Converting Constituency Structures to Dependency Structures for Sanskrit: Linguistic Issues

Pawan Goyal[1] and Amba Kulkarni[2]

[1]Department of Computer Science and Engineering,
Indian Institute of Technology, Kharagpur
[2]Department of Sanskrit Studies,
University of Hyderabad

February 8th, 2014

Outline

Outline

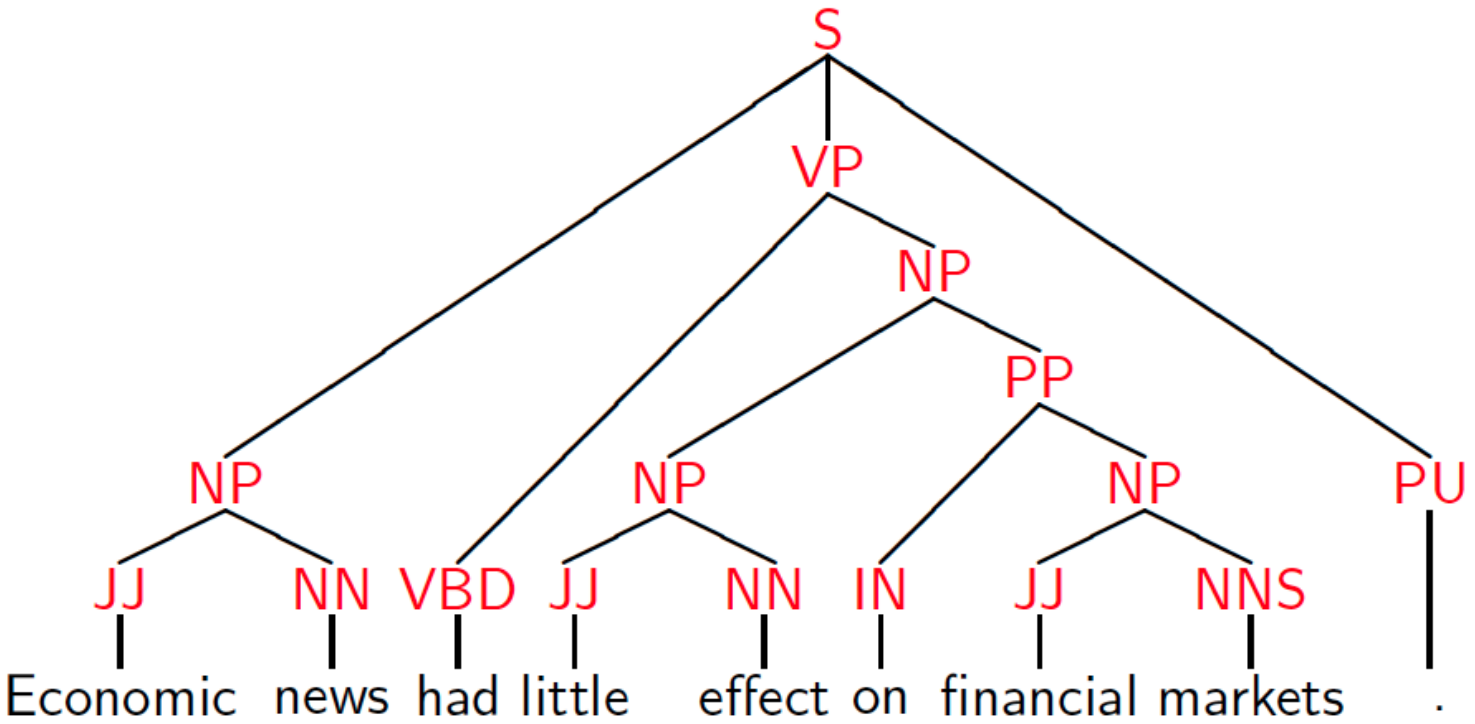
Introduction

- Verbal understanding of any utterance requires the knowledge of how words in that utterance are related to each other.
- Constituency and dependency parsers are among the main frameworks to represent this knowledge as a parse tree.

Constituency Grammars

- View sentence structure in terms of the constituency relation.
- The constituency relation derives from the subject-predicate division, based on term logic.
- Basic clause structure is understood in terms of a binary division of the clause into subject (noun phrase NP) and predicate (verb phrase VP).
- It was shown that such rules could be extended for analyzing compounds as well as derivational morphology for Sanskrit.

