Natural Language Processing

Parsing

Potsdam, 10 May 2012

Saeedeh Momtazi
Information Systems Group

based on the slides of the course book
Parsing

- Finding structural relationship between words in a sentence

Applications
- Spell checking
- Speech recognition
- Machine translation
- Language modeling
Outline

1. Phrase Structure

2. Syntactic Parsing
   CKY Algorithm

3. Statistical Parsing
Outline

1. Phrase Structure
2. Syntactic Parsing
   CKY Algorithm
3. Statistical Parsing
Constituency

- Working based on Constituency (Phrase structure)
  - Organizing words into nested constituents
  - Showing that groups of words within utterances can act as single units
  - Forming coherent classes from these units that can behave in similar ways
    - With respect to their internal structure
    - With respect to other units in the language
  - Considering a head word for each constituent
Constituency

the writer talked to the audiences about his new book.

the writer talked about his new book to the audiences. ✔

about his new book the writer talked to the audiences. ✔

the writer talked book to the audiences about his new. ✗
Grammar $G$ consists of

- Terminals ($T$)
- Non-terminals ($N$)
- Start symbol ($S$)
- Rules ($R$)
CFG

- **Terminals**
  - The set of words in the text

- **Non-Terminals**
  - The constituents in a language (noun phrase, verb phrase, ....)

- **Start symbol**
  - The main constituent of the language (sentence)

- **Rules**
  - Equations that consist of a single non-terminal on the left and any number of terminals and non-terminals on the right
CFG

\[ S \rightarrow NP \ VP \]
\[ S \rightarrow VP \]
\[ NP \rightarrow N \]
\[ NP \rightarrow Det \ N \]
\[ NP \rightarrow NP \ NP \]
\[ NP \rightarrow NP \ PP \]
\[ VP \rightarrow V \]
\[ VP \rightarrow VP \ PP \]
\[ VP \rightarrow VP \ NP \]
\[ PP \rightarrow Prep \ NP \]

\[ N \rightarrow \text{book} \]
\[ V \rightarrow \text{book} \]
\[ Det \rightarrow \text{the} \]
\[ N \rightarrow \text{flight} \]
\[ Prep \rightarrow \text{through} \]
\[ N \rightarrow \text{Houston} \]
Book the flight through Houston
S
  ├── VP
  │    └── V
  │         └── Book
  ├── NP
  │    └── DET
  │         └── the
  └── NP
      └── N
          └── flight
      ├── PP
      │    ├── PREP
      │    │    └── through
      │    └── N
      │         └── Houston
Outline

1. Phrase Structure

2. Syntactic Parsing
   CKY Algorithm

3. Statistical Parsing
Parsing

- Taking a string and a grammar and returning proper parse tree(s) for that string
- Covering all and only the elements of the input string
- Reaching the start symbol at the top of the string
- The system cannot select the correct tree among all the possible trees
Main Grammar Fragments

- Sentence
- Noun Phrase
  - Agreement
- Verb Phrase
  - Sub-categorization
Grammar Fragments: Sentence

- Declaratives
  A plane left.
  $S \rightarrow NP \ VP$

- Imperatives
  Leave!
  $S \rightarrow VP$

- Yes-No Questions
  Did the plane leave?
  $S \rightarrow Aux \ NP \ VP$

- WH Questions
  When did the plane leave?
  $S \rightarrow NP_{WH} \ Aux \ NP \ VP$
Grammar Fragments: NP

- Each NP has a central critical noun called **head**
- The head of an NP can be expressed using
  - Pre-nominals: the words that can come before the head
  - Post-nominals: the words that can come after the head
Grammar Fragments: NP

- Pre-nominals
  - Simple lexical items: the, this, a, an, ...
    - a car
  - Simple possessives
    - John’s car
  - Complex recursive possessives
    - John’s sister’s friend’s car
  - Quantifiers, cardinals, ordinals...
    - three cars
  - Adjectives
    - large cars
Grammar Fragments: NP

- Post-nominals
  - Prepositional phrases
    - flight from Seattle
  - Non-finite clauses
    - flight arriving before noon
  - Relative clauses
    - flight that serves breakfast
Agreement

- Having constraints that hold among various constituents
- Considering these constraints in a rule or set of rules

