INF4820: Algorithms for Artificial Intelligence and Natural Language Processing

Treebank Parsing

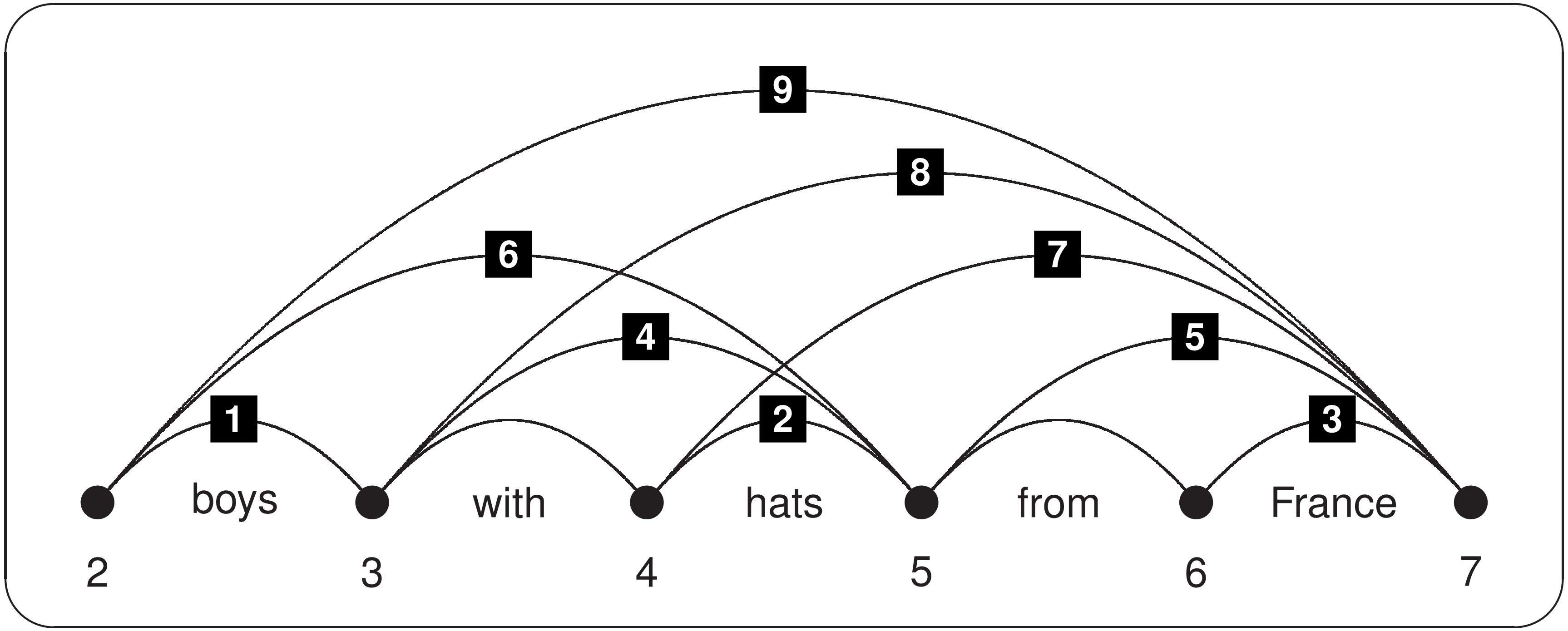
Stephan Oepen & Milen Kouylekov

Language Technology Group (LTG)

November 12, 2014

A Key Insight: Local Ambiguity

- For many substrings, more than one way of deriving the same category;
- NPs: 1 | 2 | 3 | 6 | 7 | 9; PPs: 4 | 5 | 8; 9 ≡ 1 + 8 | 6 + 5;
- *parse forest* — a single item represents multiple trees [Billot & Lang, 89].



