CS689: Computational Linguistics for Indian Languages
Dependency Parsing

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

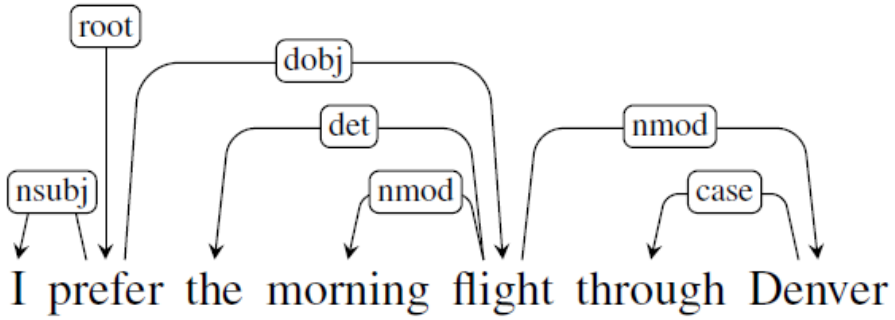
Computer Science and Engineering,
Indian Institute of Technology, Kanpur
http://web.cse.iitk.ac.in/~cs689/

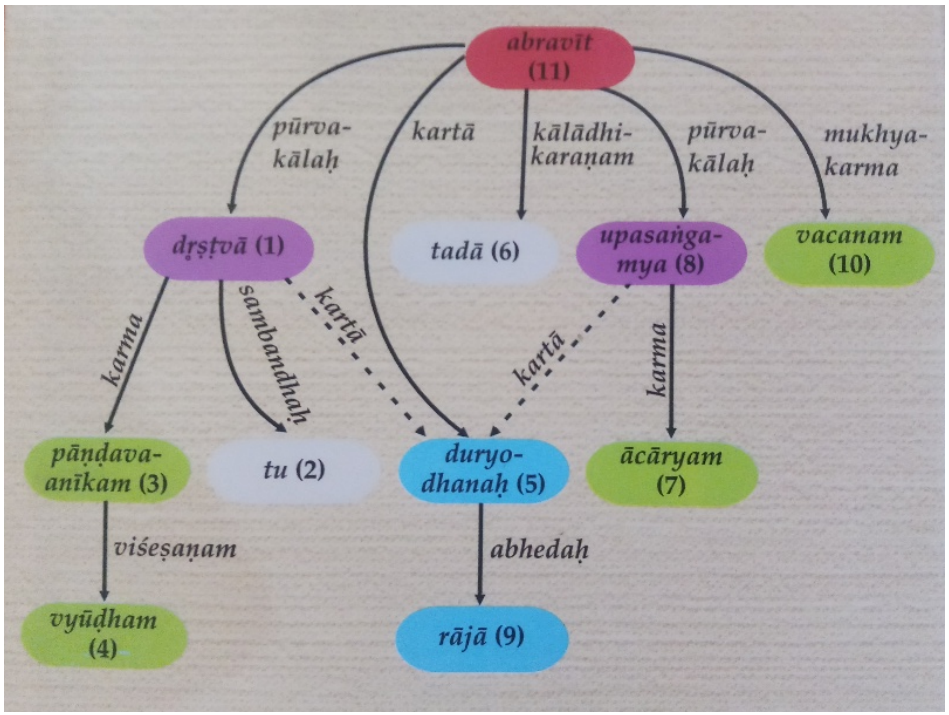
2nd semester, 2023-24

Tue 10:30–11:45, Thu 12:00–13:15 at RM101/KD102

Dependency Parsing

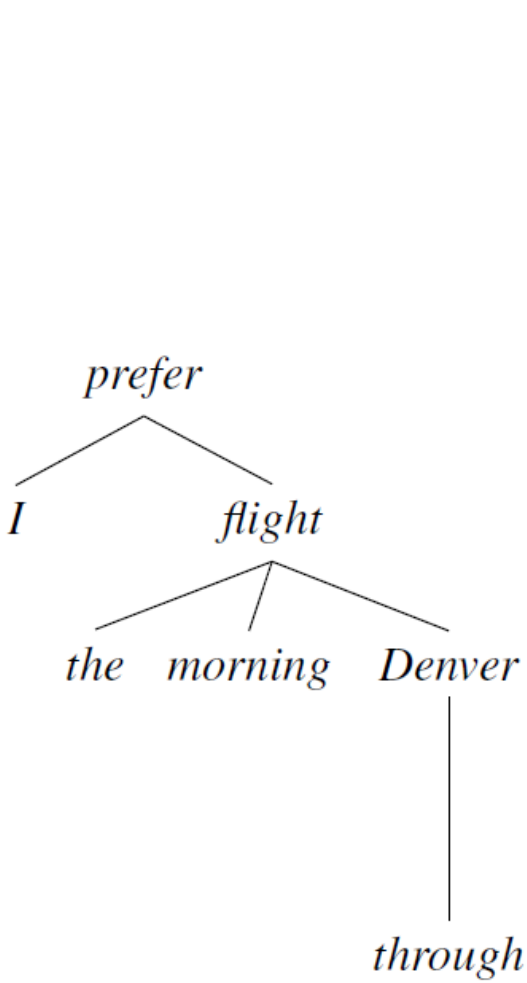
- Dependency Parsing aims to find *directed relationships* between words
- A root word for every sentence