Example: determiners and the head nouns in NPs have to agree in number

This flight ✓
Those flights ✓
This flights ✗
Those flight ✗

- Grammars that do not consider constraints will over-generate
  - Accepting and assigning correct structures to grammatical examples (this flight)
  - But also accepting incorrect examples (these flight)
Agreement at sentence level

- Considering similar constraints at sentence level

Example: subject and verb in sentences have to agree in number and person

John flies ✓
We fly ✓
John fly ✗
We flies ✗
Agreement

- Possible CFG solution

\[
S_{sg} \rightarrow NP_{sg} \ VP_{sg}
\]
\[
S_{pl} \rightarrow NP_{pl} \ VP_{pl}
\]
\[
NP_{sg} \rightarrow Det_{sg} \ N_{sg}
\]
\[
NP_{pl} \rightarrow Det_{pl} \ N_{pl}
\]
\[
VP_{sg} \rightarrow V_{sg} \ NP_{sg}
\]
\[
VP_{pl} \rightarrow V_{pl} \ NP_{pl}
\]
...

- Shortcoming:
  - Introducing many rules in the system
Grammar Fragments: VP

- VPs consist of a head verb along with zero or more constituents called arguments.

\[
\begin{align*}
VP & \rightarrow V \\
VP & \rightarrow V \ NP \\
VP & \rightarrow V \ PP \\
VP & \rightarrow V \ NP \ PP \\
VP & \rightarrow V \ NP \ NP
\end{align*}
\]

- disappear
- prefer a morning flight
- fly on Thursday
- leave Boston in the morning
- give me the flight number

- Arguments
  - Obligatory: complement
  - Optional: adjunct
Sub-categorization

- Even though there are many valid VP rules, not all verbs are allowed to participate in all VP rules

\[\text{disappear a morning flight} \times\]

- Solution:
  - Subcategorizing the verbs according to the sets of VP rules that they can participate in
  - This is a modern take on the traditional notion of transitive/intransitive
  - Modern grammars may have 100s or such classes
Sub-categorization

- Example:

  Sneeze    John sneezed
  Find      Please find [a flight to NY]_{NP}
  Give      Give [me]_{NP}[a cheaper fair]_{NP}
  Help      Can you help [me]_{NP}[with a flight]_{PP}
  Prefer    I prefer [to leave earlier]_{TO-VP}
  Told      I was told [United has a flight]_{S}

  John sneezed the book ✗
  I prefer United has a flight ✗
  Give with a flight ✗
Sub-categorization

- The over-generation problem also exists in VP rules
  - Permitting the presence of strings containing verbs and arguments that do not go together

  *John sneezed the book*
  
  \[ VP \rightarrow V \ NP \]

- Solution:
  - Similar to agreement phenomena, we need a way to formally express the constraints
**Parsing Algorithms**

- **Top-Down**
  - Starting with the rules that give us an S, since trees should be rooted with an S
  - Working on the way down from S to the words

- **Bottom-Up**
  - Starting with trees that link up with the words, since trees should cover the input words
  - Working on the way up from words to larger and larger trees
Top-Down vs. Bottom-Up

- **Top-Down**
  - Only searches for trees that can be answers (i.e., S's)
  - But also suggests trees that are not consistent with any of the words

- **Bottom-Up**
  - Only forms trees consistent with the words
  - But suggests trees that make no sense globally
In both cases, we left out how to keep track of the search space and how to make choices.

**Solutions**

- **Backtracking**
  - Making a choice, if it works out then fine
  - If not, then back up and make a different choice
    \[ \Rightarrow \text{duplicated work} \]

- **Dynamic programming**
  - Avoiding repeated work
  - Solving exponential problems in polynomial time
  - Storing ambiguous structures efficiently
Dynamic Programming Methods

- CKY: bottom-up
- Early: top-down
Outline

1. Phrase Structure
2. Syntactic Parsing
   CKY Algorithm
3. Statistical Parsing
Chomsky Normal Form

- Each grammar can be represented by a set of binary rules

\[ A \rightarrow B \ C \]

\[ A \rightarrow w \]

\[ A, B, C \text{ are noun-terminals } w \text{ is a terminal} \]
Chomsky Normal Form

- Converting to Chomsky normal form

\[ A \rightarrow B \ C \ D \]
\[ X \rightarrow B \ C \]
\[ A \rightarrow X \ D \]

\( X \) does not occur anywhere else in the grammar
Chomsky Normal Form

- Converting to Chomsky normal form

\[ A \rightarrow B \]
\[ B \rightarrow C \, D \]
\[ A \rightarrow C \, D \]
CKY Parsing

\[ A \rightarrow B \ C \]

- If there is an \( A \) somewhere in the input, then there must be a \( B \) followed by a \( C \) in the input.

