CE4045 CZ4045 SC4002
Natural Language Processing

Dependency Parsing

Dr. Sun Aixin

Dependency Parsing

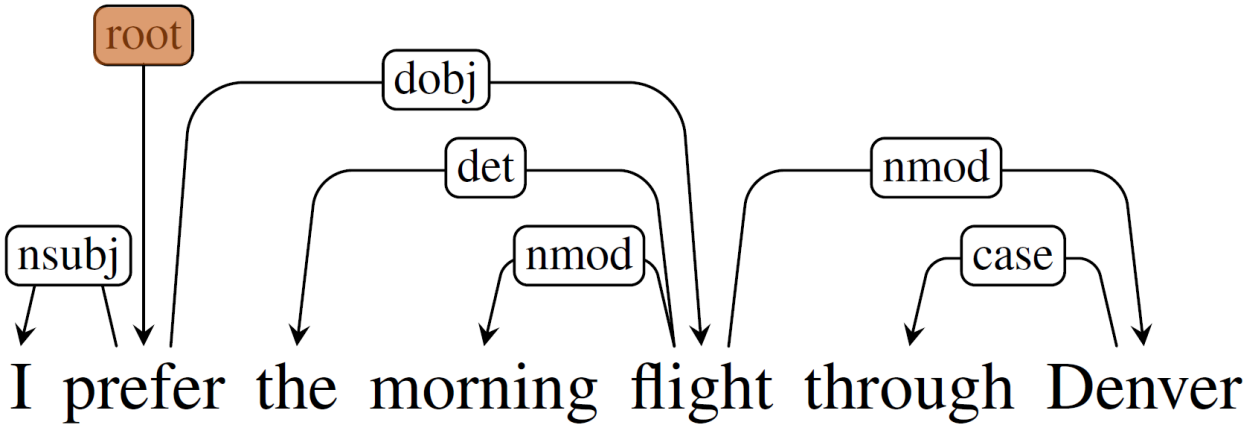
- Syntactic parsing is the task of assigning a syntactic structure to a sentence.
  - We need a **grammar**
  - We need a **parser**

- We have studied context-free grammars and constituent-based representations.

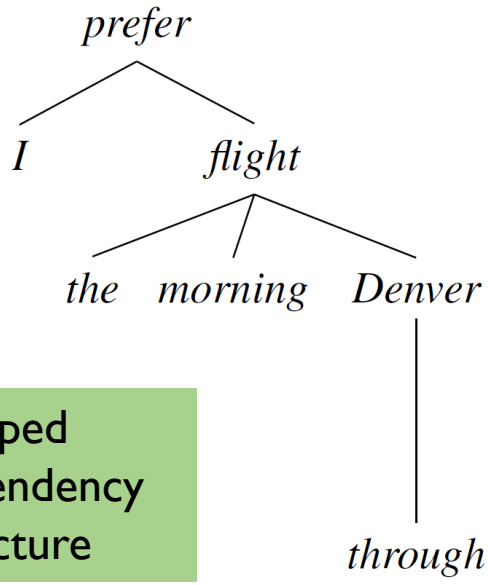
- We now study another important family of formalisms called dependency grammars.
  - In dependency formalisms, phrasal constituents and phrase-structure rules do not play a direct role. Instead,
  - The syntactic structure of a sentence is described solely in terms of **directed binary grammatical relations** between the words

