The move followed a round of similar increases by other lenders, reflecting a continuing decline in that market.

### Syntax

**Phrase Structure Parsing**
- Phrase structure parsing organizes syntax into constituents or brackets
- In general, this involves nested trees
- Linguists can, and do, argue about details
- Lots of ambiguity
- Not the only kind of syntax...

new art critics write reviews with computers

### Constituency Tests

- How do we know what nodes go in the tree?

- **Classic constituency tests:**
  - Substitution by proform
  - Question answers
  - Semantic grounds
    - Coherence
    - Reference
    - Idioms
    - Dislocation
    - Conjunction

- Cross-linguistic arguments, too

### Conflicting Tests

- **Constituency isn’t always clear**
  - Units of transfer:
    - think about → penser à
    - talk about → hablar de

  - Phonological reduction:
    - I will go → I’ll go
    - I want to go → I wanna go
    - a le centre → au centre

  - Coordination
    - He went to and came from the store.
Classical NLP: Parsing

- Write symbolic or logical rules:
  
  Grammar (CFG)  |  Lexicon
  ---|---
  ROOT → S  |  NP → NP PP
  S → NP VP  |  VP → VBP NP
  NP → DT NN  |  VP → VBP PP
  NP → NN NNS  |  PP → IN NP
  ...  |  VBP → raises

- Use deduction systems to prove parses from words
  - Minimal grammar on "Fed raises" sentence: 36 parses
  - Simple 10-rule grammar: 592 parses
  - Real-size grammar: many millions of parses

- This scaled very badly, didn't yield broad-coverage tools

Ambiguities

Ambiguities: PP Attachment

Attachments

- I cleaned the dishes from dinner
- I cleaned the dishes with detergent
- I cleaned the dishes in my pajamas
- I cleaned the dishes in the sink

Syntactic Ambiguities I

- Prepositional phrases:
  They cooked the beans in the pot on the stove with handles.

- Particle vs. preposition:
  The puppy tore up the staircase.

- Complement structures
  The tourists objected to the guide that they couldn't hear. She knows you like the back of her hand.

- Gerund vs. participial adjective
  Visiting relatives can be boring. Changing schedules frequently confused passengers.

Syntactic Ambiguities II

- Modifier scope within NPs
  impractical design requirements
  plastic cup holder

- Multiple gap constructions
  The chicken is ready to eat.
  The contractors are rich enough to sue.

- Coordination scope:
  Small rats and mice can squeeze into holes or cracks in the wall.
Dark Ambiguities

- Dark ambiguities: most analyses are shockingly bad (meaning, they don’t have an interpretation you can get your mind around)

  This analysis corresponds to the correct parse of “This will panic buyers!”

- Unknown words and new usages
- Solution: We need mechanisms to focus attention on the best ones, probabilistic techniques do this

Probabilistic Context-Free Grammars

- A context-free grammar is a tuple \( <N, T, S, R> \)
  - \( N \): the set of non-terminals
    - Phrasal categories: S, NP, VP, ADJP, etc.
    - Parts-of-speech (pre-terminals): NN, JJ, DT, VB
  - \( T \): the set of terminals (the words)
  - \( S \): the start symbol
    - Often written as ROOT or TOP
    - Not usually the sentence non-terminal \( S \)
  - \( R \): the set of rules
    - Of the form \( X \Rightarrow Y_1 Y_2 … Y_k \), with \( X, Y_i \in N \)
    - Examples: \( S \Rightarrow NP \ VP \), \( VP \Rightarrow VBD \ ADJP \)
    - Also called rewrites, productions, or local trees

- A PCFG adds:
  - A top-down production probability per rule \( P(Y_1 Y_2 … Y_k | X) \)

PCFGs

Treebank Sentences

\[
(\text{S} (\text{NP-SBJ The move}) \text{VP followed} (\text{NP (NP a round)}) \text{PP of} (\text{NP (NP similar increases}) \text{PP by} (\text{NP other lenders})) \text{PP against} (\text{NP Arizona real estate loans})) (\text{NP (NP that market))))\]

Treebank Grammars

- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn’t work well):

Treebank Grammar Scale

- Treebank grammars can be enormous
  - As FSAs, the raw grammar has ~10K states, excluding the lexicon
  - Better parsers usually make the grammars larger, not smaller

- Better results by enriching the grammar (e.g., lexicalization).
- Can also get reasonable parsers without lexicalization.
Chomsky Normal Form

- Chomsky normal form:
  - All rules of the form $X \rightarrow YZ$ or $X \rightarrow w$
  - In principle, this is no limitation on the space of (P)CFGs
  - N-ary rules introduce new non-terminals

-Unary / empties are “promoted”
  - In practice it’s kind of a pain:
  - Reconstructing n-aries is easy
  - Reconstructing unaries is trickier
  - The straightforward transformations don’t preserve tree scores
  - Makes parsing algorithms simpler!

CKY Parsing

A Recursive Parser

- Will this parser work?
- Why or why not?
- Memory requirements?