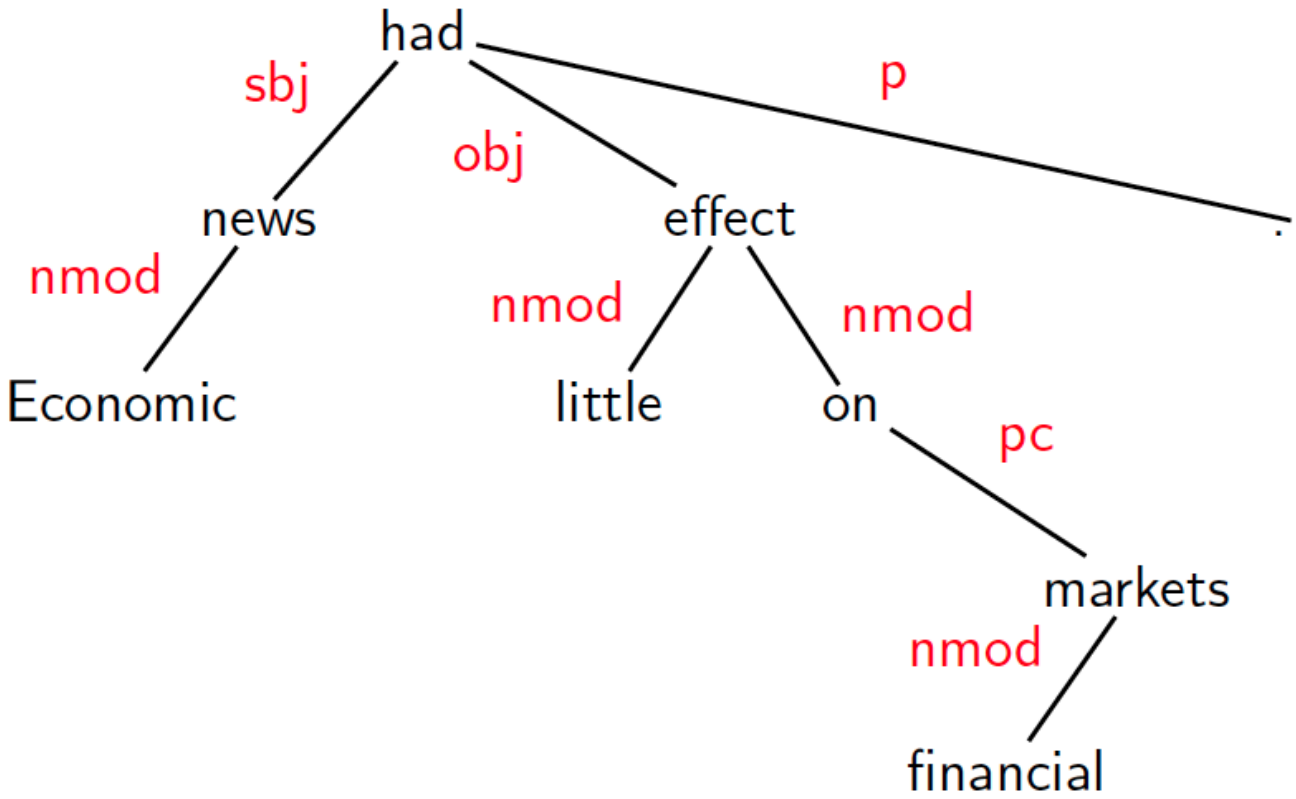
Constituency Grammars: Example



Dependency Grammars

- Basic ideas for the dependency relations come from the *kāraka* relations in *Pāṇini*'s grammar.
- A dependency parse is modeled as a directed tree with nodes representing the words and edges representing the possible relations between them.

Dependency Grammars: Example



Motivation

Main Objective

Do constituency structures preserve information about dependency relation between words?

Our Hypothesis

If it does, we can use an algorithm to convert a constituency structure to dependency structure.

Outline

Dataset

Origins

- The dataset has its origin in Apte's student guide.
- The work was initiated in 1986 by Brendan Gillon, who assigned a syntactic parse to the prose exercise sentences from Apte's Student Guide.
- In 1991, Brendan Gillon transferred the material from a paper format to an electronic format, making revisions.

Dataset

Example Sentence

```
Example{3}
Source{1.1.3 (P) <U 4.5.3>} % Apte{7,3}
Parse
[S [INJ haa ] [ADV katham ]
   [NP1s [NP6 (mahaaraaja<Dazarathasya) ] (dharma<daaraa.h) ]
   [VP 0 [NP1 (priya<sakhii) [NP6 me ] [NP1 Kauzalyaa ] ] ] ]
Gloss{Oh, how is it that the legal wife of King Dazaratha is my
dear friend Kauzalyaa}
% Better: How is my dear friend K, the wife of ...
Comment{copula: covert: predicational: NP1s VP }
```

Dataset

Towards Abstract Syntax

- In 2004, Gerard Huet re-engineered the document in order to parse it mechanically.
- He verified its correct syntactic structure after typographical corrections.
- He devised an abstract syntax to formalize this constituency structure.