The CKY (Cocke, Kasami, & Younger) Algorithm

for $(0 \leq i < |\mathit{input}|)$ do
  $\mathit{chart}_{[i,i+1]} \leftarrow \{\alpha \,|\, \alpha \rightarrow \mathit{input}_i \in P\}$;
for $(1 \leq l < |\mathit{input}|)$ do
  for $(0 \leq i < |\mathit{input}| - l)$ do
    for $(1 \leq j \leq l)$ do
      if $(\alpha \rightarrow \beta_1\, \beta_2 \in P \wedge \beta_1 \in \mathit{chart}_{[i,i+j]} \wedge \beta_2 \in \mathit{chart}_{[i+j,i+l+1]})$ then
        $\mathit{chart}_{[i,i+l+1]} \leftarrow \mathit{chart}_{[i,i+l+1]} \cup \{\alpha\}$;

Kim adored snow in Oslo

	1	2	3	4	5
0	NP		S		S
1		V	VP		VP
2			NP		NP
3				P	PP
4					NP

Chart Parsing: Key Ideas

- The parse *chart* is a two-dimensional matrix of *edges* (aka chart items);
- an edge is a (possibly partial) rule instantiation over a substring of input;
- the chart indexes edges by start and end string position (aka vertices);
- dot in rule RHS indicates degree of completion: $\alpha \rightarrow \beta_1 \ldots \beta_{i-1} \bullet \beta_i \ldots \beta_n$;
- *active* edges (aka *incomplete* items) — partial RHS: $[1, 2, \text{VP} \rightarrow \text{V} \bullet \text{NP}]$;
- *passive* edges (aka *complete* items) — full RHS: $[1, 3, \text{VP} \rightarrow \text{V NP}\bullet]$;

The Fundamental Rule

$$[i,\ j,\ \alpha \rightarrow \beta_1 ... \beta_{i-1} \bullet \beta_i ... \beta_n] + [j,\ k,\ \beta_i \rightarrow \gamma^+ \bullet]$$
$$\mapsto [i,\ k,\ \alpha \rightarrow \beta_1 ... \beta_i \bullet \beta_{i+1} ... \beta_n]$$

An Example of a (Near-)Complete Chart

	1	2	3	4	5
0	NP → NP • PP S → NP • VP NP → kim •				S → NP VP •
1		VP → V • NP V → adores •	VP → VP • PP VP → V NP •		VP → VP • PP VP → VP PP • VP → V NP •
2			NP → NP • PP NP → snow •		NP → NP • PP NP → NP PP •
3				PP → P • NP P → in •	PP → P NP •
4					NP → NP • PP NP → oslo •

$_0$ *Kim* $_1$ *adores* $_2$ *snow* $_3$ *in* $_4$ *Oslo* $_5$

Combinatorics: Keeping Track of Remaining Work

The Abstract Goal

- Any chart parsing algorithm needs to check all pairs of adjacent edges.

A Naïve Strategy

- Keep iterating through the complete chart, combining all possible pairs, until no additional edges can be derived (i.e. the fixpoint is reached);
- frequent attempts to combine pairs multiple times: deriving 'duplicates'.

An Agenda-Driven Strategy

- Combine each pair exactly once, viz. when both elements are available;
- maintain *agenda* of new edges, yet to be checked against chart edges;
- new edges go into agenda first, add to chart upon retrieval from agenda.

Backpointers: Recording the Derivation History

	0	1	2	3
0	2: S → • NP VP 1: NP → • NP PP 0: NP → • kim	10: S → 8 • VP 9: NP → 8 • PP 8: NP → kim •		17: S → 8 15 •
1		5: VP → • VP PP 4: VP → • V NP 3: V → • adores	12: VP → 11 • NP 11: V → adores •	16: VP → 15 • PP 15: VP → 11 13 •
2			7: NP → • NP PP 6: NP → • snow	14: NP → 13 • PP 13: NP → snow •
3				

- Use edges to record derivation trees: backpointers to daughters;
- a single edge can represent multiple derivations: backpointer sets.

