This is a simple sentence be 3sg present SIMPLE1 having few parts SENTENCE1 string of words satisfying the grammatical rules of a language

But it is an instructive one.
Syntax & Grammar

• Syntax
  • From Greek syntaxis, meaning “setting out together”
  • refers to the way words are arranged together.

• Grammar
  • Set of structural rules governing composition of clauses, phrases, and words in any given natural language
  • Descriptive, not prescriptive
  • Panini’s grammar of Sanskrit ~2000 years ago
Syntax and Grammar

• Goal of syntactic theory
  • “explain how people combine words to form sentences and how children attain knowledge of sentence structure”

• Grammar
  • implicit knowledge of a native speaker
  • acquired without explicit instruction
  • minimally able to generate all and only the possible sentences of the language

[Philips, 2003]
Syntax in NLP

- Syntactic analysis can be useful in many NLP applications
  - Grammar checkers
  - Dialogue systems
  - Question answering
  - Information extraction
  - Machine translation
  - ...

- Sequence models can go a long way but syntactic analysis is particularly useful
  - In low resource settings
  - In tasks where precise output structure matters
Two views of syntactic structure

• Constituency (phrase structure)
  • Phrase structure organizes words in nested constituents

• Dependency structure
  • Shows which words depend on (modify or are arguments of) which on other words
Constituency

• Basic idea: groups of words act as a single unit

• Constituents form coherent classes that behave similarly
  • With respect to their internal structure: e.g., at the core of a noun phrase is a noun
  • With respect to other constituents: e.g., noun phrases generally occur before verbs
Constituency: Example

• The following are all noun phrases in English...

<table>
<thead>
<tr>
<th>Harry the Horse</th>
<th>a high-class spot such as Mindy’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>the Broadway coppers</td>
<td>the reason he comes into the Hot Box</td>
</tr>
<tr>
<td>they</td>
<td>three parties from Brooklyn</td>
</tr>
</tbody>
</table>

• Why?
  • They can all precede verbs
  • They can all be preposed/postposed
  • ...
Grammars and Constituency

• For a particular language:
  • What are the “right” set of constituents?
  • What rules govern how they combine?

• Answer: not obvious and difficult
  • There are many different theories of grammar and competing analyses of the same data!
An Example Context-Free Grammar

<table>
<thead>
<tr>
<th>Grammar Rules</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP\ VP$</td>
<td>I + want a morning flight</td>
</tr>
<tr>
<td>$NP \rightarrow$</td>
<td></td>
</tr>
<tr>
<td>$Pronoun$</td>
<td>I</td>
</tr>
<tr>
<td>$Proper-Noun$</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>$Det\ Nominal$</td>
<td>a + flight</td>
</tr>
<tr>
<td>$Nominal \rightarrow$</td>
<td></td>
</tr>
<tr>
<td>$Nominal\ Noun$</td>
<td>morning + flight</td>
</tr>
<tr>
<td>$Noun$</td>
<td>flights</td>
</tr>
<tr>
<td>$VP \rightarrow$</td>
<td></td>
</tr>
<tr>
<td>$Verb$</td>
<td>do</td>
</tr>
<tr>
<td>$Verb\ NP$</td>
<td>want + a flight</td>
</tr>
<tr>
<td>$Verb\ NP\ PP$</td>
<td>leave + Boston + in the morning</td>
</tr>
<tr>
<td>$Verb\ PP$</td>
<td>leaving + on Thursday</td>
</tr>
<tr>
<td>$PP \rightarrow$</td>
<td></td>
</tr>
<tr>
<td>$Preposition\ NP$</td>
<td>from + Los Angeles</td>
</tr>
</tbody>
</table>
Parse Tree: Example

```
S
   NP  VP
     |    |  
   Pro Verb NP
     |    |    |  
   I prefer Det Nom
     |      |    |  
   a Nom Noun
     |    |    |  
   Noun flight
     |    |  
   morning
```
Dependency Grammars

• Context-Free Grammars focus on constituents
  • Non-terminals don’t actually appear in the sentence

• In dependency grammar, a parse is a graph (usually a tree) where:
  • Nodes represent words
  • Edges represent dependency relations between words
    (typed or untyped, directed or undirected)
Example Dependency Parse

They hid the letter on the shelf

Compare with constituent parse…

What's the relation?
Dependency Grammars

- Syntactic structure = lexical items linked by binary asymmetrical relations called dependencies
Example Dependency Parse