An example dependency parsing

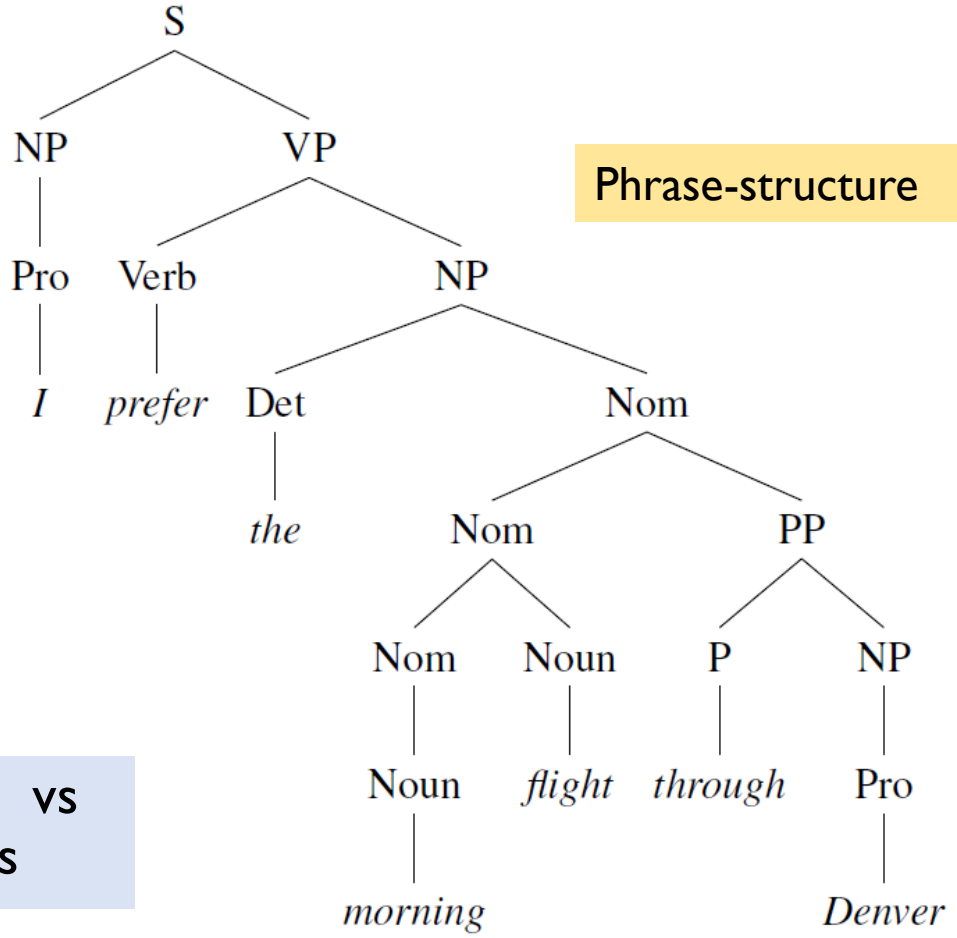
- Relations among the words are illustrated with **directed, labeled typed arcs** from **heads** to **dependents**
  - We call this a **typed dependency structure** because the labels are drawn from a predefined list of grammatical relations.
  - In **untyped dependency structure**, the relations are not labeled.
  - A **root** node explicitly marks the root of the tree, the **head of the entire structure**