- If the \( A \) spans from \( i \) to \( j \) in the input, then there must be a \( k \) such that \( i < k < j \).
  - \( B \) spans from \( i \) to \( k \)
  - \( C \) spans from \( k \) to \( j \)
CKY Parsing

<table>
<thead>
<tr>
<th>[0,1]</th>
<th>[0,2]</th>
<th>[0,3]</th>
<th>[0,4]</th>
<th>[0,5]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[1,2]</td>
<td>[1,3]</td>
<td>[1,4]</td>
<td>[1,5]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[2,3]</td>
<td>[2,4]</td>
<td>[2,5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[3,4]</td>
<td>[3,5]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4,5]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Book the flight through Houston
CKY Parsing

N → book
V → book
[0,1]
[0,2]
[0,3]
[0,4]
[0,5]
Det → the
[1,2]
[1,3]
[1,4]
[1,5]
N → flight
[2,3]
[2,4]
[2,5]
Prep → through
[3,4]
[3,5]
N → houston
[4,5]

Book the flight through Houston
### CKY Parsing

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>book([0,1])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>book([0,1])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>N([0,1])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VP</td>
<td>V([0,1])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>VP([0,1])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Det</td>
<td>the([1,2])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>flight([2,3])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>N([2,3])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prep</td>
<td>through([3,4])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>houston([4,5])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Saeedeh Momtazi | NLP | 10.05.2012
### CKY Parsing

<table>
<thead>
<tr>
<th>N → book[^0,1]</th>
<th>V → book[^0,1]</th>
<th>NP → N[^0,1]</th>
<th>VP → V[^0,1]</th>
<th>S → VP[^0,1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,1]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Det → the[^1,2]</td>
<td>NP → Det[^1,2], N[^2,3]</td>
<td>[1,2]</td>
<td>[1,3]</td>
<td>[1,4]</td>
</tr>
<tr>
<td>N → flight[^2,3]</td>
<td>NP → N[^2,3]</td>
<td>[2,3]</td>
<td>[2,4]</td>
<td>[2,5]</td>
</tr>
<tr>
<td>Prep → through[^3,4]</td>
<td></td>
<td>[3,4]</td>
<td>[3,5]</td>
<td></td>
</tr>
<tr>
<td>N → houston[^4,5]</td>
<td>NP → N[^4,5]</td>
<td>[4,5]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Book</th>
<th>the</th>
<th>flight</th>
<th>through</th>
<th>Houston</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Saeedeh Momtazi | NLP | 10.05.2012
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>→</td>
<td>book&lt;sub&gt;[0,1]&lt;/sub&gt;</td>
<td>V</td>
<td>→</td>
</tr>
<tr>
<td>NP</td>
<td>→</td>
<td>N&lt;sub&gt;[0,1]&lt;/sub&gt;</td>
<td>VP</td>
<td>→</td>
</tr>
<tr>
<td>S</td>
<td>→</td>
<td>VP&lt;sub&gt;[0,1]&lt;/sub&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,1]</td>
<td>[0,2]</td>
<td>[0,3]</td>
<td>[0,4]</td>
</tr>
<tr>
<td>Det</td>
<td>→</td>
<td>the&lt;sub&gt;[1,2]&lt;/sub&gt;</td>
<td>NP</td>
</tr>
<tr>
<td>[1,2]</td>
<td>[1,3]</td>
<td>[1,4]</td>
<td>[1,5]</td>
</tr>
<tr>
<td>N</td>
<td>→</td>
<td>flight&lt;sub&gt;[2,3]&lt;/sub&gt;</td>
<td>NP</td>
</tr>
<tr>
<td>[2,3]</td>
<td>[2,4]</td>
<td>[2,5]</td>
<td></td>
</tr>
<tr>
<td>Prep</td>
<td>→</td>
<td>through&lt;sub&gt;[3,4]&lt;/sub&gt;</td>
<td>PP</td>
</tr>
<tr>
<td>[3,4]</td>
<td>[3,5]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>→</td>
<td>houston&lt;sub&gt;[4,5]&lt;/sub&gt;</td>
<td>NP</td>
</tr>
<tr>
<td>[4,5]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Book the flight through Houston
CKY Parsing