IDataset

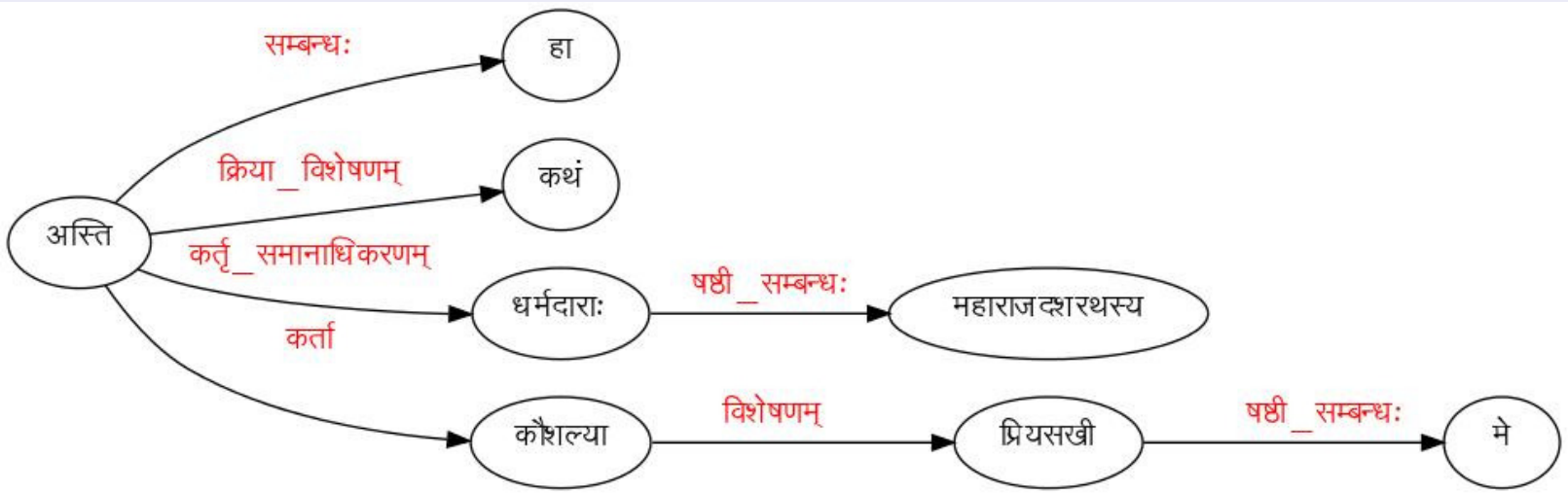
Abstract Syntax

```
list Tag_tree.syntax =
[S
  [INJ ("haa", 3); ADV [("katham", 4)];
   NP
    ([Case 1; Role Subject],
    [NP ([Case 6], [N (Compound (Stem <mahaaraaja>, Stem <Dazara
     N (Compound (Stem <dharma>, Stem <daaraa.h>), 6)]);
   VP0
    [NP
      ([Case 1],
      [N (Compound (Stem <priya>, Stem <sakhii>), 7);
       NP ([Case 6], [N (Stem <me>, 8)]);
       NP ([Case 1], [N (Stem <Kauzalyaa>, 9)])]);
     NIL 10]]]
```

Dataset

Tagging with Dependency Relations

The same dataset was tagged with dependency relations at University of Hyderabad.



Outline

Algorithm

Notion of 'head'