Ambiguity Packing in the Chart

General Idea

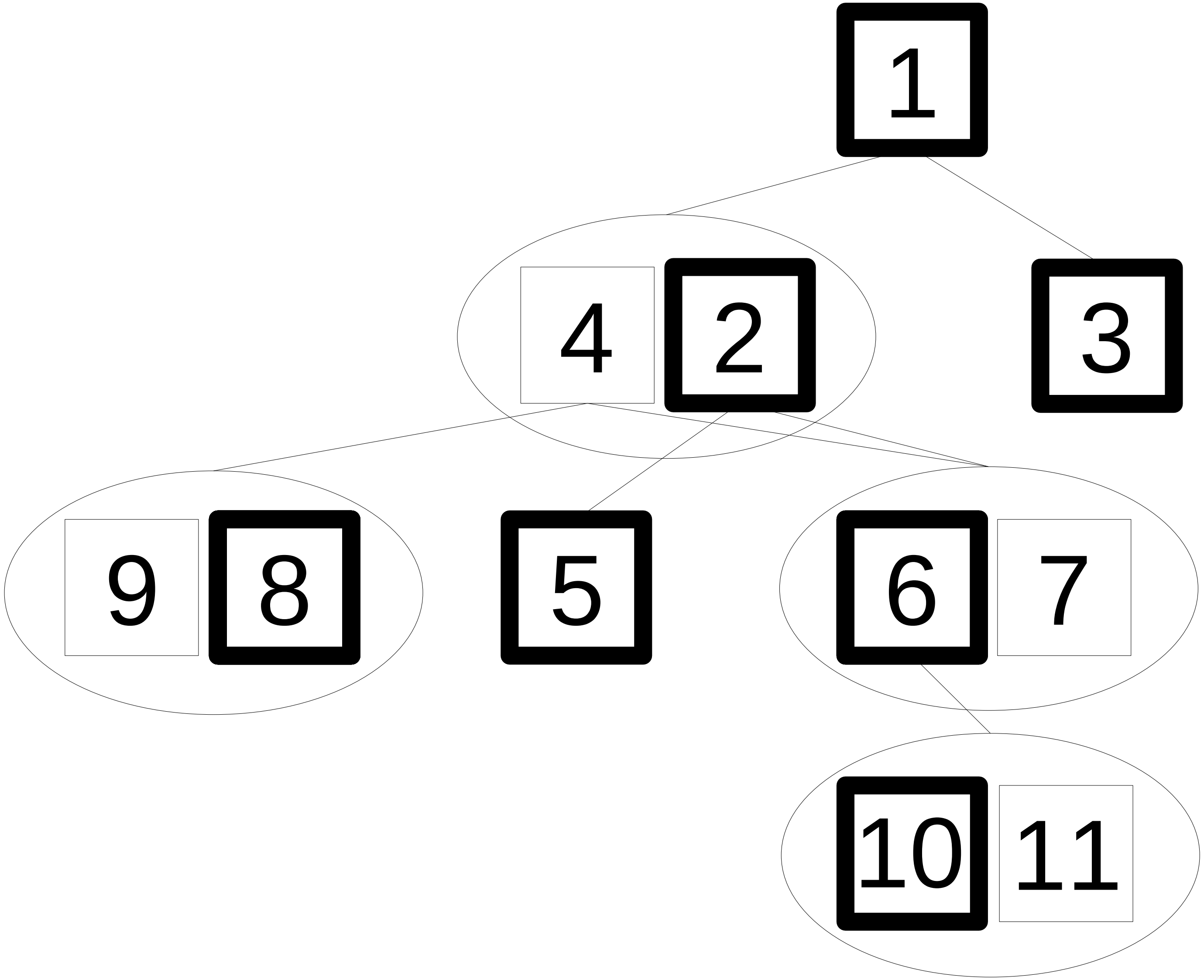
- Maintain only one edge for each α from i to j (the 'representative');
- record alternate sequences of daughters for α in the representative.

Implementation

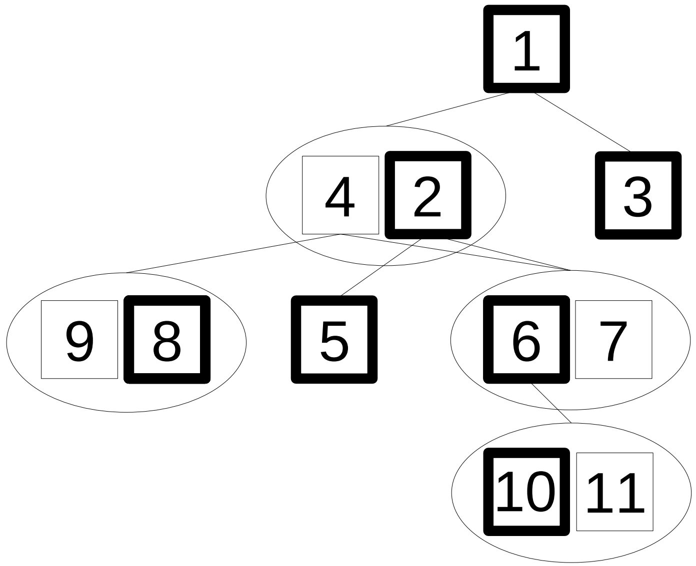
- Group passive edges into *equivalence classes* by identity of α, i, and j;
- search chart for existing equivalent edge (h, say) for each new edge e;
- when h (the 'host' edge) exists, *pack* e into h to record equivalence;
- e *not* added to the chart, no derivations with or further processing of e;

$\rightarrow$ *unpacking* multiply out all alternative daughters for all result edges.

