BSCS5002: Introduction to Natural Language Processing

Dependency Parsing

Parameswari Krishnamurthy

Language Technologies Research Centre
IIIT-Hyderabad

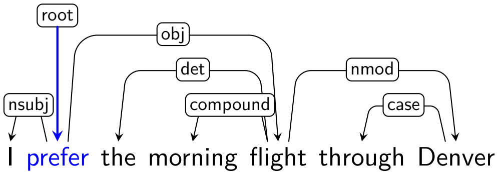
param.krishna@iiit.ac.in

Dependency Parsing

- Dependency parsing focuses on analyzing the grammatical structure of sentences by establishing binary grammatical relations between words, forming a typed dependency structure.

- **Dependency Structure:** Each word in a sentence is linked through directed, labeled arcs, representing relationships like subject or object.

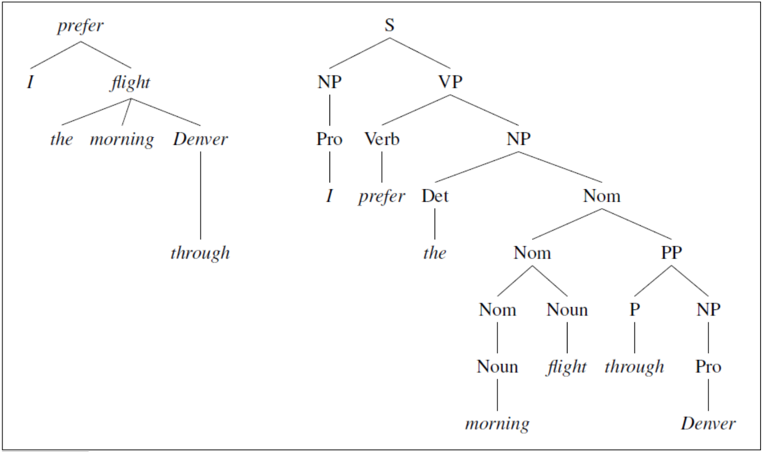
  For example, in *I prefer the morning flight through Denver*, the word *prefer* is linked as the root, and all other words depend on it.



Dependency Parsing

- **Difference from Constituent Parsing:** Dependency parsing focuses on the relations between individual words, not larger phrase structures.
- **Good for Free Word Order Languages:** It works well for languages with flexible word order (e.g., Tamil, Hindi) because it captures relationships between words without needing many grammar rules.
- **Typed Relations:** Relations like `nsubj` (subject), `obj` (object), and `nmod` (modifier) are used to describe the roles of words.
- **Example:** *I prefer the morning flight through Denver*, with relations like `nsubj`, `obj`, and `det`.

Dependency Relations



Dependency and constituent analyses for *I prefer the morning flight through Denver*.

Dependency Relations

- **Head-Dependent Relations:** In a dependency structure, each relation consists of a head (central organizing word) and a dependent (modifier). For example, *prefer* is the head, and *flight* is the dependent in *I prefer the flight*.
- **Grammatical Relations:** These relations include core functions like nominal subject (`nsubj`), direct object (`obj`), and clausal complement (`ccomp`). The relationship is not determined solely by word order but by grammatical roles.
- **Flexible Word Order:** In flexible languages (e.g., Tamil, Hindi), grammatical relations are crucial, as word order may not follow the typical subject-verb-object pattern.

Dependency Relations

- **Universal Dependencies (UD):** The UD project developed a cross-linguistic standard, creating a taxonomy of 37 relations applicable across 150+ languages. For instance, in *United canceled the morning flights to Houston*, we have relations like `nsubj`, `obj`, and `nmod`.
- **Core Relations:** The most common grammatical relations include subject, object, and modifiers like nominal modifiers (`nmod`) and adjectival modifiers (`amod`). Relations are categorized as either clausal (linked to verbs) or nominal (modifying nouns).

Dependency Relations

Clausal Argument Relations	Description
NSUBJ	Nominal subject
OBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Some of the Universal Dependency relations (de Marneffe et al., 2021).