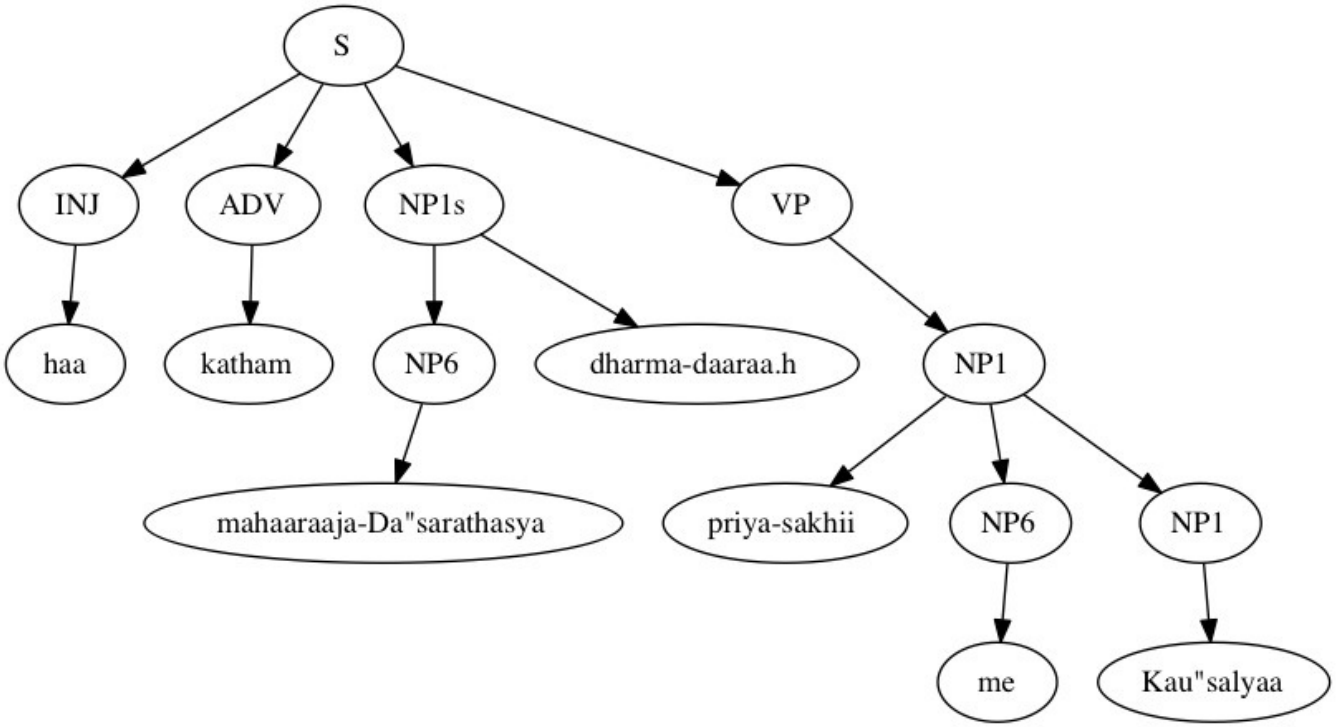
- The notion of 'head' is very important for both the constituency and dependency structures.
- Each phrase structure has a head, that determines the main properties of the phrase and a head has several levels of projection.
- In a dependency structure, on the other hand, the head is linked to its dependents.

Algorithm

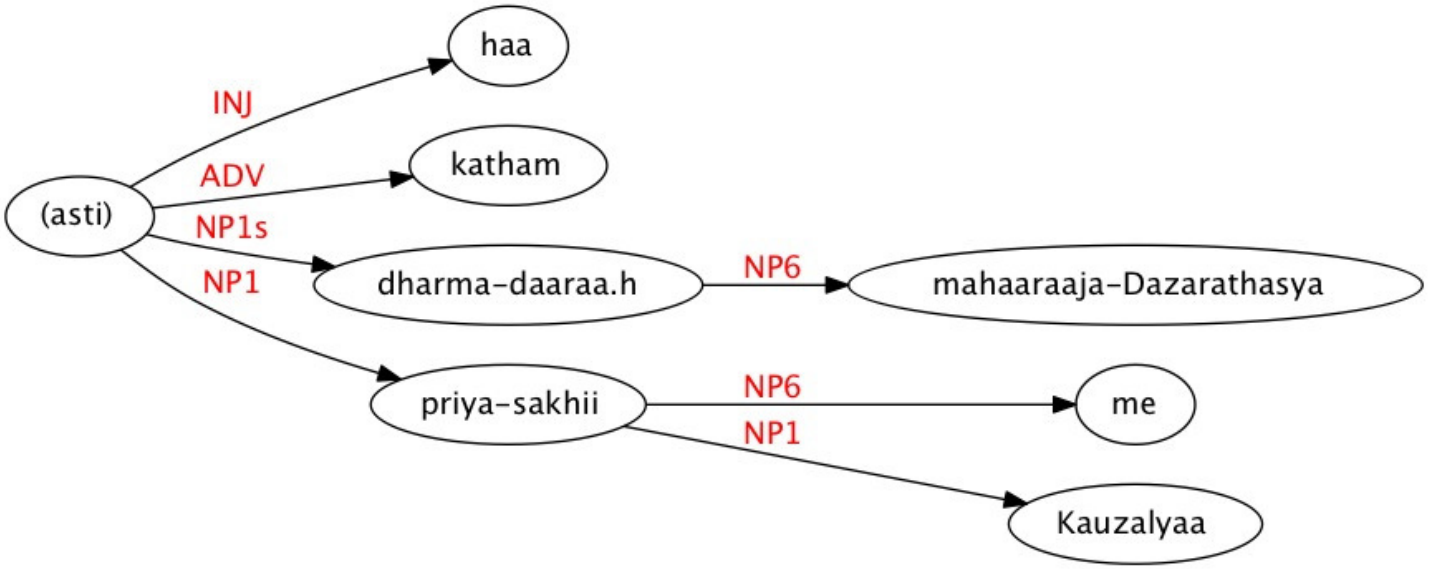
- The head of VP is the ROOT node in the dependency tree.
  - If CNJ with 'yadi' is present, then it is the head.
  - If AUX is present, AUX is the head.
  - If PRT with 'iti' is present, then it is the head.