N → book[0,1]
V → book[0,1]
NP → N[0,1]
VP → V[0,1]
S → VP[0,1]
[0,1]

NP → NP[0,1], NP[1,3]
VP → VP[0,1], NP[1,3]
S → VP[0,3]

Det → the[1,2]
NP → Det[1,2], N[2,3]

N → flight[2,3]
NP → N[2,3]

[2,3]

Prep → through[3,4]
PP → Prep[3,4], NP[4,5]

N → houston[4,5]
NP → N[4,5]

[4,5]

Book the flight through Houston

0 1 2 3 4 5

Saeedeh Momtazi | NLP | 10.05.2012
### CKY Parsing

The CKY parsing algorithm is used to parse a sentence into its constituent parts. The algorithm uses three tables: `N`, `VP`, and `S`. Each table entry corresponds to a non-terminal symbol and an interval `[i, j]`, where `i` is the start index and `j` is the end index of the sentence.

- **Non-terminal symbols**:
  - `N` for noun phrase
  - `V` for verb phrase
  - `S` for sentence

- **Production rules**:
  - `N → book[0,1]`
  - `V → book[0,1]`
  - `NP → N[0,1]`
  - `VP → V[0,1]`
  - `S → VP[0,1]`
  - `Det → the[1,2]`
  - `NP → Det[1,2], N[2,3]`
  - `Prep → through[3,4]`
  - `PP → Prep[3,4], NP[4,5]`

The sentence parsed is: `Book the flight through Houston`.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Book</td>
<td>the</td>
<td>flight</td>
<td>through</td>
<td>Houston</td>
</tr>
</tbody>
</table>
CKY Parsing

N → book[0,1]
V → book[0,1]
NP → N[0,1]
VP → V[0,1]
S → VP[0,1]

[0,1] [0,2] [0,3] [0,4] [0,5]

Det → the[1,2]
NP → Det[1,2], N[2,3]

[1,2] [1,3] [1,4] [1,5]

N → flight[2,3]
NP → N[2,3]

[2,3] [2,4] [2,5]

Prep → through[3,4]
PP → Prep[3,4], NP[4,5]

[3,4] [3,5] [4,5]

N → houston[4,5]
NP → N[4,5]

Book the flight through Houston

0 1 2 3 4 5
Book the flight through Houston
Outline

1. Phrase Structure
2. Syntactic Parsing
   - CKY Algorithm
3. Statistical Parsing
Grammar $G$ consists of

- Terminals ($T$)
- Non-terminals ($N$)
- Start symbol ($S$)
- Rules ($R$)
- Probability function ($P$)

- $P : R \to [0, 1]$
- $\forall X \in N, \sum_{X \to \lambda \in R} P(X \to \lambda) = 1$
CFG

\[
\begin{align*}
S & \rightarrow NP \ VP \\
S & \rightarrow VP \\
NP & \rightarrow N \\
NP & \rightarrow Det \ N \\
NP & \rightarrow NP \ NP \\
NP & \rightarrow NP \ PP \\
VP & \rightarrow V \\
VP & \rightarrow VP \ PP \\
VP & \rightarrow VP \ NP \\
PP & \rightarrow Prep \ NP
\end{align*}
\]
<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \rightarrow NP\ VP$</td>
<td>0.9</td>
</tr>
<tr>
<td>$S \rightarrow VP$</td>
<td>0.1</td>
</tr>
<tr>
<td>$NP \rightarrow N$</td>
<td>0.3</td>
</tr>
<tr>
<td>$NP \rightarrow Det\ N$</td>
<td>0.4</td>
</tr>
<tr>
<td>$NP \rightarrow NP\ NP$</td>
<td>0.1</td>
</tr>
<tr>
<td>$NP \rightarrow NP\ PP$</td>
<td>0.2</td>
</tr>
<tr>
<td>$VP \rightarrow V$</td>
<td>0.1</td>
</tr>
<tr>
<td>$VP \rightarrow VP\ PP$</td>
<td>0.3</td>
</tr>
<tr>
<td>$VP \rightarrow VP\ NP$</td>
<td>0.6</td>
</tr>
<tr>
<td>$PP \rightarrow Prep\ NP$</td>
<td>1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N \rightarrow book$</td>
<td>0.5</td>
</tr>
<tr>
<td>$V \rightarrow book$</td>
<td>1.0</td>
</tr>
<tr>
<td>$Det \rightarrow the$</td>
<td>1.0</td>
</tr>
<tr>
<td>$N \rightarrow flight$</td>
<td>0.4</td>
</tr>
<tr>
<td>$Prep \rightarrow through$</td>
<td>1.0</td>
</tr>
<tr>
<td>$N \rightarrow Houston$</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Treebank