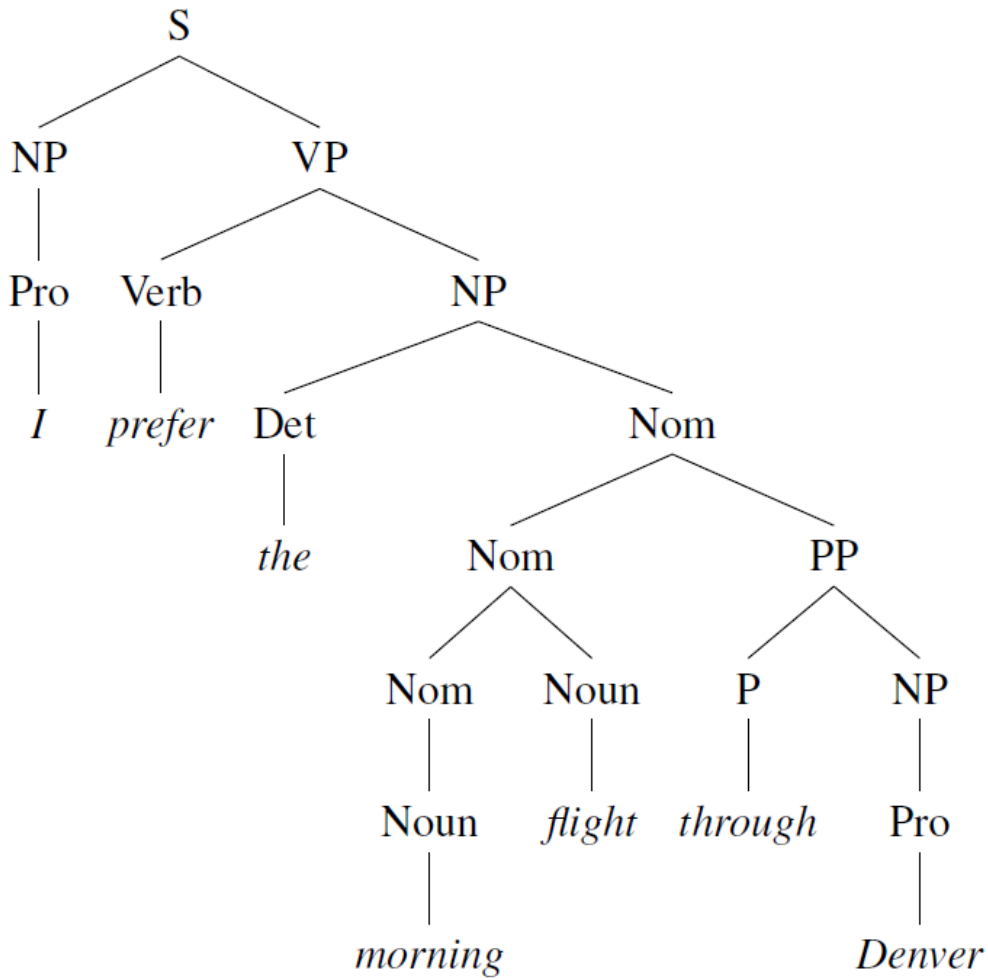




Dependency versus Constituency

- Dependency tree versus constituency tree





Dependency Tags

- Universal Dependency relations or UD dependency tags
  - Based on subject (nsubj), object (dobj, iobj), modifier (nmod, amod), verb describer (case), etc.
- In Indian languages
  - Based on kāraka theory

Kāraka Theory

- Relations of a noun to the verb: case
- Kartṛ kāraka (कर्तृकारक): subject (nominative case)
- Karma kāraka (कर्मकारक): object (accusative case)
- Karaṇa kāraka (करणकारक): instrument (instrumental case)
- Sampradāna kāraka (सम्प्रदानकारक): recipient (dative case)
- Apādāna kāraka (अपादानकारक): separation (ablative case)
- Adhikaraṇa kāraka (अधिकरणकारक): locus, both temporal and spatial (locative case)
- Relations of a noun to another noun
- Sambandha (सम्बन्ध): possessive relation (genitive case)
- Other relations
- Sambodhana (सम्बोधन): in conversation (vocative case)
- Vibhakti (विभक्ति) or case marker generally indicates the case
  - In order, {1-5, 7}, {6 (genitive)}, {1 (vocative)}