  (The later rule is stronger than the previous.)
- All the XPs within VP are dependent on the ROOT.
- If S is the parent of VP, then all the XPs which are children of S are also dependent on this ROOT.
- This definition is used recursively to determine head-modifier relation in the dependency graph.

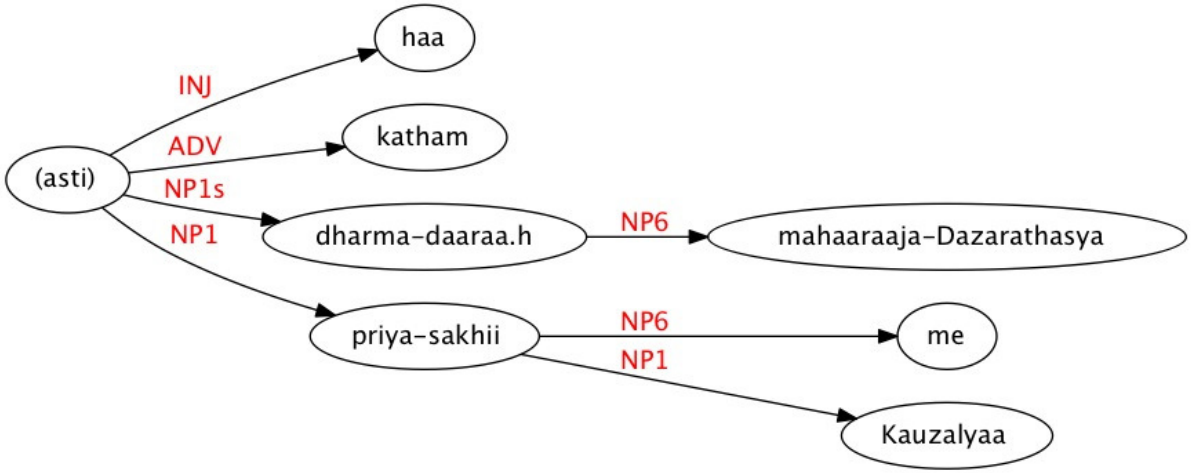
Example: Constituency Parse

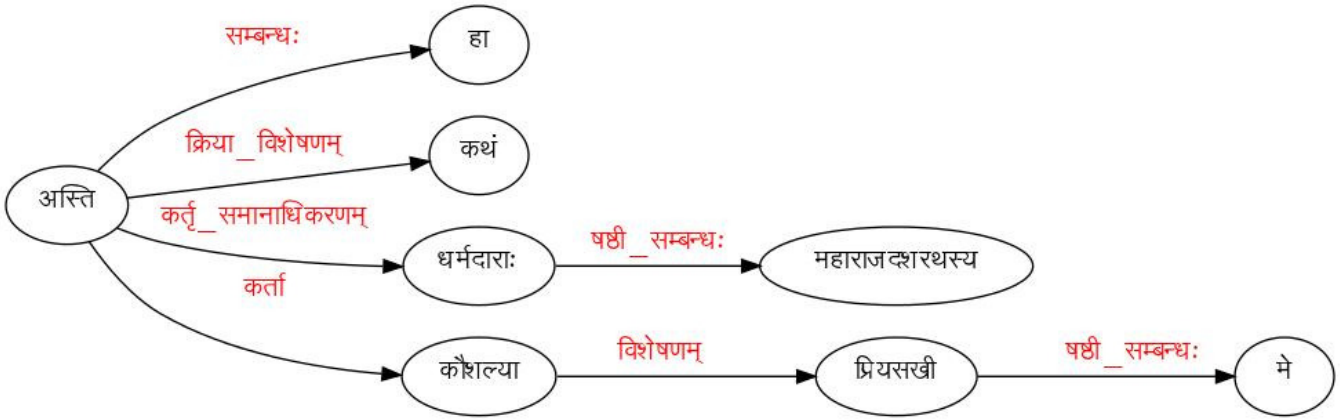


Dependency Graph generated by the Algorithm



Comparing with the Gold Standard Dependency





Dislocations

Dislocation information was was missing from the tree notations but was provided in the comments.

```
Parse - original
[S [NP6 (saartha<vaahasya) [NP6 Arthapate.h ] ]
   [NP1s Vimardaka.h ]
   [VP 0 [NP1 [AP1 (bahis<caraa.h) ] __ praa.naa.h ] ] ]
Gloss{Vimardaka is the external life of the merchant Arthapati.
Comment{copula: covert: predicational: NP1s VP
left extraposition from VC (pred NP) of NP6 within MC.
apposition}
```