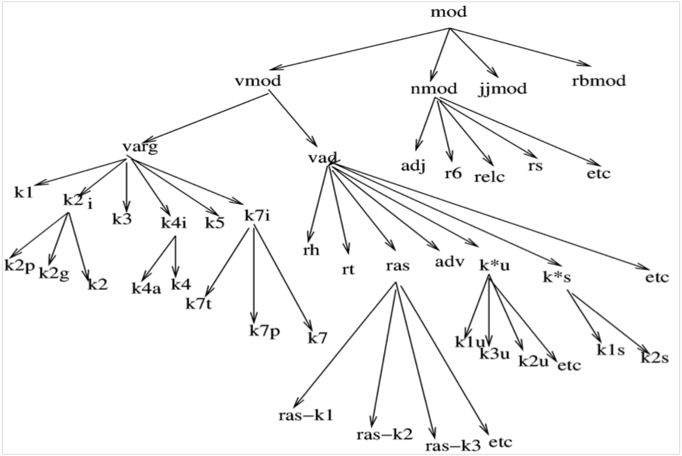
Paninian Dependency and Tags

- Paninian Dependency Model is a framework derived from ancient Paninian grammar (Sanskrit), adapted for modern linguistic analysis, especially for free-word order languages like Hindi and Sanskrit.
- Core Idea: The relationship between words is described using Kāraka relations (semantic roles), which define the syntactic and semantic relationship between a verb and its arguments.
- Key Kāraka Relations:
  - Karta (Subject/Agent): Doer of the action.
  - Karma (Object): The object or goal of the action.
  - Karana (Instrument): The means or instrument by which the action is performed.
  - Sampradana (Recipient): The recipient of the action.

Paninian Dependency and Tags

- Dependency Tags:
  - Similar to modern dependency tags like 'nsubj' (subject), 'obj' (object), the Paninian model uses Kāraka tags to represent semantic roles.
  - Examples:
    - Kartā $\rightarrow$ Subject (nsubj)
    - Karma $\rightarrow$ Object (obj) etc.

Paninian Dependency and Tags



Dependency Formalisms

- **Graph Representation:** Dependency structures are represented as directed graphs with vertices corresponding to words and arcs representing grammatical functions (head-dependent relations). This formalism helps visually illustrate syntactic dependencies.

- **Arcs and Vertices:** Each word in the sentence is a vertex, and the arcs denote relations.
  - For example, in *We booked her the flight*, the arc shows that *we* is the head of the relation with *booked*, while *her* is dependent on *booked*.

Dependency Formalisms

- **Connectedness and Root:** A dependency tree must be connected, with a single root node that governs the structure.
  - For example, *prefer* would be the root in the sentence *I prefer the flight*.
- **Acyclic Structures:** Dependency graphs must be acyclic, meaning they cannot have cycles where a word points back to itself. This ensures that there is a clear hierarchical relationship between words.
- **Planarity Constraint:** In addition to acyclic and connected requirements, the structure must often be planar, meaning the graph can be drawn without crossing arcs for simpler, more interpretable diagrams.

Dependency Relations

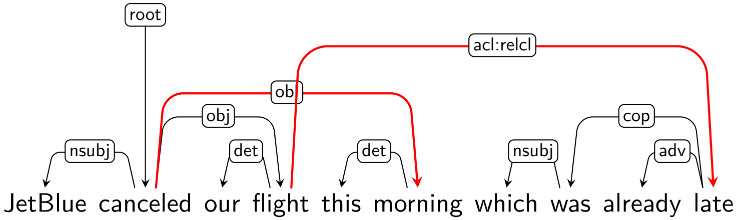
Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
OBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
COMPOUND	We took the **morning** *flight*.
NMOD	*flight* to **Houston**.
AMOD	Book the **cheapest** *flight*.
APPOS	*United*, a **unit** of UAL, matched the fares.
DET	**The** *flight* was canceled.
	Which *flight* was delayed?
CONJ	We *flew* to Denver and **drove** to Steamboat.
CC	We flew to Denver **and** *drove* to Steamboat.
CASE	Book the flight **through** *Houston*.

Examples of some Universal Dependency relations.

Projectivity

- **Definition of Projectivity:** An arc between a head and dependent is projective if there is no intervening word between them.
  - For example, the relation between *flight* and *morning* in *the morning flight* is projective because no word interrupts this relationship.

- **Projective vs. Non-Projective:** Non-projective constructions, common in free word-order languages, lead to crossing arcs in a dependency tree. In the example *"JetBlue canceled our flight this morning which was already late"*, *late* modifies *flight*, but the intervening words *this morning* create a non-projective arc.



Dependency Treebanks

- **Purpose of Treebanks:** Dependency treebanks serve as essential resources for training and evaluating parsers. They provide annotated examples of sentences with dependency structures, acting as gold standards for evaluating parser accuracy.