Kāraka: Syntax and Semantics

- Principle of ākāṅkṣā
- All in kartṛ-kāraka due to independence
  - Devadattaḥ pacati: Devadatta cooks
  - Sthālī pacati: Vessel cooks
  - Edhāḥ pacanti: Logs cook
- Locus as most desirable object
  - Hariḥ vaikuṇṭham adhiśete: Hari sleeps in Vaikuntha
  - Muniḥ śilāpaṭṭam adhitiṣṭhati: Sage sits on stone slab
- Upapada-vibhakti
- World of language and real world may not correspond one-to-one
- Kāraka mostly follows syntax
  - Makes it easier for automated processing

Indian Language Dependency Tags: Kāraka

- Based on kāraka and śābdabodha theories
- Kartā: doer/agent/subject
  - Anubhava-kartā: मुझ को राम बुद्धिमान् लगता है।
  - Prayojya-kartā and Prayojaka-kartā: राम रावण से अधर्म करवाता है।
- Karma: object/patient/goal/destination
  - Mukhya-karma and Gauṇa-karma: राम सीता को अयोध्या ले आया।
- Karaṇa: instrument
- Sampradāna: beneficiary
- Apādāna: departure/separation/transformation/source
  - दूध से दही, मिट्टि से घट
- Adhikaraṇa: locus
  - Abhyantara: घर मे बैठा है
  - Aupaśleṣika: छत पर बैठा है
  - Sāmīpya: भिखारी द्वार पर बैठा है
- Sambandha: relation/possession
- Sambodhana: calling/addressing

Indian Language Dependency Tags: Non-kāraka

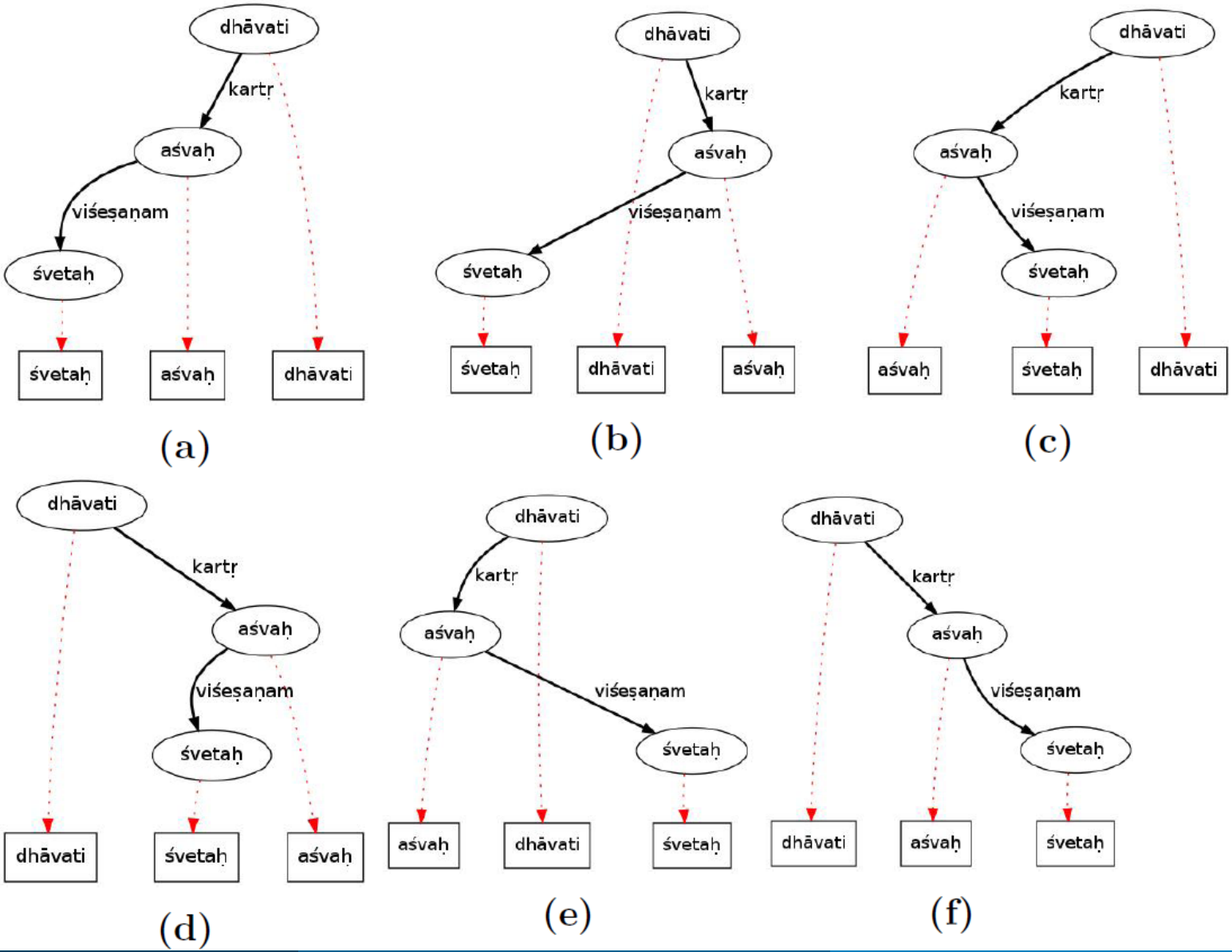
- Upamāna: analogy
  - चांद जैसा मुख
- Tulanā: comparison
  - चांद जैसा कुछ भी नेहि
- Kriyāviśeṣaṇa: adverb
  - हाथी धीरे धीरे चलता है।
- Viśeṣaṇa: adjective
  - सुन्दर किला
- Tīvrātādarśī: intensifier
  - बहुत सुन्दर किला, बहुत सुन्दर गाती है
- Prayojana: motive
  - वह पढने के लिये विद्यालय जाता है।
- Hetu: cause
  - वह काम से कानपुर आ हीं है।
- Samuccaya: connectors
- Vākyāṁśayojaka: connectors bewteen clauses
  - मै घर गया था ऐसा उन्होने बोला।
- Around 60 such relations