Dependency parsing



untyped dependency structure

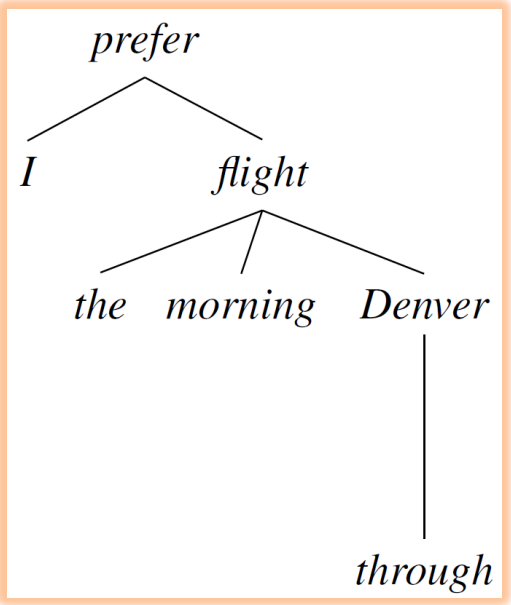


Phrase-structure

Dependency relation between words vs Constituency relation between words

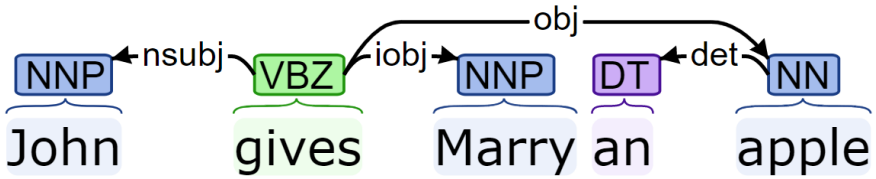
Dependency parsing

- The internal structure of the dependency parse consists solely of **directed relations between lexical items** in the sentence
- These **head-dependent** relationships directly encode important information that is often buried in the more complex phrase-structure parses.
  - E.g., **I** prefer the morning **flight** through Denver
  - A dependency grammar approach abstracts away from **word order information**, representing only the information that is necessary for the parse
  - The head-dependent relations provide **an approximation to the semantic relationship** between predicates and their arguments.
  - This is useful for many applications such as coreference resolution, question answering and information extraction.



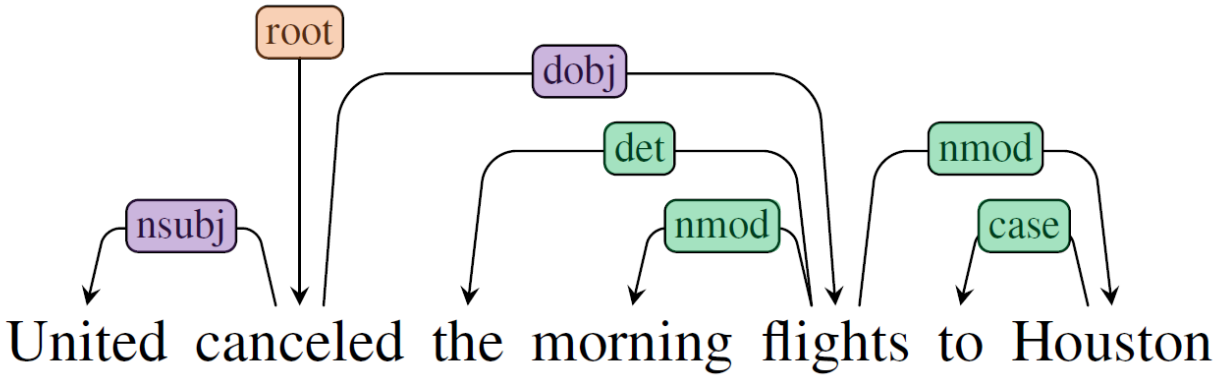
Dependency grammar

- The traditional linguistic notion of **grammatical relation** provides the basis for the binary relations in dependency structure
  - The arguments to these relations consist of a **head** and a **dependent**
  - The head is the word in a phrase that is grammatically the most important.
    - Typically, N is the head of an NP, V is the head of a VP
    - In a noun phrase various modifiers can occur before or after the head noun.
  - In dependency grammar, the **head-dependent** relationship is made **explicit** by directly linking heads to the words that are immediately dependent on them
  - The grammatical relation tells the grammatical function the dependent plays with respect to its head, e.g., *subject*, *direct object* and *indirect object*.



The Universal Dependencies (UD) project

- Linguists have developed taxonomies of relations that includes the familiar notions of subject and object.
- The UD project provides dependency relations that are linguistically motivated, computationally useful, and cross-linguistically applicable
  - The core set of **frequently used relations** can be broken into **two sets**
  - **Clausal relations** describe syntactic roles with respect to a predicate (e.g., a verb)
  - **Modifier relations** categorize the ways that words can modify their heads.



The Universal Dependencies (UD) project

Relation	Examples with *head* and **dependent**
NSUBJ	**United** *canceled* the flight.
DOBJ	United *diverted* the **flight** to Reno.
	We *booked* her the first **flight** to Miami.
IOBJ	We *booked* **her** the flight to Miami.
NMOD	We took the **morning** *flight*.
AMOD	Book the **cheapest** *flight*.
NUMMOD	Before the storm JetBlue canceled **1000** *flights*.
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*.