- **Creation of Treebanks:** These can either be created manually by linguists or through automatic parsers corrected by human annotators. For example, the Universal Dependencies project involves extensive human involvement in creating high-quality treebanks across multiple languages.

- **Translation of Phrase-Structure Trees:** Some early treebanks were derived by converting phrase-structure annotations into dependency trees. This deterministic process provides a bridge between different parsing formalisms.

Dependency Treebanks

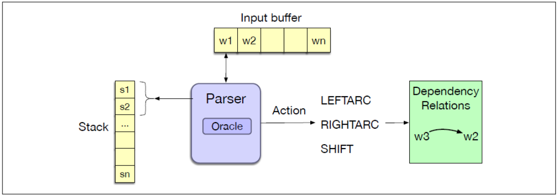
- **Universal Dependencies (UD) Project:** The UD project is the largest community-driven initiative for creating and maintaining dependency treebanks.
- It has nearly 200 treebanks in over 150 languages, each with standardized annotations for cross-linguistic comparison.

Transition-Based Dependency Parsing

- **Overview:** Transition-based parsing uses a shift-reduce approach, similar to techniques used in programming language parsing.
- **Stack and Buffer:** A stack is used to hold words being processed, while the buffer stores the remaining input words to be parsed.
- **Oracle:** An oracle determines which action (SHIFT, LEFTARC, or RIGHTARC) should be applied to the words on the stack to create a dependency tree.
- **Efficiency:** Transition-based parsers work in linear time, making a single pass through the sentence to construct the dependency tree.
- **Greedy Nature:** The parser makes greedy decisions without backtracking, which can sometimes result in incorrect parses.

Transition Operators

- **LEFTARC:** Asserts a head-dependent relation between the top two words on the stack, removing the second word from the stack.
- **RIGHTARC:** Establishes a head-dependent relation between the second word and the top word on the stack, removing the top word from the stack.
- **SHIFT:** Moves the next word from the buffer to the stack, progressing through the sentence.
- **Constraints:** Certain restrictions are applied, such as preventing ROOT from having incoming arcs.



Basic transition-based parser

Arc Standard Parsing Approach

- **Parsing Mechanism:** Arc standard parsing works by applying transitions between elements at the top of the stack.
- **Element Removal:** Once an element is assigned its head, it is removed from the stack and no longer available for processing.
- **Simple and Effective:** Despite being simple to implement, the arc standard approach is effective for many sentence structures.
- **Example:** For the sentence *Book me the morning flight*, LEFTARC links *flight* to *morning*, and RIGHTARC links *book* to *flight*.

```
function DEPENDENCYPARSE(words) returns dependency tree

  state ← {[root], [words], [] }  ; initial configuration
  while state not final
    t ← ORACLE(state)         ; choose a transition operator to apply
    state ← APPLY(t, state)   ; apply it, creating a new state
  return state
```

A generic transition-based dependency parser

Illustration

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book $\rightarrow$ me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning $\leftarrow$ flight)
7	[root, book, the, flight]	[]	LEFTARC	(the $\leftarrow$ flight)
8	[root, book, flight]	[]	RIGHTARC	(book $\rightarrow$ flight)
9	[root, book]	[]	RIGHTARC	(root $\rightarrow$ book)
10	[root]	[]	Done	

Illustration of the operation of the parser with the sequence of transitions

Graph-based Syntactic Parsing: Introduction

- Graph-based Syntactic Parsing is a method where parsing is framed as finding the best graph structure to represent the dependencies between words in a sentence.
- The sentence is treated as a graph, where:
  - Each word is a node
  - The dependencies (relationships) between words are edges
- The goal is to find the maximum spanning tree (MST) that captures the correct syntactic structure of the sentence.

Graph-based Parsing: Process

- Step 1: Treat the sentence as a fully connected graph.
- Step 2: Assign weights to the edges (relationships between words) based on a scoring function.
- Step 3: Use algorithms (e.g., Chu-Liu/Edmonds' algorithm) to find the Maximum Spanning Tree (MST), which represents the most likely syntactic structure.
- Advantages:
  - Works well for global optimization (best structure for the whole sentence).
  - Can handle languages with complex structures.