Dependencies form a tree:
- Connected
- Acyclic
- Single-head

India won the world cup by beating Lanka
# Dependency Relations

<table>
<thead>
<tr>
<th>Argument Dependencies</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsubj</td>
<td>nominal subject</td>
</tr>
<tr>
<td>csubj</td>
<td>clausal subject</td>
</tr>
<tr>
<td>dobj</td>
<td>direct object</td>
</tr>
<tr>
<td>iobj</td>
<td>indirect object</td>
</tr>
<tr>
<td>pobj</td>
<td>object of preposition</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modifier Dependencies</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tmod</td>
<td>temporal modifier</td>
</tr>
<tr>
<td>appos</td>
<td>appositional modifier</td>
</tr>
<tr>
<td>det</td>
<td>determiner</td>
</tr>
<tr>
<td>prep</td>
<td>prepositional modifier</td>
</tr>
<tr>
<td>Relation</td>
<td>Examples with <em>head</em> and <strong>dependent</strong></td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>NSUBJ</td>
<td>United <em>canceled</em> the flight.</td>
</tr>
<tr>
<td>DOBJ</td>
<td>United <em>diverted</em> the <strong>flight</strong> to Reno.</td>
</tr>
<tr>
<td></td>
<td>We <em>booked</em> her the first <strong>flight</strong> to Miami.</td>
</tr>
<tr>
<td>IOBJ</td>
<td>We <em>booked</em> her the flight to Miami.</td>
</tr>
<tr>
<td>NMOD</td>
<td>We took the <strong>morning flight</strong>.</td>
</tr>
<tr>
<td>AMOD</td>
<td>Book the <strong>cheapest flight</strong>.</td>
</tr>
<tr>
<td>NUMMOD</td>
<td>Before the storm JetBlue canceled <strong>1000 flights</strong>.</td>
</tr>
<tr>
<td>APPROS</td>
<td><em>United</em>, a <strong>unit</strong> of UAL, matched the fares.</td>
</tr>
<tr>
<td>DET</td>
<td><strong>The flight</strong> was canceled.</td>
</tr>
<tr>
<td></td>
<td><strong>Which flight</strong> was delayed?</td>
</tr>
<tr>
<td>CONJ</td>
<td>We <em>flew</em> to Denver and <strong>drove</strong> to Steamboat.</td>
</tr>
<tr>
<td>CC</td>
<td>We flew to Denver <strong>and drove</strong> to Steamboat.</td>
</tr>
<tr>
<td>CASE</td>
<td>Book the flight <strong>through</strong> Houston.</td>
</tr>
</tbody>
</table>

**Figure 14.3** Examples of core Universal Dependency relations.
Universal Dependencies project

• Set of dependency relations that are
  • Linguistically motivated
  • Computationally useful
  • Cross-linguistically applicable
  • [Nivre et al. 2016]

• Universaldependencies.org
Outline

• Syntax & Grammar

• Two views of syntactic structures
  • Context-Free Grammars
  • Dependency grammars
  • Can be used to capture various facts about the structure of language (but not all!)

• Dependency Parsing
Data-driven dependency parsing

**Goal:** learn a good predictor of dependency graphs

- Input: sentence
- Output: dependency graph/tree $G = (V,A)$

Can be framed as a structured prediction task
- very large output space
- with interdependent labels

2 dominant approaches: transition-based parsing and graph-based parsing
Transition-based dependency parsing

- Builds on shift-reduce parsing [Aho & Ullman, 1972]

- Configuration
  - Stack
  - Input buffer of words
  - Set of dependency relations

- Goal of parsing
  - find a final configuration where
  - all words accounted for
  - Relations form dependency tree
Defining Transitions

• **Transitions**
  • Are functions that produce a new configuration given current configuration
  • Parsing is the task of finding a sequence of transition that leads from start state to desired goal state

• **Start state**
  • Stack initialized with ROOT node
  • Input buffer initialized with words in sentence
  • Dependency relation set = empty

• **End state**
  • Stack and word lists are empty
  • Set of dependency relations = final parse
Arc Standard Transition System defines 3 transition operators [Covington, 2001; Nivre 2003]

**LEFT-ARC**
- create head-dependent relation between word at top of stack and 2\textsuperscript{nd} word (under top)
- remove 2\textsuperscript{nd} word from stack

**RIGHT-ARC**
- Create head-dependent relation between word on 2\textsuperscript{nd} word on stack and word on top
- Remove word at top of stack

**SHIFT**
- Remove word at head of input buffer
- Push it on the stack
Arc standard transition systems

- Preconditions
  - ROOT cannot have incoming arcs
  - LEFT-ARC cannot be applied when ROOT is the 2\textsuperscript{nd} element in stack
  - LEFT-ARC and RIGHT-ARC require 2 elements in stack to be applied
Transition-based Dependency Parser

```python
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
```

Figure 14.6 A generic transition-based dependency parser

Properties of this algorithm:
- Linear in sentence length
- A greedy algorithm
- Output quality depends on oracle
Let’s parse this sentence
### Transition-Based Parsing Illustrated

![Diagram of a transition-based parse](image)

<table>
<thead>
<tr>
<th>Step</th>
<th>Stack</th>
<th>Word List</th>
<th>Action</th>
<th>Relation Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[root]</td>
<td>[book, me, the, morning, flight]</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>[root, book]</td>
<td>[me, the, morning, flight]</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>[root, book, me]</td>
<td>[the, morning, flight]</td>
<td>RIGHTARC</td>
<td>(book \rightarrow me)</td>
</tr>
<tr>
<td>3</td>
<td>[root, book]</td>
<td>[the, morning, flight]</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>[root, book, the]</td>
<td>[morning, flight]</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>[root, book, the, morning]</td>
<td>[flight]</td>
<td>SHIFT</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>[root, book, the, morning, flight]</td>
<td>[]</td>
<td>LEFTARC</td>
<td>(morning \leftarrow flight)</td>
</tr>
<tr>
<td>7</td>
<td>[root, book, the, flight]</td>
<td>[]</td>
<td>LEFTARC</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>[root, book, flight]</td>
<td>[]</td>
<td>RIGHTARC</td>
<td>(book \rightarrow flight)</td>
</tr>
<tr>
<td>9</td>
<td>[root, book]</td>
<td>[]</td>
<td>RIGHTARC</td>
<td>(root \rightarrow book)</td>
</tr>
<tr>
<td>10</td>
<td>[root]</td>
<td>[]</td>
<td>Done</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 14.7** Trace of a transition-based parse.
Outline

• Syntax & Grammar

• Two views of syntactic structures
  • Context-Free Grammars
  • Dependency grammars
  • Can be used to capture various facts about the structure of language (but not all!)

• Dependency Parsing
  • Transition-based parser
Where do we get an oracle?

• Multiclass classification problem
  • Input: current parsing state (e.g., current and previous configurations)
  • Output: one transition among all possible transitions
  • Q: size of output space?

• Supervised classifiers can be used
  • E.g., perceptron

• Open questions
  • What are good features for this task?
  • Where do we get training examples?