Definitions

https://universaldependencies.org/

https://universaldependencies.org/u/dep/index.html

The Universal Dependencies (UD) project

https://universaldependencies.org/guidelines.html

UD Guidelines

- Basic principles
  - Tokenization and word segmentation
  - Morphology
  - Syntax
  - Enhanced dependencies
  - CoNLL-U format and its extensions
  - Typos and other errors in underlying text
- Annotation guidelines
  - Nominals
  - Simple clauses
  - Complex clauses
  - Comparative constructions – working group materials
  - Other constructions
- Documentation of tags, features and relations
  - POS tags (single document)
  - Features (single document)
    - Layered features
    - Features in data (list of **all** features and values used in treebanks, including those that are r
  - Syntactic relations (single document)
    - Relations in data (list of **all** relation subtypes that are used in treebanks)
  - Conversion from other tagsets to UD tags and features
  - MISC attributes
- Incubator for Construction-Oriented Documentation (it will be moved here when it is mature enough)

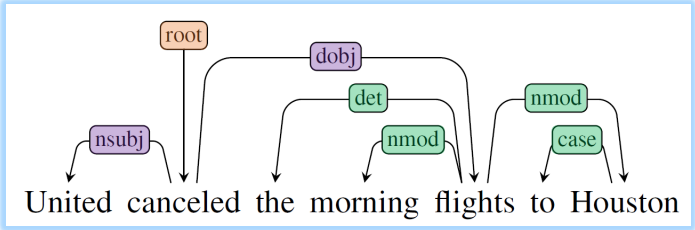
Search for keywords that we have covered in this course!

Dependency Formalisms

- A dependency structure is a directed graph $G = (V, A)$,
  - A set of vertices V. For simplicity, we consider V corresponds exactly to the set of words in a given sentence
  - A set of arcs A. Arcs are **ordered pairs** of vertices, which captures the head-dependent and grammatical function relationships between the elements in V.

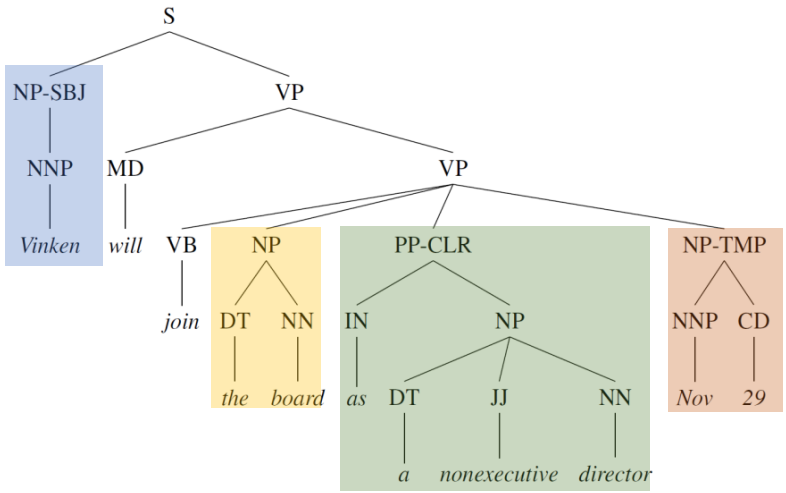
- A dependency tree is a directed graph that satisfies the following constraints:
  - There is a single designated **root** node that has no incoming arcs.
  - Other than the root node, **each vertex has exactly one incoming arc**.
  - There is **a unique path** from the root node to each vertex in V.