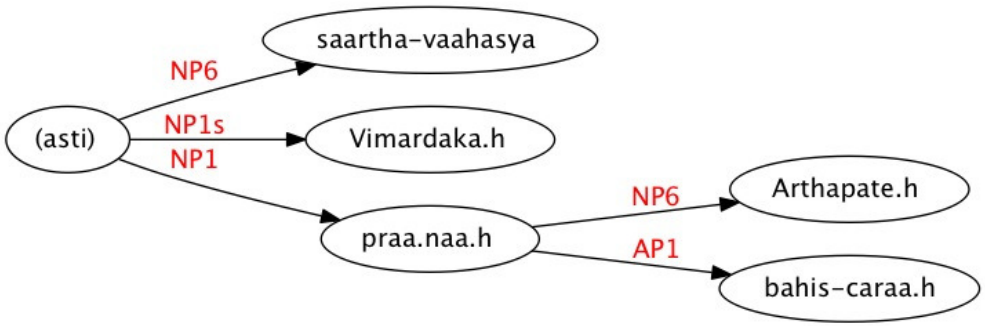
Dislocations

- We used '!' and '$' to indicate the dislocation.
- '!' indicates the position from where a component is dislocated,
- '$' gives the dislocated component.

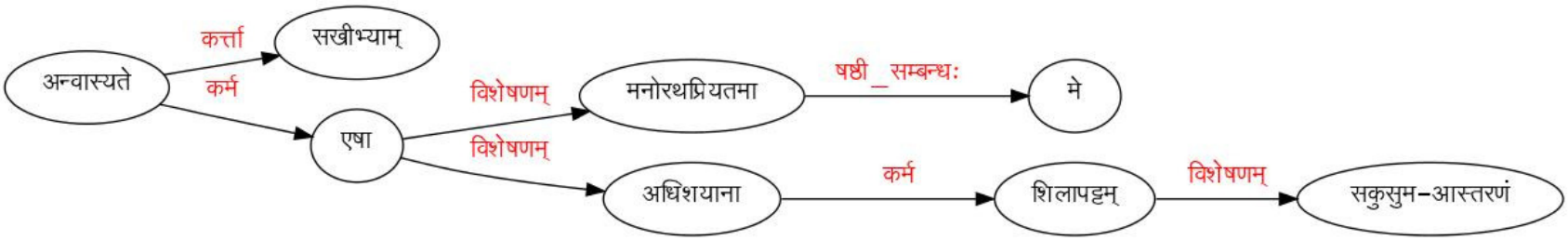
```
Parse - original
[S [NP6 (saartha<vaahasya) [NP6 Arthapate.h ] ]
   [NP1s Vimardaka.h ]
   [VP 0 [NP1 [AP1 (bahis<caraa.h) ] __ praa.naa.h ] ] ]
Parse - modified
[S [NP6 (saartha<vaahasya) [NP6 Arthapate.h $1] ]
   [NP1s Vimardaka.h ]
   [VP 0 [NP1 [AP1 (bahis<caraa.h) ] !1 praa.naa.h ] ] ]
```

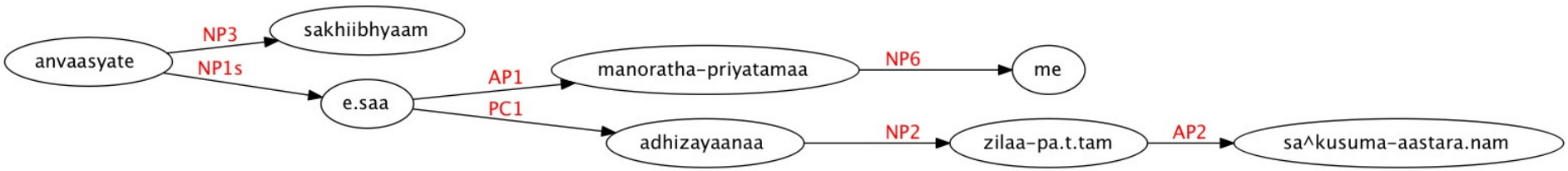
Handling Dislocations

```
Parse - modified
[S [NP6 (saartha<vaahasya) [NP6 Arthapate.h $1] ]
   [NP1s Vimardaka.h ]
   [VP 0 [NP1 [AP1 (bahis<caraa.h) ] !1 praa.naa.h ] ] ]
```