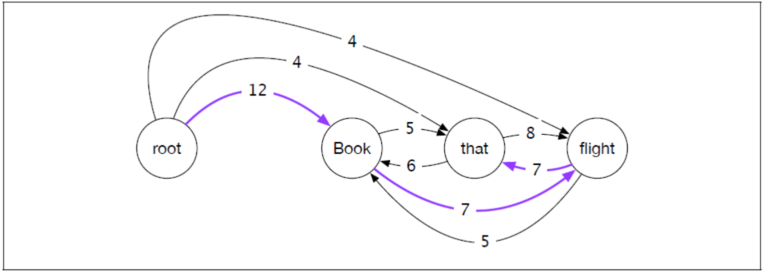
Graph-Based Dependency Parsing

- Graph-based parsing searches the space of all possible trees for a sentence and selects the tree with the maximum score based on dependency edges.
- **Edge-Factored Parsing:** The score of a dependency tree is calculated as the sum of individual edge scores. Each edge represents a head-dependent relation.
- **Global Decisions:** Unlike transition-based parsing, graph-based methods evaluate the entire tree at once, rather than making local, greedy decisions.
- **Handling Non-Projectivity:** Graph-based methods are capable of handling non-projective dependency trees, which are important for languages with free word order.
- **Applications:** Graph-based parsing is especially useful for long sentences and complex syntactic structures, where local decisions may lead to errors.

Parsing via Maximum Spanning Tree (MST)

- **Directed Graph Representation:** The sentence is represented as a directed graph, where vertices are words and edges represent possible head-dependent relationships.
- **Root Node:** A special ROOT node is included, with edges directed toward every other word. This ROOT node is necessary to form a valid parse tree.
- **Maximum Spanning Tree:** The goal of graph-based parsing is to find the maximum spanning tree (MST) of the directed graph, which represents the optimal dependency structure.
- **Edge Scoring:** Each edge is assigned a score by a scoring function, which considers various features (e.g., word forms, POS tags). The MST algorithm then finds the tree with the highest cumulative score.
- **Example:** For the sentence *Book that flight*, edges represent possible dependencies like *book* $\rightarrow$ *flight*. The MST selects the tree with the highest overall score .

Example



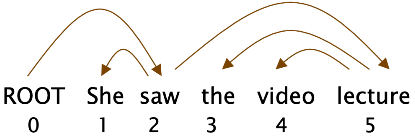
Initial rooted, directed graph for *Book that flight.*

Evaluation of Dependency Parsing

- **Exact Match (EM):** Measures sentences parsed without any errors.
- **Labeled Attachment Score (LAS):** Percent of words with correct head and dependency label.
- **Unlabeled Attachment Score (UAS):** Percent of words with correct head, ignoring labels.

Labeled and Unlabeled Attachment Score (LAS and UAS)

- **Labeled Attachment Score (LAS):** Accuracy of both head and dependency label.
- **Unlabeled Attachment Score (UAS):** Accuracy of head assignment only.
- **Label Accuracy Score (LS):** Focuses on the correctness of dependency labels.



Acc = $\frac{\text{\# correct deps}}{\text{\# of deps}}$

UAS = 4 / 5 = 80%

LAS = 2 / 5 = 40%

Gold			
1	2	She	nsubj
2	0	saw	root
3	5	the	det
4	5	video	nn
5	2	lecture	dobj

Parsed			
1	2	She	nsubj
2	0	saw	root
3	4	the	det
4	5	video	nsubj
5	2	lecture	ccomp

Precision and Recall in Parsing

- **Precision:** Percentage of correctly predicted dependency relations.
- **Recall:** Percentage of correct relations retrieved from total correct.
- **F1 Score:** Harmonic mean of precision and recall, indicating overall performance.

Summary

- **Dependency Grammar Overview:** Describes sentence structure with binary relations between words, focusing on grammatical relationships.
- **Transition-Based Parsing:** Constructs dependency trees with operators like SHIFT, LEFTARC, and RIGHTARC; efficient but can have local errors.
- **Graph-Based Parsing:** Evaluates entire trees, handling non-projective dependencies and long sentences effectively.
- **Dependency Treebanks:** Essential for training and evaluating parsers; Universal Dependencies offers a cross-linguistic standard.
- **Evaluation Metrics:** Metrics include Labeled Attachment Score (LAS), Unlabeled Attachment Score (UAS), precision, and recall.
- **Benefits of Dependency Grammar:** Especially effective for languages with free word order, focusing on word-level relations.
- **Future Research:** Advances in neural models and algorithms are driving progress in parsing multilingual texts.