A Memoized Parser

- One small change:

A Bottom-Up Parser (CKY)

- Can also organize things bottom-up

Unary Rules

- Unary rules?
**CNF + Unary Closure**

- We need unaries to be non-cyclic
  - Can address by pre-calculating the unary closure
  - Rather than having zero or more unaries, always have exactly one

- Alternate unary and binary layers
- Reconstruct unary chains afterwards

**Alternating Layers**

- $\text{bestScoreU}(X, i, j, s)$
  - if $(j = i+1)$
    - return $\text{tagScore}(X, s[i])$
  - else
    - return $\max \max \text{score}(X \rightarrow Y) \times \text{bestScoreU}(Y, i, k) \times \text{bestScoreU}(Z, k, j)$

**Memory**

- How much memory does this require?
  - Have to store the score cache
  - Cache size: $|\text{symbols}| \times n^2$ doubles
  - For the plain treebank grammar:
    - $X \sim 20K$, $n \sim 40$, double $\sim 8$ bytes $\sim 256\text{MB}$
    - Big, but workable.

- Pruning: Beams
  - $\text{score}(X[i][j])$ can get too large (when?)
  - Can keep beams (truncated maps $\text{score}[i][j]$) which only store the best few scores for the span $[i, j]$.

- Pruning: Coarse-to-Fine
  - Use a smaller grammar to rule out most $X[i, j]$.
  - Much more on this later...

**Analysis**

**Time: Theory**

- How much time will it take to parse?
  - For each diff $(\leq n)$
    - For each $i (\leq n)$
      - For each rule $X \rightarrow Y Z$
        - For each split point $k$
          - Do constant work
  - Total time: $|\text{rules}| \times n^3$
  - Something like 5 sec for an unoptimized parse of a 20-word sentences

**Time: Practice**

- Parsing with the vanilla treebank grammar:
  - ~20K Rules (not an optimized parser!) Observed exponent: 3.6

- Why’s it worse in practice?
  - Longer sentences “unlock” more of the grammar
  - All kinds of systems issues don’t scale
**Same-Span Reachability**

- ADJP ADVP
- FRAG INTJ NP
- PP PRN OP S
- SBAR UCP VP
- WHNP
- TOP
- NO
- SQ
- X
- RRC
- LST
- CONJP
- NAC
- SBARQ
- SINV
- RRC
- SQ
- PRT
- X

- Many states are more likely to match larger spans!

**Rule State Reachability**

- Example: NP CC •
  - NP x CC
  - 0 n 1 Alignment

- Example: NP CC NP •
  - NP x CC x NP
  - 0 n-k 1 n Alignments

- Many states are more likely to match larger spans!

**Efficient CKY**

- Lots of tricks to make CKY efficient
  - Some of them are little engineering details:
    - E.g., first choose k, then enumerate through the Y[i,j] which are non-zero, then loop through rules by left child.
    - Optimal layout of the dynamic program depends on grammar, input, even system details.
  - Another kind is more important (and interesting):
    - Many X[i,j] can be suppressed on the basis of the input string
    - We’ll see this next class as figures-of-merit, A* heuristics, coarse-to-fine, etc

**Agenda-Based Parsing**

- Agenda-based parsing is like graph search (but over a hypergraph)
- Concepts:
  - Numbering: we number fenceposts between words
  - "Edges" or items: spans with labels, e.g. PP[3,5], represent the sets of trees over those words rooted at that label (cf. search states)
  - A chart: records edges we’ve expanded (cf. closed set)
  - An agenda: a queue which holds edges (cf. a fringe or open set)

**Word Items**

- Building an item for the first time is called discovery. Items go into the agenda on discovery.
- To initialize, we discover all word items (with score 1.0).

AGENDA

<table>
<thead>
<tr>
<th>critics[0,1], write[1,2], reviews[2,3], with[3,4], computers[4,5]</th>
</tr>
</thead>
</table>

CHART [EMPTY]
**Unary Projection**

- When we pop a word item, the lexicon tells us the item successors (and scores) which go on the agenda

![Unary Projection Diagram]

**Item Successors**

- When we pop items off of the agenda:
  - Graph successors: unary projections (NNS → critics, NP → NNS)
    \[ Y[i,j] \text{ with } X \rightarrow Y \text{ forms } X[i,j] \]
  - Hypergraph successors: combine with items already in our chart
    \[ Y[i,j] \text{ and } Z[j,k] \text{ with } X \rightarrow Y Z \text{ form } X[i,k] \]
- Queries a chart must support:
  - Is edge \( X[i,j] \) in the chart? (What score?)
  - What edges with label \( Y \) end at position \( j \)?
  - What edges with label \( Z \) start at position \( i \)?

![Item Successors Diagram]

**An Example**

- Sometimes we want to posit nodes in a parse tree that don’t contain any pronounced words:
  - I want you to parse this sentence
  - I want [ ] to parse this sentence
- These are easy to add to a chart parser!
  - For each position \( i \), add the “word” edge \( c[i,i] \)
  - Add rules like \( NP \rightarrow \epsilon \) to the grammar
  - That’s it!

![An Example Diagram]

**Empty Elements**

- With weighted edges, order matters
  - Must expand optimal parse from bottom up (subparses first)
  - CKY does this by processing smaller spans before larger ones
  - UCS pops items off the agenda in order of decreasing score
  - \( A^* \) search also well defined
- You can also speed up the search without sacrificing optimality
  - Can select which items to process first
  - Can do with any “figure of merit” [Charniak 98]
  - If your figure-of-merit is a valid \( A^* \) heuristic, no loss of optimality [Klein and Manning 03]

![Empty Elements Diagram]

**UCS / A***

- There was nothing magical about words spanning exactly one position.
- When working with speech, we generally don’t know how many words there are, or where they break.
- We can represent the possibilities as a lattice and parse these just as easily.

![UCS / A* Diagram]

**Speech) Lattices**