Chart Parsing

Allen’s Chapter 3
J & M’s Chapter 10
Chart Parsing

General Principles:

A *Bottom-Up* parsing method

– Construct a parse starting from the input symbols
– Build constituents from sub-constituents
– When all constituents on the RHS of a rule are matched, create a constituent for the LHS of the rule

The *Chart* allows storing partial analyses, so that they can be shared.

Data structures used by the algorithm:

– **The Key**: the current constituent we are attempting to “match”
– **An Active Arc**: a grammar rule that has a partially matched RHS
– **The Agenda**: Keeps track of newly found unprocessed constituents
– **The Chart**: Records processed constituents (non-terminals) that span substrings of the input
The Chart Parsing Algorithm

Extending Active Arcs with a Key:
- Each **Active Arc** has the form:
  \[ <p_i> [A \rightarrow X_1 \ldots \bullet C \ldots X_m] <p_j> \]
- A Key constituent has the form: \[ <p_j> \bullet <p_k> \]
- When processing the Key \[ <p_1> \bullet <p_2> \], we search the active arc list for an arc \[ <p_0> [A \rightarrow X_1 \ldots \bullet C \ldots X_m] <p_1> \], and then create a new active arc \[ <p_0> [A \rightarrow X_1 \ldots \bullet \ldots X_m] <p_2> \]
- If the new active arc is a completed rule:
  \[ <p_0> [A \rightarrow X_1 \ldots \bullet] <p_2> \], then we add \[ <p_0> A <p_2> \] to the Agenda
- After “using” the key to extend all relevant arcs, it is entered into the Chart
The Chart Parsing Algorithm

The Main Algorithm: parsing input $x = x_1 \ldots x_n$

1. $i = 0$
2. If Agenda is empty and $i < n$ then set $i = i + 1$, find all POS of $x_i$ and add them as constituents $<p_i>C<p_{i+1}>$ to the Agenda
3. Pick a Key constituent $<p_j>C<p_k>$ from the Agenda
4. For each grammar rule of form $A \rightarrow CX_1 \ldots X_m$, add $<p_j>[A \rightarrow • CX_1 \ldots X_m]<p_j>$ to the list of active arcs
5. Use Key to extend all relevant active arcs
6. Add LHS of any completed active arcs into the Agenda
7. Insert the Key into the Chart
8. If Key is $<1>S<n>$ then Accept the input Else goto (2)
The Grammar:

1. $S \rightarrow NP \ VP$
2. $NP \rightarrow ART \ ADJ \ N$
3. $NP \rightarrow ART \ N$
4. $NP \rightarrow ADJ \ N$
5. $VP \rightarrow AUX \ VP$
6. $VP \rightarrow V \ NP$
Chart Parsing - Example

The input:

“x = The large can can hold the water”

POS of Input Words:

- the: ART
- large: ADJ
- can: N, AUX, V
- hold: N, V
- water: N, V
The large can hold the water

Figure 3.9 The chart after seeing an ADJ in position 2
The large can can hold the water
The large can can hold the water
The large can can hold the water

<table>
<thead>
<tr>
<th>ART1</th>
<th>ADJ1</th>
<th>AUX1</th>
<th>AUX2</th>
<th>N3</th>
<th>ART2</th>
<th>N4</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>large</td>
<td>can</td>
<td>can</td>
<td>hold</td>
<td>the</td>
<td>water</td>
</tr>
</tbody>
</table>

**Figure 3.14** The chart after all the NPs are found, omitting all but the crucial active arcs
The final Chart

<table>
<thead>
<tr>
<th>S1 (rule 1 with NP1 and VP2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2 (rule 1 with NP2 and VP2)</td>
</tr>
<tr>
<td>VP3 (rule 5 with AUX1 and VP2)</td>
</tr>
<tr>
<td>NP2 (rule 4)</td>
</tr>
<tr>
<td>NP1 (rule 2)</td>
</tr>
<tr>
<td>N1</td>
</tr>
<tr>
<td>V1</td>
</tr>
<tr>
<td>ART1</td>
</tr>
<tr>
<td>1 the</td>
</tr>
</tbody>
</table>

Figure 3.15 The final chart
Improved Chart Parser

Increasing the Efficiency of the Algorithm

- Observation: The algorithm creates constituents that cannot “fit” into a global parse structure. Example: 
  \[2\] [NP \rightarrow ADJ N] [3]
- These can be eliminated using *predictions*
- Using Top-down predictions:
  - For each non-terminal A, define First(A) = the set of all leftmost non-terminals derivable from A
  - Compute in advance First(A) for all non-terminals A
  - When processing the key \[p_j\]C\[p_k\] we look for grammar rules that form an active arc \[p_j\] [A \rightarrow \bullet C X_2 \ldots X_m] \[p_j\]
  - Add the active arc *only* if A is *predicted*: there already exists an active arc \[p_k\] [B \rightarrow X_1 \ldots \bullet D \ldots X_m] \[p_j\] such that A \in First(D)
  - For \(p_k = 1\) (arcs of the form \[1\] [A \rightarrow \bullet C X_2 \ldots X_m] \[1\]), add the active arc *only* if A \in First(S)
Improved Chart Parser

Top-Down Arc Introduction Algorithm
To add an arc $S \rightarrow C_1 \ldots \circ C_i \ldots C_n$ ending at position $j$, do the following:

For each rule in the grammar of form $C_i \rightarrow X_1 \ldots X_k$, recursively add the new arc $C_i \rightarrow \circ X_1 \ldots X_k$ from position $j$ to $j$.

Top-Down Chart Parsing Algorithm
Initialization: For every rule in the grammar of form $S \rightarrow X_1 \ldots X_k$, add an arc labeled $S \rightarrow \circ X_1 \ldots X_k$ using the arc introduction algorithm.

Parsing: Do until there is no input left:
1. If the agenda is empty, look up the interpretations of the next word and add them to the agenda.
2. Select a constituent from the agenda (call it constituent $C$).
3. Using the arc extension algorithm, combine $C$ with every active arc on the chart. Any new constituents are added to the agenda.
4. For any active arcs created in step 3, add them to the chart using the top-down arc introduction algorithm.
Improved Chart Parser - Example

**NP1 (rule 2)**

<table>
<thead>
<tr>
<th>ART1</th>
<th>ADJ1</th>
<th>N1</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>large</td>
<td>can</td>
</tr>
</tbody>
</table>

NP → ART°N

NP → ART°ADJ N

NP → ART ADJ°N

S → NP°VP

S → °NP VP

NP → °ADJ N

NP → °ART N

NP → °ART ADJ N

VP → °AUX VP

VP → °V·NP

Figure 3.24 The chart after building the first NP
The large can can hold the water

Figure 3.25 The chart after adding *hold*, omitting arcs generated for the first NP
The large can can hold the water

Figure 3.26 The final chart for the top-down filtering algorithm
Chart Parsing: Pros and Cons

Pros

- Relatively efficient because each constituent is generated exactly once.
- Easy to generate a single parse, N parses, or all possible parses.
- If no complete parse is found, then easy to gather pieces and construct a partial parse.
- Could search for only certain types of constituents, such as NPs and VPs.

Cons

- Parsing is still not cheap, especially if the grammar is large.
- The chart can become quite large in some cases.