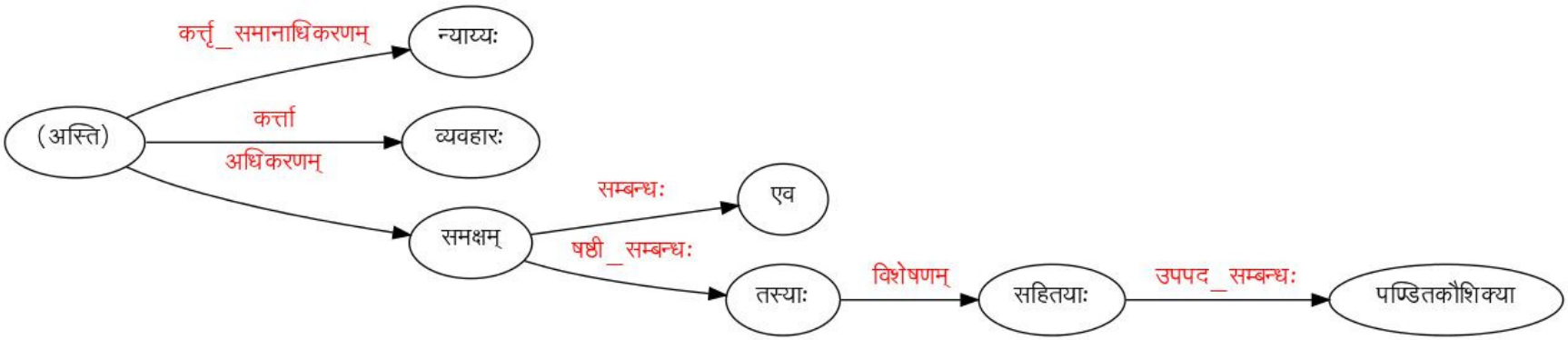


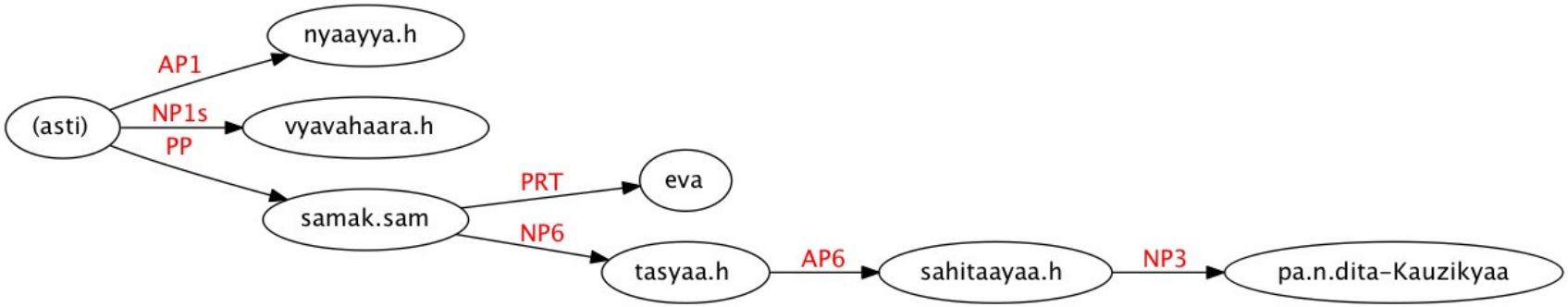
Some More Examples





Some More Examples



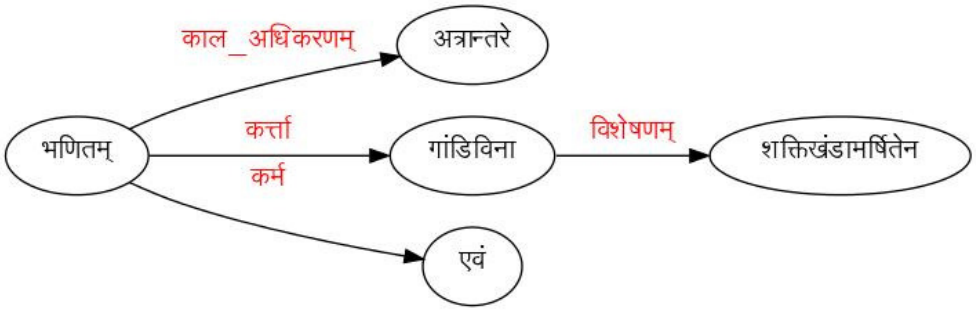


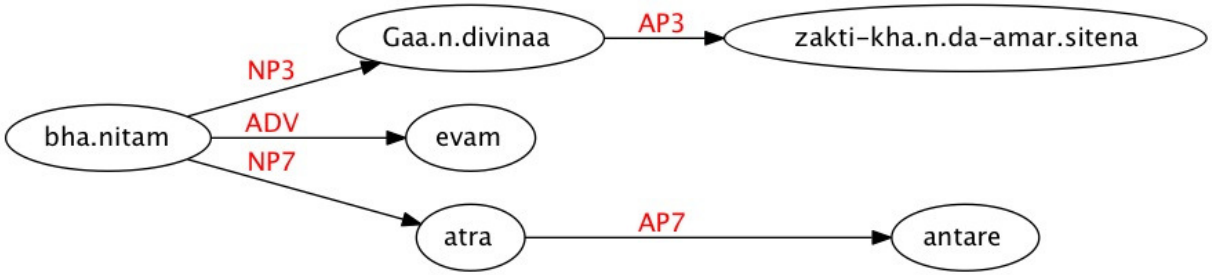
Outline

Results

- Evaluation was done by comparing the parses produced by our algorithm with respect to the gold standard dependency graphs.
- We match all the pair of words in the two graphs (i.e. the number of relations)
- Out of 232 cases, we found 97 cases with exact match. For the rest of the cases,
  - In 95 cases, some of the relations do not match.
  - In 40 cases, number of words in dependency and phrase-converted graph are different.