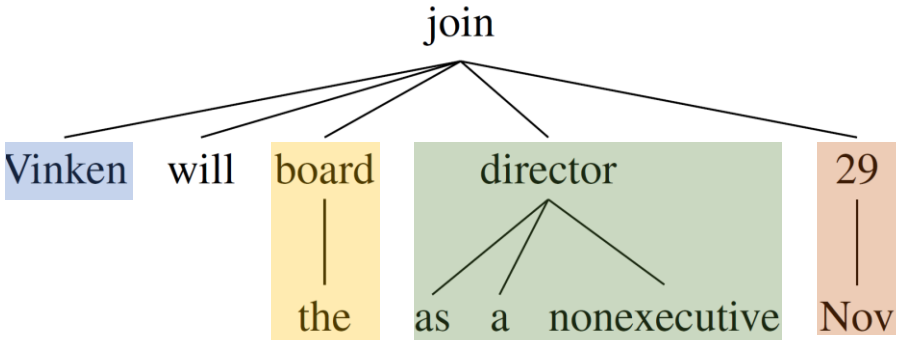
- A dependency tree, by dentition
  - Each word has a single head,
  - The dependency structure is connected,
  - There is a single root node from which one can follow a unique directed path to each of the words in the sentence.



Dependency treebanks

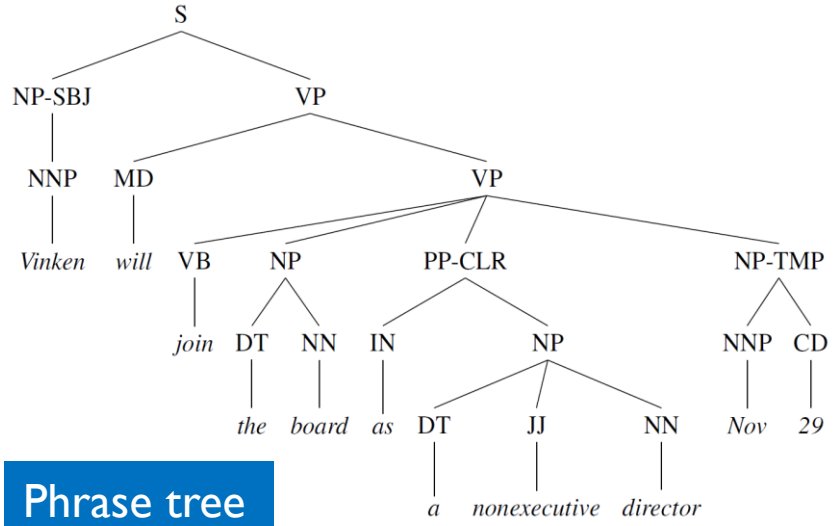
- Dependency treebanks have been created manually or with help of parsers
  - having human annotators label dependency structures for a given corpus,
  - using automatic parsers to provide an initial parse, then hand correct the parses
- Dependency treebanks can also be translated from constituent-based treebanks
  - Identifying all the head-dependent relations in the constituent-based structure
  - Identifying the correct dependency relations for these relations