An Example (Hypothetical) Parse Forest



Unpacking: Cross-Multiplying Local Ambiguity



$$\boxed{1} \rightarrow \langle \boxed{2}\,\boxed{3} \rangle \mid \langle \boxed{4}\,\boxed{3} \rangle$$
$$\boxed{2} \rightarrow \langle \boxed{5}\,\boxed{6} \rangle \mid \langle \boxed{5}\,\boxed{7} \rangle$$
$$\boxed{4} \rightarrow \langle \boxed{8}\,\boxed{6} \rangle \mid \langle \boxed{8}\,\boxed{7} \rangle \mid \langle \boxed{9}\,\boxed{6} \rangle \mid \langle \boxed{9}\,\boxed{7} \rangle$$
$$\boxed{6} \rightarrow \langle \boxed{10} \rangle \mid \langle \boxed{11} \rangle$$

How many complete trees in total?

Generalized Chart Parsing

Initialization

- for each *word* in input string
  - add passive lexical edge $\langle word \bullet \rangle$ to chart
  - for each $\alpha \to word \in P$
    - add passive $\langle \alpha \to word \bullet \rangle$ edge to agenda

Main Loop

- while *edge* ← pop-agenda()
  - if equivalent edge in chart, pack; otherwise add *edge* and
  - if *edge* is passive
    - for each active edge *a* to the left, fundamental-rule(*a*, *edge*)
    - predict new edges from *P*, and add to the agenda
  - else
    - for each passive edge *p* to the right, fundamental-rule(*edge*, *p*)

Termination

- return all edges with category *S* that span the full input

Viterbi Decoding over the Parse Forest

- Recall the Viterbi algorithm for HMMs

$$v_i(x) = \max_{k=1}^{L} [v_{i-1}(k) \cdot P(x|k) \cdot P(o_i|x)]$$

- For our trees, we no longer have a linear order, but we still build up cached Viterbi values successively:

$$v(e) = \max \left[P(\beta_1, \ldots \beta_n | \alpha) \times \prod_i v(\beta_i) \right]$$

- Similar to HMM decoding, we also need to keep track of the set of daughters that led to the maximum probability.
- Implementation: cache the highest-scoring edge within e, recording the maximum probability of its sub-tree, and the daughter sequence that led to it.

Chart Parsing Summary

- Organize *edges* in an *agenda*, and process them sequentially.
- A passive edge is complete, an active edge is still looking for daughters.
- Processing records the edge in the chart, and then may add other edges to the agenda, either through the fundamental rule, or through (active) edge prediction, or both.
- The edge data structure records:
  - category (LHS)
  - seen elements of rule RHS as daughter *edges*
  - unseen (unanalyzed) elements of rule RHS
  - input span covered (as start and end chart vertices)
  - probability of the sub-tree, reflecting rule probabilities
  - highest-scoring (cached) edge after Viterbi processing
- The agenda ordering and prediction strategy influence the search order (which can be varied fully freely).

Parser Evaluation

There are a number of aspects to consider in judging parser performance:

- **Coverage** the percentage of inputs for which we we found an analysis.
- **Overgeneration** the percentage of *ungrammatical* inputs (incorrectly) assigned an analysis.
- **Efficiency** time and memory used by the parser.
- **Accuracy** Sentence accuracy measures the percentage of input sentences which received the right tree.

  Since full trees can be quite complex, this is a very **strict** metric, and so most statistical parsers report accuracy according to the **granular** ParsEval metric.

ParsEval

- The ParsEval metric (Black, et al., 1991) measures constituent overlap.
- The original formulation only considered the shape of the (unlabeled) bracketing.
- The modern ‘standard’ uses a tool called `evalb`, which reports precision, recall and F_1 score for **labeled** brackets, as well as the number of crossing brackets.

