verb
 - Certain verbs take only human subjects etc.,
-

Conclusion

- Isolating crucial clues present in language which help in parsing

 - Different Features
 - vibhakti
 - GNP information
 - Minimal semantics

 - Some hard to learn linguistic constructions
-

Thank you

MaltParser

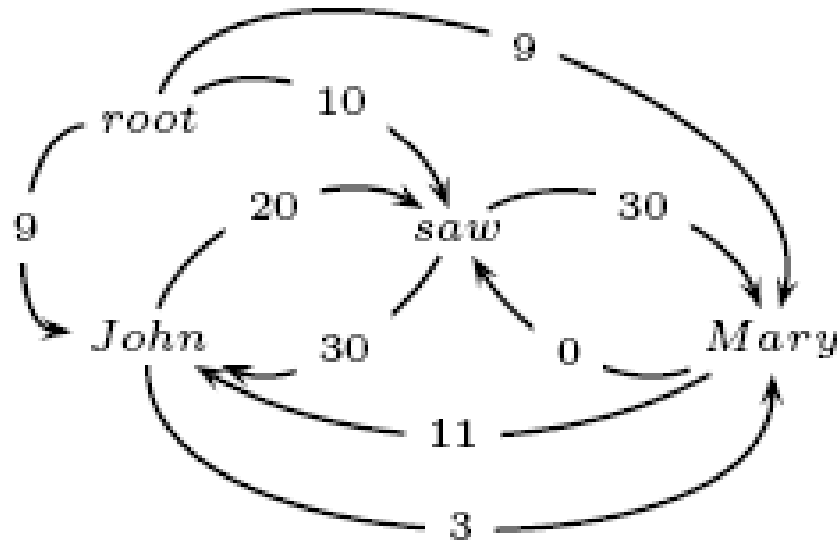
- ❑ Malt uses arc-eager parsing algorithm.
 - ❑ History-based feature models are used for predicting the next parser action.
 - ❑ Support vector machines are used for mapping histories to parser actions.
 - ❑ It uses graph transformation to handle non-projective trees.
-

MST Parser

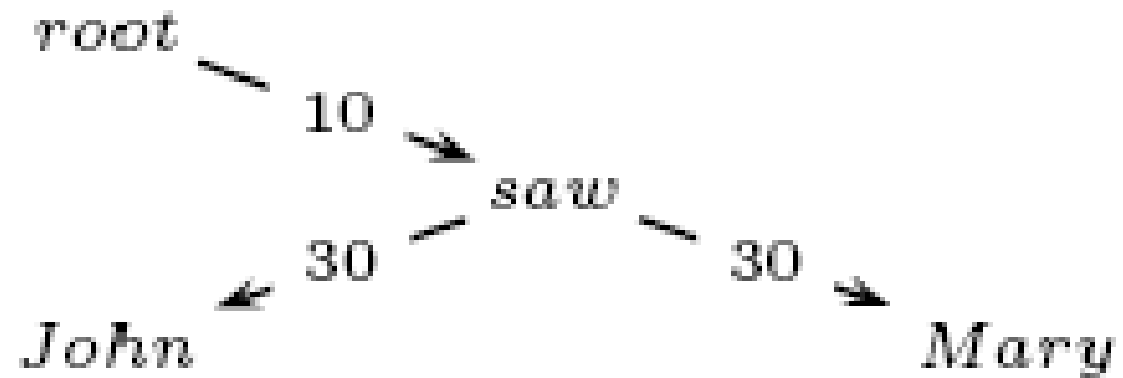
- ❑ Formalizes dependency parsing as search for Maximum Spanning Tree (MST) in weighted directed graphs.
 - ❑ Constructs a weighted complete directed graph for the sentence and connects all nodes to dummy **root** node.
 - ❑ Constructs an MST out of this using Chu-Liu Edmonds algorithm. (Chu and Liu, 1965; Edmonds, 1967)
 - ❑ MSTparser uses online large margin learning as the learning algorithm.
-

MST Parser

- John saw Mary.



MST Parser



General Observations

- MST outperforms Malt in similar conditions.
 - Best LA result for MST and Malt are 69.64% and 67.99% respectively
 - Malt performance can be improved by tuning the SVM parameters
 - MST performs consistently well, in identifying the root of the tree and conjunct relations.
 - MST is far better in identifying longer dependency arcs, whereas Malt does better with shorter ones.
 - for distance >7 the f-measure for Malt is $\sim 62\%$, for MST it is $\sim 65\%$.
-

General Observations

- Overall performance of both the parsers for LA is low, 67.99% and 69.64% for MST and Malt respectively.

 - Reasons could be
 - Training size
 - 1200 sentences for training
 - But training size alone is not good criteria for low performance (Hall et al., 2007)
 - Type of tags
 - Syntactico-semantic tags
 - Learning such tags is difficult (Nivre et al., 2007a)
 - Non-projectivity
 - Data has around 10% non-projective trees.
-

General Observations

- Investigate learning issues in building a Hindi parser using a dependency treebank
 - Discover/revalidate useful learning features
 - To bring out specific problems (ambiguity)
 - Can some of the problems be solved?
-

□ TAM Class

- TAM labels in Hindi constraint the postpositions which can appear on the subject or object
- TAM labels which apply similar constraints can be grouped into a class
- wA_hE, 0_rahA_hE Vs wA_hE, yA