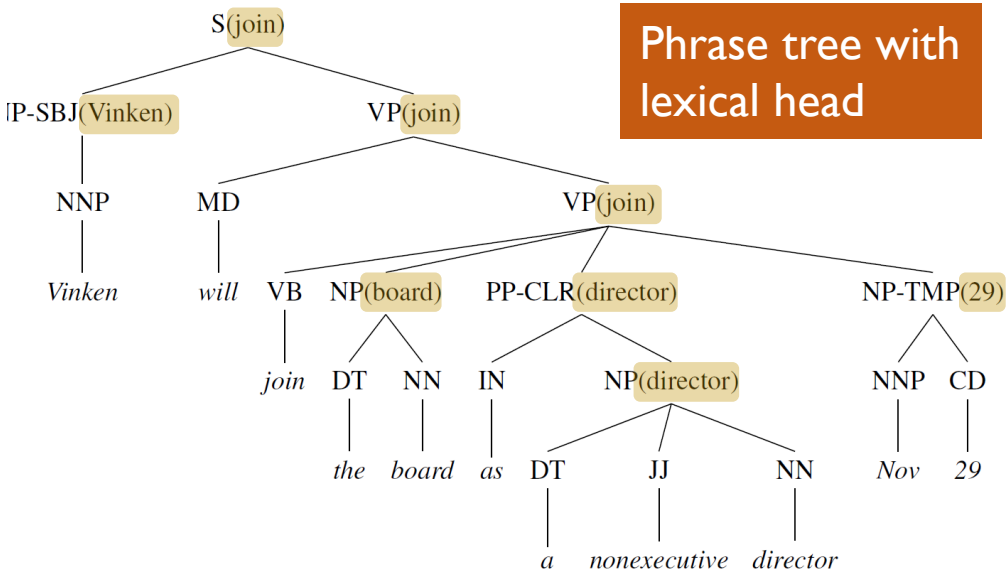


Dependency treebanks

- The head is the word in the phrase that is grammatically the most important.
  - *N* is the head of an *NP*; *V* is the head of a *VP*.
- In a phrase parse tree, heads are passed up; thus, each non-terminal in a parse tree is annotated with its lexical head



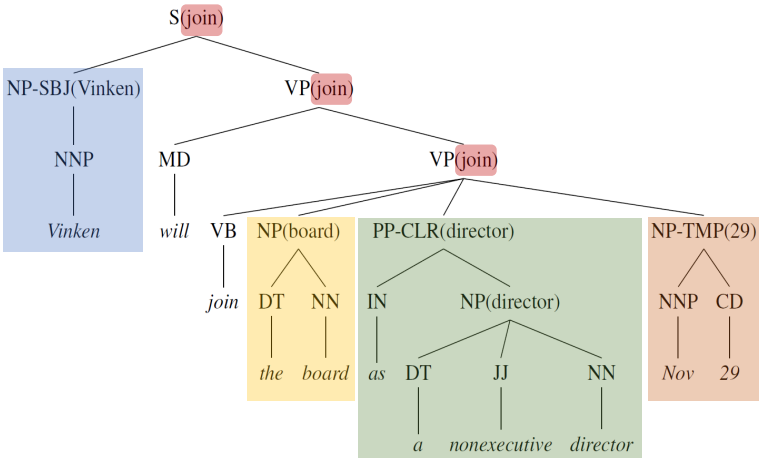
Phrase tree

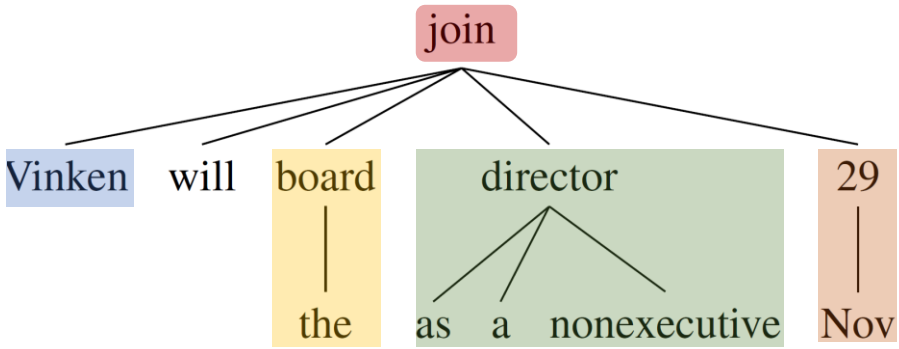


Phrase tree with lexical head

From constituent-based to dependency treebanks

- Translate constituent-based treebanks to dependency treebanks
  - Mark the head child of each node in a phrase structure, e.g., NP → DT NN, the head child is NN in this rule.
  - In the dependency structure, make the head of each non-head child depend on the head
  - Grammatical relations and function tags in the constituent-based parse trees can be used to label the edges in the resulting dependency tree