Projectivity

- Principle of `sannidhi`
- Projectivity principle or adjacency principle constraints type of dependency parse trees
- An edge from a head to a dependent word is *projective* if path from head to all intermediate words lies within this edge
- A dependency tree is *projective* if all its edges are projective
- Alternatively, a sentence is *projective* if and only if a dependency tree can be drawn such that every node can be projected by a vertical line onto the word in the sentence without crossing another projection or dependency edge
- Useful constraint for free-order languages
  - Sanskrit is "clause-internal" free-order

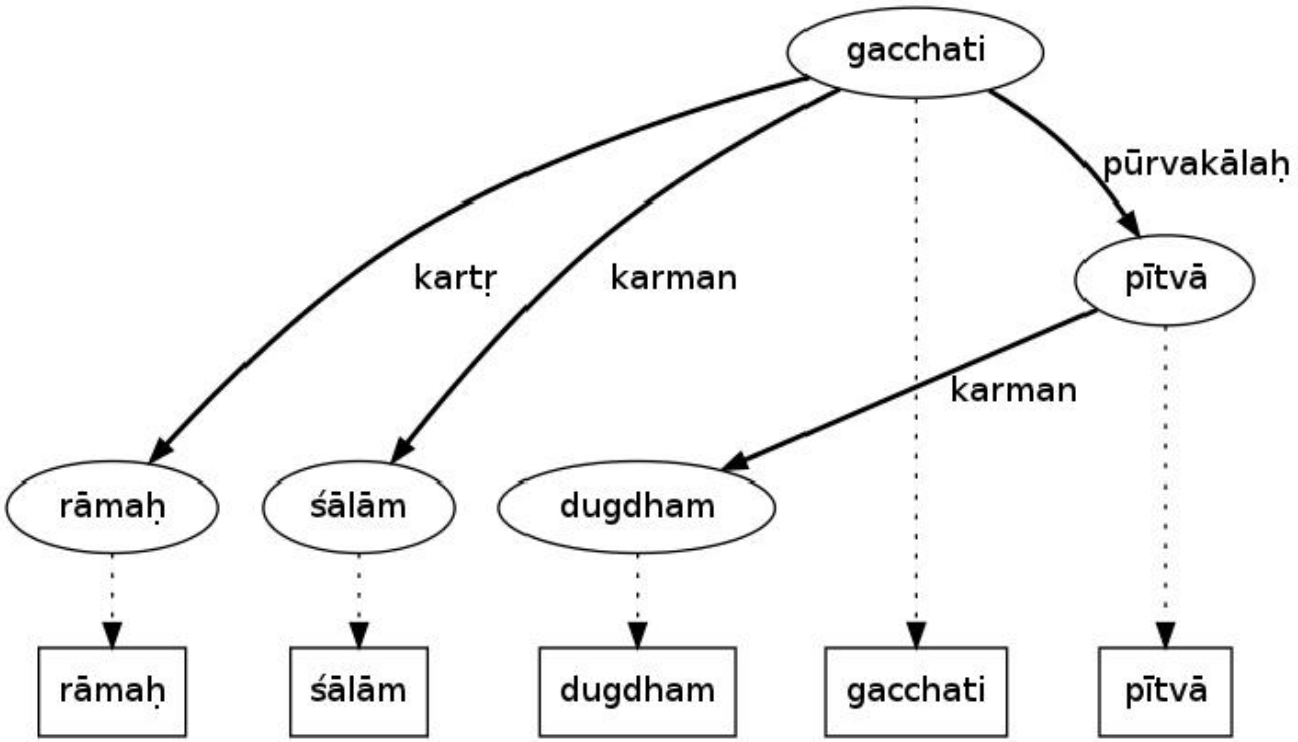
Example

- śvetaḥ aśvaḥ dhāvati (श्वेतः अश्वः धावति) has $3! = 6$ permutations



Example with Two Verb Forms

- rāmaḥ dugdhaṃ pītvā śālāṃ gacchati (रामः दुग्धं पीत्वा शालां गच्छति)
  can be re-written as
  rāmaḥ śālāṃ dugdhaṃ gacchati pītvā (रामः शालां दुग्धं गच्छति पीत्वा)