Analyzing Mismatches: Word Mismatch





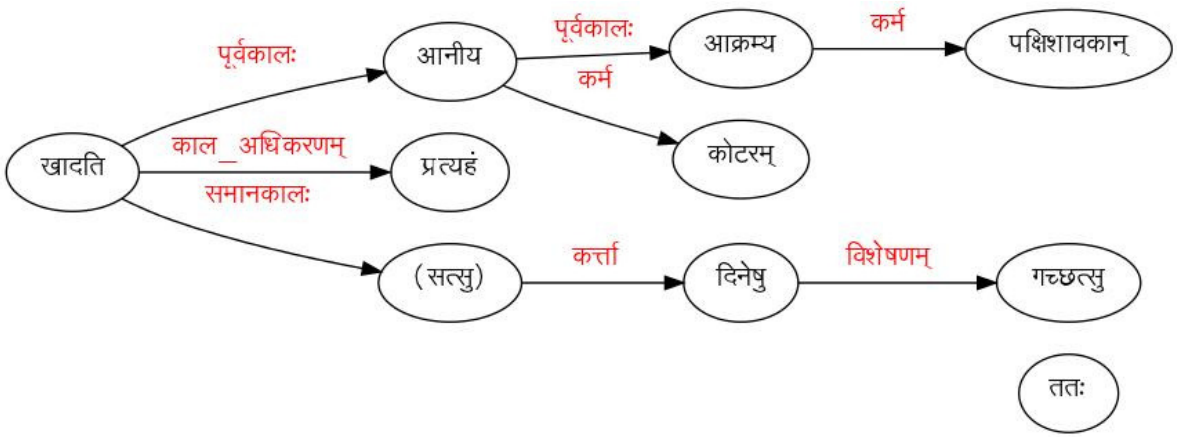
Other such words: *kadcit*, *yadyapi*, *tathāpi*, *athavā*, *kathamapi* etc.

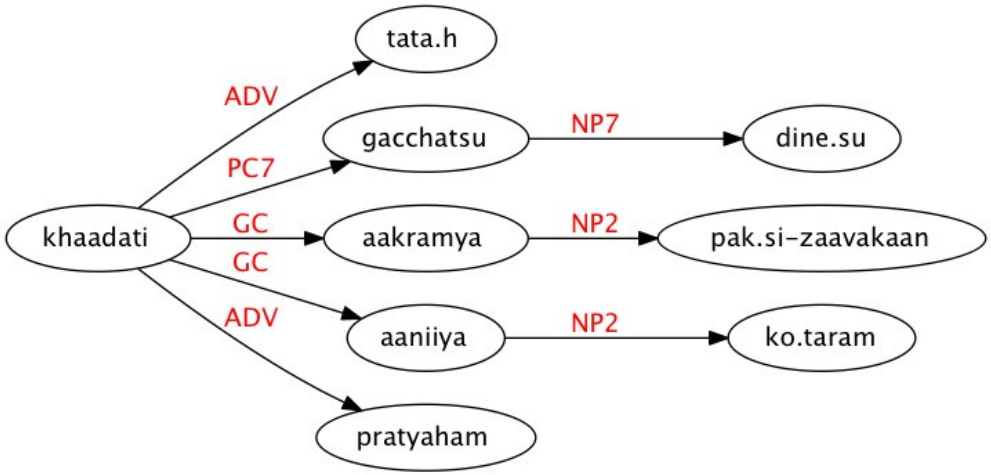
Analyzing Mismatches: Missing Relations

```
Parse
[S [ADV tata.h ]
   [AC [NP7 dine.su ] [PC7 gacchatsu ] ]
   [GC [NP2 (pak.si<zaavakaan) 1] aakramya ]
   [GC [NP2 [E] 1] [NP2 ko.taram ] aaniiya ]
   [ADV pratyaham ]
   [VP [NP2 [E] 1] khaadati ] ]
Gloss{Then, as the days went by, he overpowered the baby birds,
took them to his hallow and ate them, day after day.}
```

Relation between *ākramya* and *ānīya* is not marked.

Analyzing Mismatches: Missing Relations





This is also an example, where ellipsis *satsu* is introduced in the dependency.

Questions?

Thank you for your kind attention.