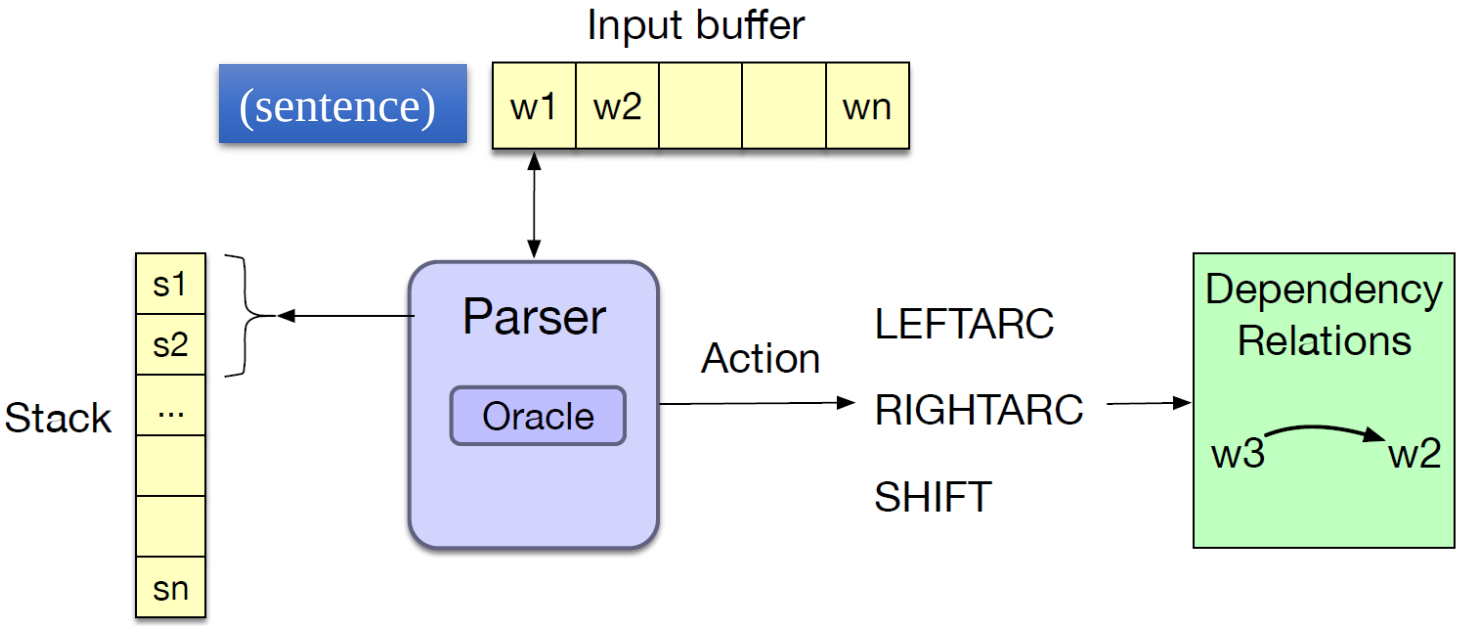




- This method works reasonably well for English. For other languages most dependency treebanks are developed directly using human annotators.

Transition-Based Dependency Parsing

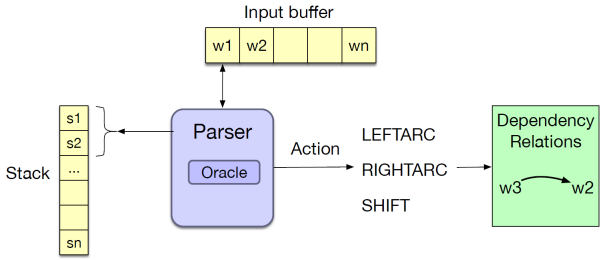
- This architecture is based on **shift-reduce parsing**, a paradigm originally developed for analyzing programming languages



- **Oracle** is a **classifier** to make one of the three decisions: LeftArc, RighArc, Shift.