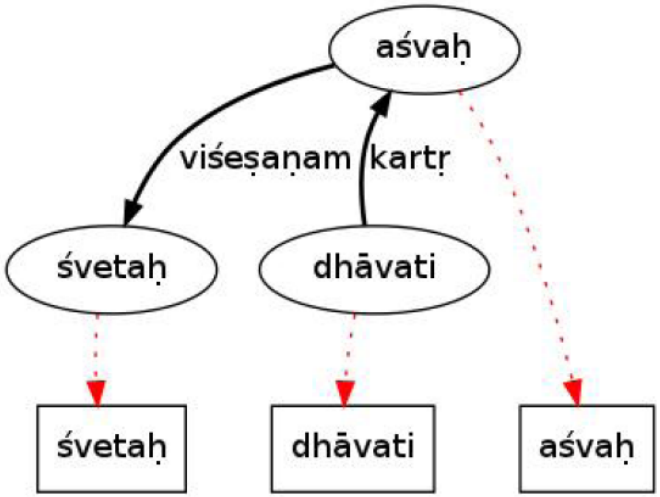


Planarity

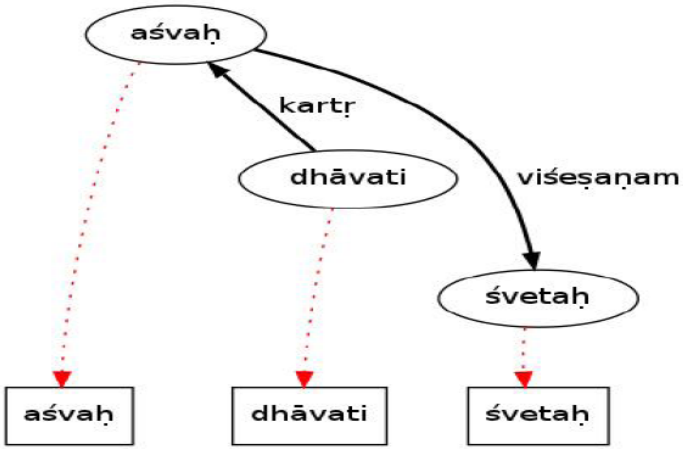
- Planarity principle or weak non-projectivity principle loosens the projectivity principle
- A dependency tree is *planar* if it does *not* have two edges $w_i \leftrightarrow w_j$ and $w_k \leftrightarrow w_l$ such that $i < k < j < l$
- Alternatively, a sentence is *planar* if and only if a dependency tree can be drawn such that no edge crosses another edge
- Every projective tree is planar but not vice versa
- Edges are always drawn *above* the sentence
- Projectivity does not allow any upward directed edge
- Planarity allows upward directed edges

Back to Examples

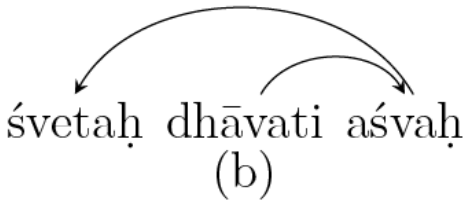
- The sentence श्वेतः धावति अश्वः is planar
- The sentence रामः शालां दुग्धं गच्छति पीत्वा is not

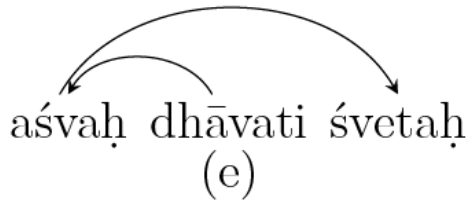


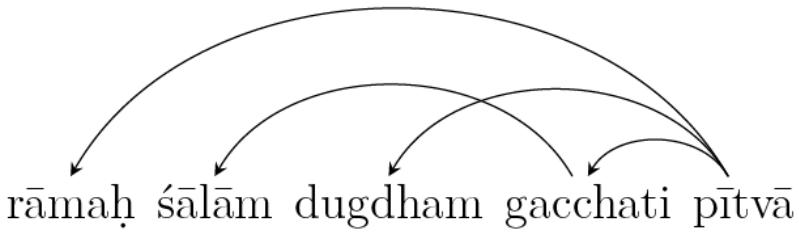
(a) *graph for sentence (b)*



(b) *graph for sentence (e)*







Difference in Sentences

- The sentence रामः दुग्धं पीत्वा शालां गच्छति has `utthita ākāṅkṣā` (*mutual expectancy*)
- The sentence श्वेतः अश्वः धावति has `utthāpya ākāṅkṣā` (*unilateral expectancy*)
- Violation of `sannidhi` makes a tree non-planar