- A treebank is a corpus in which each sentence has been paired with a parse tree

- These are generally created by
  - Parsing the collection with an automatic parser
  - Correcting each parse by human annotators if required

- Requirement:
  - Detailed annotation guidelines that provide
    - A POS tagset
    - A grammar
    - Annotation schema
      - Instructions for how to deal with particular grammatical constructions
Penn Treebank

- Penn Treebank is a widely used treebank for English
  - Most well-known section: Wall Street Journal Section
    - 1 M words from 1987-1989

(S (NP (NNP John))
 (VP (VPZ flies)
   (PP (IN to)
    (NNP Paris)))))
 (. .))
Statistical Parsing

- Considering the corresponding probabilities while parsing a sentence
- Selecting the parse tree which has the highest probability

- Tree and string probabilities
  - $P(t)$: the probability of a tree $t$
    - Product of the probabilities of the rules used to generate the tree
  - $P(s)$: the probability of a string $s$
    - Sum of the probabilities of the trees which created to parse the string
PCFG

\[
\begin{align*}
S & \rightarrow NP \ VP & 0.9 \\
S & \rightarrow VP & 0.1 \\
NP & \rightarrow N & 0.3 & N \rightarrow book & 0.5 \\
NP & \rightarrow Det \ N & 0.4 & V \rightarrow book & 1.0 \\
NP & \rightarrow NP \ NP & 0.1 & Det \rightarrow the & 1.0 \\
NP & \rightarrow NP \ PP & 0.2 & N \rightarrow flight & 0.4 \\
VP & \rightarrow V & 0.1 & Prep \rightarrow through & 1.0 \\
VP & \rightarrow VP \ PP & 0.3 & N \rightarrow Houston & 0.1 \\
VP & \rightarrow VP \ NP & 0.6 \\
PP & \rightarrow Prep \ NP & 1.0
\end{align*}
\]
Statistical Parsing

\[ P(t) = 0.2 \times 0.4 \times 1.0 \times 1.0 \times 0.4 \times 1.0 \times 0.1 = 0.0032 \]
Probabilistic CKY Parsing

- \( N \rightarrow \text{book}[0,1] \)
- \( V \rightarrow \text{book}[0,1] \)
- \( NP \rightarrow N[0,1] \)
- \( VP \rightarrow V[0,1] \)
- \( S \rightarrow \text{VP}[0,1] \)

- \( 0.1 \times 0.15 \times 0.16 = 0.0024 \)
- \( 0.6 \times 0.1 \times 0.16 = 0.0096 \)
- \( 0.1 \times 0.0096 = 0.00096 \)

- \( NP \rightarrow NP[0,1] \), \( NP[1,3] \)
- \( VP \rightarrow VP[0,1] \), \( VP[1,3] \)
- \( S \rightarrow VP[0,3] \)

- \( Det \rightarrow \text{the}[1,2] \)
- \( NP \rightarrow \text{Det}[1,2], N[2,3] \)
- \( 0.4 \times 1.0 \times 0.4 = 0.16 \)

- \( N \rightarrow \text{flight}[2,3] \)
- \( NP \rightarrow N[2,3] \)
- \( 0.4 \times 0.3 \times 0.4 = 0.12 \)

- \( Prep \rightarrow \text{through}[3,4] \)
- \( PP \rightarrow Prep[3,4], NP[4,5] \)

- \( N \rightarrow \text{houston}[4,5] \)
- \( NP \rightarrow N[4,5] \)

Book  the  flight  through  Houston

0  1  2  3  4  5
Exercise

- Implement the probabilistic CKY algorithm which works based on the grammar rules $R$. 
Further Reading

- Speech and Language Processing
  - Chapters 12, 13, 14, 15