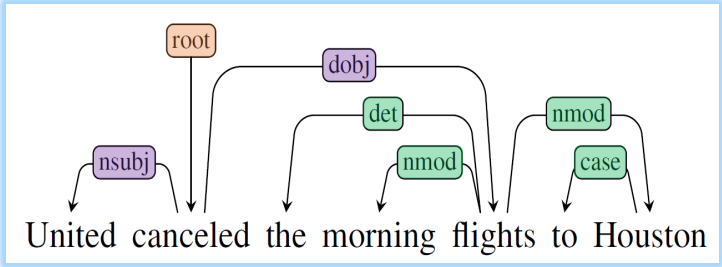
Shift-reduce parsing

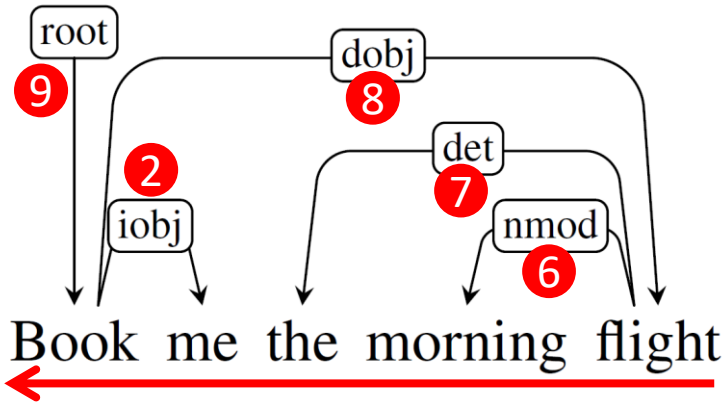
- The parser walks through the sentence left-to-right, successively shifting items from the buffer onto the stack

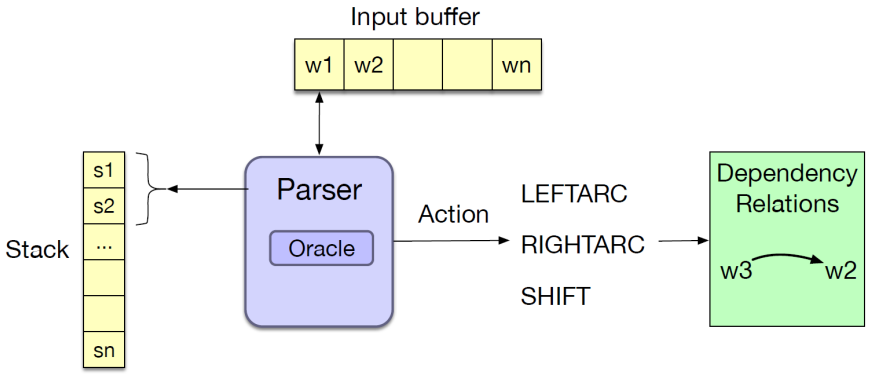


- At each time point we examine the top two elements on the stack, and decide:
  - LeftArc: Assert a head (top word)-dependent (second word) relation; remove the second word from the stack.
  - RightArc: Assert a head (second word)-dependent (top word) relation; remove the top word from the stack;
  - Shift: Remove the word from the front of the input buffer and push it onto the stack.

- LeftArc and RightArc are REDUCE actions
  - Once an element has been assigned its head, it is removed from the stack







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 → 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 ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

- **Initial state**: stack contains the ROOT node, the buffer contains all tokens in the sentence
- **Final goal state**: the stack and the word list **should be empty**

More on dependency parsing

- Multiple paths
  - There could be multiple shift-reduce sequences that might lead to a reasonable parse.
  - There may be other transition sequences that lead to different equally valid parses.
- Oracle (classifier) decisions
  - In this example, we assume that the oracle always provides the correct operator at each point in the parse, which is unrealistic.
  - Incorrect decisions will lead to incorrect parses because the parser has no opportunity to go back and pursue alternative choices.
  - In this example, we do not consider labels. To produce labeled trees, LeftArc and RightArc operators need to come with dependency labels, e.g., LeftArc(NSUBJ) or RightArc(DOBJ). This is equivalent to expanding the set of transition operators.
- There are enhancement for Transition-Based Dependency Parsing, and alternative parsing algorithms like Graph-Based Dependency Parsing and neural based models.

Summary

- Dependency: Head-dependent
- Dependency formalism
- Dependency parsing
- Reference
  - Chapter 19 https://web.stanford.edu/~jurafsky/slp3/