Semantic Restrictions

- Principle of yogyatā
- Word meanings should be mutually compatible with reference to that relation
- Noun and verb lakṣaṇa charts
- Defines compatibility
- Verb jānā (जाना) in Hindi: *to go*
- Suffix se (से) can denote both apādāna and karaṇa
  - rāma kānapura se kolakātā ṭrena se jātā hai (राम कानपुर से कोलकाता ट्रेन से जाता है)
- Kāraka chart with lakṣaṇa for jānā

Kāraka	Necessity	Case Marker	Semantic Constraint
kartṛ	mandatory	0 / ko	–
karma	mandatory	0	–
karaṇa	desirable	se	vehicle
apādāna	desirable	se	*not* vehicle

Significance of Meaning

- Word śakti *significative power* has three types:
  - abhidhā or *primary meaning*
  - lakṣaṇā or *implication*
  - vyañjanā or *suggestion*
- यानं ग्रामं गच्छति (yānaṃ grāmaṃ gacchati) has two meanings
  - Vehicle goes to village
  - Village (people) goes to (see) vehicle
- By default, mukhyārtha (मुख्यार्थ), i.e., abhidhā or primary meaning, has to be understood
  - Filters away some grammatically possible parse trees
- Using only morphology, yānaṃ and grāmaṃ could have been *adjectives* (viśeṣaṇa) of each other as well
- Ruled out using adjective lakṣaṇa charts
  - Viśeṣaṇa should have some *quality* guṇa
- Morphology of gacchati as saptamī vibhakti of gacchat ("going") yields no grammatically correct parse

Secondary Meanings

- Lakṣaṇā or implication is invoked only in case of ānarthakyadoṣa (आनर्थक्यदोष), i.e., when primary meaning makes no sense
- vahninā siñcati (वह्निना सिञ्चति) makes no sense with literal meaning of "fire"
  - May mean "anger", "scolding", etc.
  - Thus, it is grammatically correct and should be parsed
- payasā siñcati (पयसा सिञ्चति) does not require any implication and can simply mean "sprinkle with water"
- Vyañjanā depends completely on context and discourse and, hence, cannot be handled at parse level

Dependency Parsing Algorithms

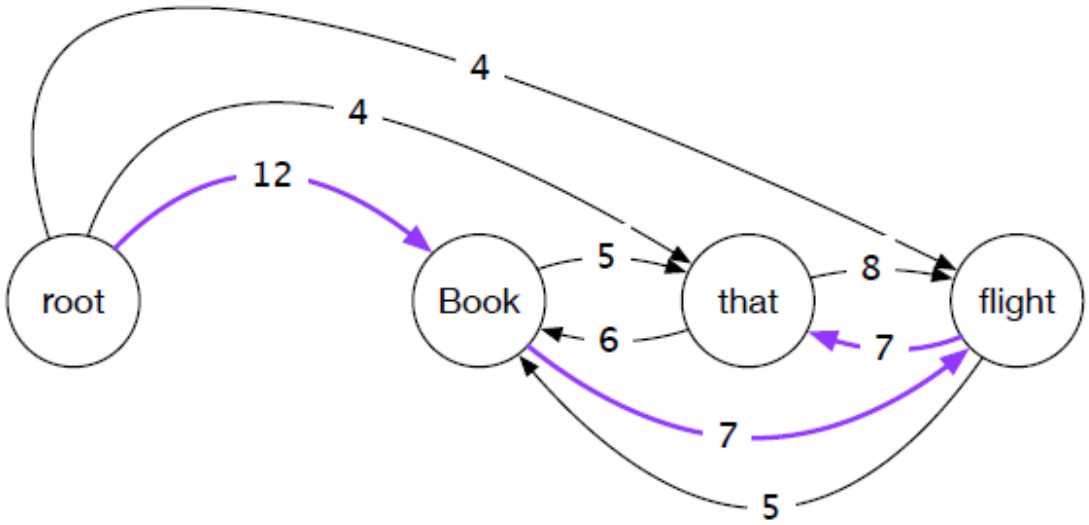
- Two main types
- Transition-based
  - More suitable for rigid word-order languages
- Graph-based
  - Works well for free word-order languages