ParsEval

Gold Standard

(NP (DT *a*)
(ADVP (RB *pretty*)
(JJ *big*))
(NOM (NN *dog*)
(POS *'s*)
(NN *house*)))

0,6	NP	1,2	RB	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,3	ADVP	3,6	NOM	5,6	NN

System Output

(NP (DT *a*)
(JJ *pretty*)
(NOM (JJ *big*)
(NOM (NN *dog*)
(POS *'s*)
(NN *house*))))

0,6	NP	2,6	NOM	3,4	NN
0,1	DT	2,3	JJ	4,5	POS
1,2	JJ	3,6	NOM	5,6	NN

Recall: $\frac{\textit{Correct}}{\textit{Gold}} = \frac{7}{9}$ Precision: $\frac{\textit{Correct}}{\textit{System}} = \frac{7}{9}$ F_1 score: $\frac{7}{9}$

Crossing Brackets: 1

Finally: I Might Need Some Bonus Points

Rules of the Game

- Up to **four** bonus points towards completion of Obligatory Exercise (3).
- Get one post-it; at the top, write down your **first** and **last** name.
- Further, write down your **UiO account name** (e.g. **oe**, in my case).
- Write each answer on a line of its own, prefix by **question number**.
- Do **not** consult with your neighbors; they will likely mess things up.

After the Quiz

- Post your answers at the **front of your table**, we will collect all notes.
- Discuss your answers with your neighbor(s); explain why **you are right**.

Question (1): HMM Viterbi vs. Forward

Recall the recursive formulation of the Viterbi Algorithm:

$$v_i(x) = \max_{k=1}^{L} [v_{i-1}(k) \cdot P(x|k) \cdot P(o_i|x)]$$

(1) What is different in the Forward Algorithm; and what HMM-related task does it compute?

Question (2): Natural Language Ambiguity

Assume a 'toy' grammar of English:

$$S \rightarrow NP$$
$$NP \rightarrow Det\ NOM$$
$$NOM \rightarrow NOM\ NOM$$

Det → *the*

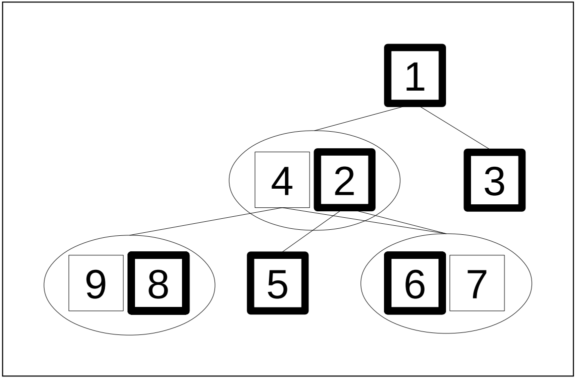
NOM → *kitchen* | *gold* | *towel* | *rack*

(2) How many different syntactic analyses, if any, does the grammar assign to the following strings?

(a) *the kitchen towel rack*

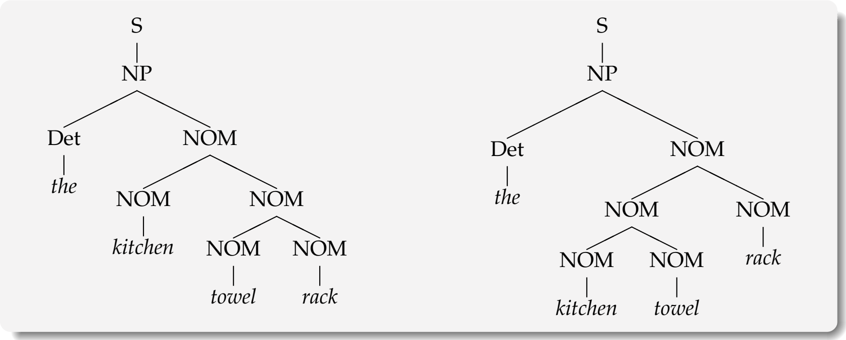
(b) *the kitchen gold towel rack*

Question (3): Packed Parse Forests



(3) How many complete trees are represented in this forest?

Question (4): Parser Evaluation



(4) What are the ParsEval precision and recall scores for this pair of trees (gold on the left; system on the right)?

In conclusion

In the second half of the class, we set out to determine:

- which string is most likely: ✓
  - *How to recognise speech* vs. *How to wreck a nice beach*

- which tag sequence is most likely for *flies like flowers*: ✓
  - **NNS VB NNS** vs. **VBZ P NNS**

- which syntactic analysis is most likely: ✓