Graph-based Dependency Parsing

- Nodes are words
- Edges are *directed* relationships between words
- A special root node
- Each edge has a *score* that depends on how good the relationship is
- Any *compatible* spanning tree is a parse
- Every node has *exactly one* incoming edge
  - Except *root*
  - Every word satisfies *exactly one* role in the sentence
- No cycles or self-loops
  - No word satisfies its own requirement
- For a sentence with n words, exactly n edges
- *Best* parse is *spanning tree* that *maximizes* total score of edges
- Run minimum spanning tree algorithm after *reversing the order* of weights on edges

Example

- "Book that flight"



- Spanning tree with total weight $12 + 7 + 7 = 26$ is the best
  - root $\rightarrow$ book; book $\rightarrow$ flight; flight $\rightarrow$ that

Scores on Edges

- Features: POS tags, morphology, *chunks*
- Grammar rules
- Learning from corpus data
- Sentences with tags on words and corresponding best parses
- Dependency relations are also marked in the reference parse
- Training similar to perceptron training rule
- Start with random scores on edges
- Get the current best parse B
- Find edges in B that are not in the reference parse R
- Reduce their weights at some *learning rate*
- For edges in B that are in R, do nothing

Morphology

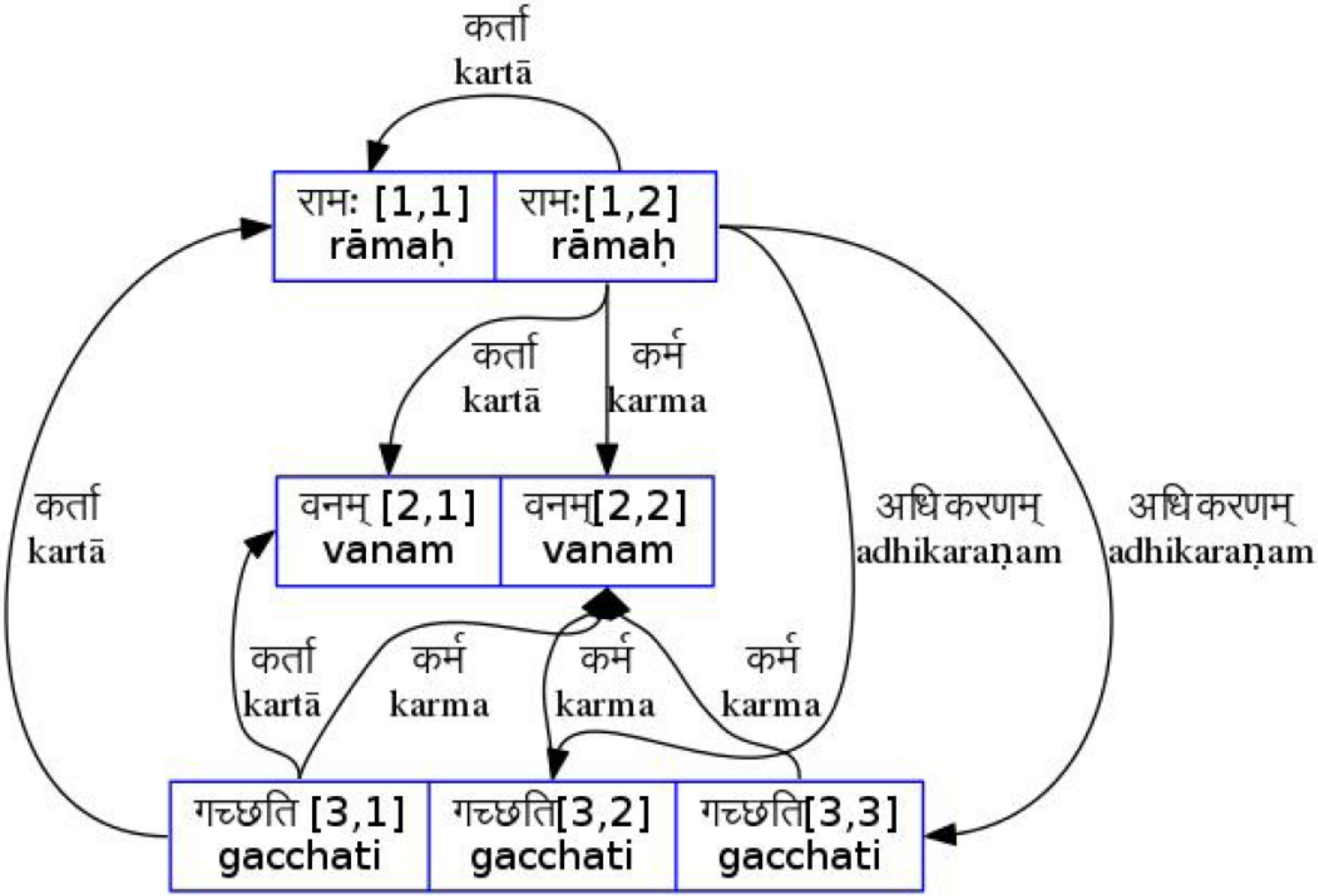
- Graphs assume only one kind of directed relationships between words
- Hence, morphological features of a word get fixed
- This requires a morphological analyzer
- POS tags also get fixed
- This requires a POS tagger
- Languages with a large number of noun and verb lemmas may generate a word from different roots
- Thus, should be made in context
- Just analyzing at word level requires more work later

Example

- rāmaḥ vanam gacchati (रामः वनम् गच्छति)
- rāmaḥ (रामः) has 2 different analyses
  - rāma (राम): noun: masculine, case 1, singular
  - rā (रा): verb: simple present, first person, plural, active voice, parasmaipadī
- gacchati (गच्छति) has 3 different analyses
  - gam (गम्): verb: simple present, third person, singular, active voice, parasmaipadī
  - gacchat (गच्छत्): (gam + śatṛ) noun: masculine, case 7, singular
  - gacchat (गच्छत्): (gam + śatṛ) noun: neuter, case 7, singular
- vanam (वनम्) has 2 different analyses
  - vana (वन): noun: neuter, case 1, singular
  - vana (वन): noun: neuter, case 2, singular
- Instead of 3 nodes, graph now has $2 + 3 + 2 = 7$ nodes

Graph

- Resulting graph has this structure



- Constraints need to be defined for parsing

Constraints for Full Graph

- Every relation is a 5-tuple: (i, j, R, k, l)
  - j^{th} analysis of word i has a directed edge of relationship R from l^{th} analysis of word k
- Initially, if there is an analysis possible, 5-dimensional *constraint matrix* $C[\cdot, \cdot, \cdot, \cdot, \cdot] = 1$; otherwise, it is 0
  - Entries are due to `ākāṅkṣā`, `yogyatā`, `sannidhi`, etc.
- Global constraints
- All the words should be connected
- There should be exactly n edges (assuming a *root* node)
- Final parse tree should be planar

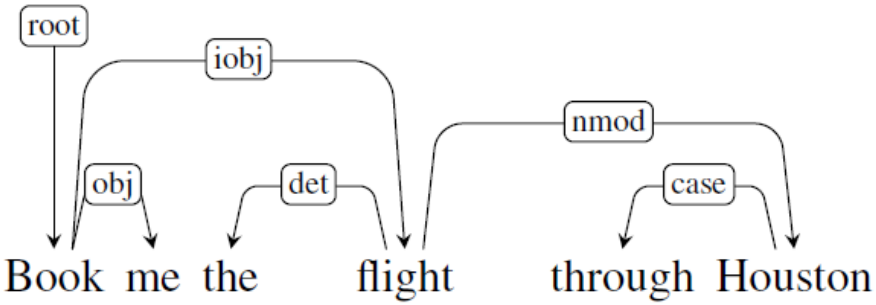
Local Constraints

- Local constraints
- For any word, there is one and exactly one analysis possible
- Moreover, there should be exactly 1 incoming edge
- No edge can be between analyses of the same word
- If there is an incoming edge to a particular analysis of word, the outgoing edge(s) should also be from the same analysis
- Except `adhikaraṇa`, there cannot be more than 1 `kāraka` relation from a verb
  - `dvikarmaka kriyā` has `mukhyakarma` and `gauṇakarma`
- Space quickly blows up
- More efficient graph algorithms if *word order* is imposed or assumed

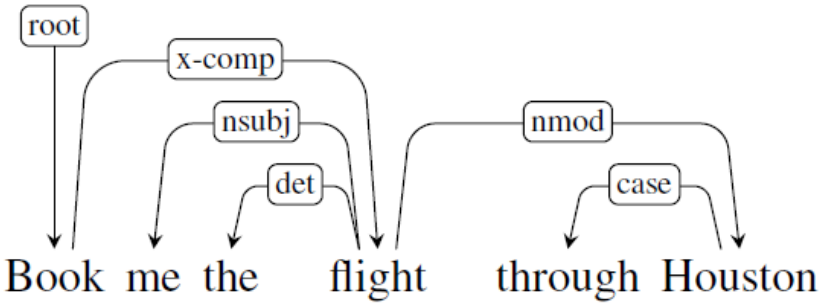
Evaluation

- How good is a dependency parse produced by an algorithm as compared to a *reference parse*?
- A yes/no decision is too harsh
- Attachment of a word to its head
- Labeled attachment score (LAS) measures how many words are correctly attached to its head with the *correct* dependency relationship
- Unlabeled attachment score (UAS) measures how many words are correctly attached to its head, ignoring the dependency relationship
- Label accuracy score (LS) measures how many words are attached with the correct dependency relationship, ignoring where it is attached to
- LAS is the strictest
- UAS is always greater than or equal to LAS

Example



(a) Reference



(b) System

- LAS $= 4/6$
- UAS $= 5/6$
- LS $= 4/6$