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

---

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
  - Reference
  - Idioms
  - Dislocation
  - Conjunction
- Cross-linguistic arguments, too

Conflicting Tests

- Constituency isn’t always clear
- Units of transfer:
  - think about → pensar a
  - 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.

Structure Depth

- Q: Do we model deep vs surface structure?

[Example: Johnson (02)]

[Example: Johnson (02)]
Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes

<table>
<thead>
<tr>
<th>Open class (lexical) words</th>
<th>Closed class (functional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nouns</td>
<td>Determiners</td>
</tr>
<tr>
<td>Pronouns</td>
<td>Prepositions</td>
</tr>
<tr>
<td>Article</td>
<td>Conjunctions</td>
</tr>
<tr>
<td>Adjectives</td>
<td>Adverbs</td>
</tr>
<tr>
<td>Prepositions</td>
<td>Interjections</td>
</tr>
<tr>
<td>Voice</td>
<td>Numbers</td>
</tr>
<tr>
<td>Tense</td>
<td>Dates</td>
</tr>
<tr>
<td>Mood</td>
<td>Time</td>
</tr>
<tr>
<td>Sentence structure</td>
<td>Time expressions</td>
</tr>
<tr>
<td>Punctuation</td>
<td>Place</td>
</tr>
<tr>
<td>Exclamation mark</td>
<td>Time expressions</td>
</tr>
</tbody>
</table>

- Words can have multiple parts of speech

```
Fed raises interest rates 0.5 percent

Fed    VBD    raises    raise
rates  NNS    0.5    .5
percent NNS   NN    NN
```

- Two basic sources of constraint:
  - Grammatical environment
  - Identity of the current word

- Many more possible features:
  - Suffixes, capitalization, name databases (gazetteers), etc.
Why POS Tagging?

- Useful in and of itself (more than you’d think)
  - Text-to-speech: record, lead
  - Lemmatization: saw[x] → see, saw[n] → saw
- Quick-and-dirty NP-chunk detection: grep [U] [NN* [NN | NNS]]
- Useful as a pre-processing step for parsing
  - Less tag ambiguity means fewer parses
    - However, some tag choices are better decided by parsers

```
DT NNP NN VBD VBN IN NN NNS
The Georgia branch had taken on loan commitments …

DT NN IN NN VBD NNS VBD
The average of interbank offered rates plummeted …
```

Classical NLP: Parsing

- Write symbolic or logical rules:
  - 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

```
ROOT → S
S → NP VP
NP → DT NN
VP → VBP NP
VBP → raises
NP → NN NNS
PP → IN NP
VBD → raises
```

Ambiguities: PP Attachment

- 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

Attachments

```
The board approved a supplemental bond issuance (the bond issuance (of parents) 
for K.M. in a house) 
let the monthly meeting) …
```
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. participle 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)

- Unknown words and new usages

- Solution: We need mechanisms to focus attention on the best ones; probabilistic techniques do this

Ambiguities as Trees
PCFGs

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.
  - \(T\): the set of terminals (the words)
  - \(S\): the start symbol
    - Often written as ROOT or TOP
  - \(R\): the set of rules
    - Of the form \(X \rightarrow Y_1 Y_2 \ldots Y_k\), with \(X, Y_i \in N\)
    - Examples: \(S \rightarrow NP \text{ VP}\), \(VP \rightarrow VP \text{ CC VP}\)
    - Also called rewrites, productions, or local trees
  - A PCFG adds:
    - A top-down production probability per rule \(P(Y_1 Y_2 \ldots | X)\)

Treebank Sentences

- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn’t work well):
- Better results by enriching the grammar (e.g., lexicalization).
- Can also get state-of-the-art parsers without lexicalization.
Treebank Grammar Scale

- Treebank grammars can be enormous
  - As E16A, the raw grammar has ~50K states, excluding the lexicon
  - Better parsers usually make the grammars larger, not smaller

Chomsky Normal Form

- Chomsky normal form:
  - All rules of the form $X \rightarrow Y Z$ or $X \rightarrow w$
  - In principle, this is no limitation on the space of (P)CFGs
  - In practice, this is kind of a pain:
    - Unaries / empties are "promoted"
    - Reconstructing unaries is trickier
    - Reconstructing n-aries is easier
    - The straightforward transformations don’t preserve tree scores
    - Makes parsing algorithms simpler!

CKY Parsing

```
bestScore(X, i, j) =
  if (j = i+1)
    return tagScore(X, s[i])
  else
    return max(score(X-YZ) * bestScore(Y, i, k) * bestScore(Z, k, j))
```

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

- One small change:

\[
\text{bestScore}(X, i, j) =
\begin{cases}
\text{null} & \text{if } (j - i = 1) \\
\text{tagScore}(X, s[i]) & \text{else}
\end{cases}
\]

\[
\text{bestScore}(X, i, j) = \max \left( \text{bestScore}(Y, i, k) \times \text{bestScore}(Z, k, j) \right) 
\]

A Bottom-Up Parser (CKY)

- Can also organize things bottom-up

\[
\text{bestScore}(X) =
\begin{cases}
\text{null} & \text{if } (j - i = 1) \\
\text{tagScore}(X, s[i]) & \text{else}
\end{cases}
\]

\[
\text{bestScore}(X, i, j) = \max \left( \text{bestScore}(X, i, j), \right.
\]

\[
\left. \text{bestScore}(Y, i, k) \times \text{bestScore}(Z, k, j) \right)
\]

Unary Rules

- Unary rules?

\[
\text{bestScore}(X, i, j) =
\begin{cases}
\text{null} & \text{if } (j - i = 1) \\
\text{tagScore}(X, s[i]) & \text{else}
\end{cases}
\]

\[
\text{bestScore}(X, i, j) = \max \left( \text{bestScore}(X, i, j), \right.
\]

\[
\left. \text{bestScore}(Y, i, k) \times \text{bestScore}(Z, k, j) \right)
\]

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

```python
bestScoreU(X, i, j, s)
return max score X \rightarrow Y * bestScoreU(Y, i, j)
bestScoreB(X, i, j, s)
return max score X \rightarrow Y * bestScoreB(Y, i, j) * bestScoreU(Z, k, j)
```

Learning PCFGs

Treebank PCFGs [Charniak 96]
- Use PCFGs for broad coverage parsing
- Can take a grammar right off the trees (doesn't work well):

```
ROOT ® S
S ® NP VP .
NP ® PRP
VP ® VBD ADJP
...
```

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>72.0</td>
</tr>
</tbody>
</table>

Conditional Independence?
- Not every NP expansion can fill every NP slot
- A grammar with symbols like "NP" won't be context-free
- Statistically, conditional independence too strong
Non-Independence

- Independence assumptions are often too strong.

- Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).

- Also: the subject and object expansions are correlated!

Grammar Refinement

- Example: PP attachment

Structural Annotation

- Structure Annotation [Johnson '98, Klein&Manning '03]
- Lexicalization [Collins '99, Charniak '00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. '06]
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
- Structural annotation

Lexicalization

- Annotation refines base treebank symbols to improve statistical fit of the grammar
- Structural annotation [Johnson '98, Klein and Manning '03]
- Head lexicalization [Collins '99, Charniak '00]

Problems with PCFGs

- If we do no annotation, these trees differ only in one rule:
  - VP → VP PP
  - NP → NP PP
- Parse will go one way or the other, regardless of words
- We addressed this in one way with unlexicalized grammars (how?)
- Lexicalization allows us to be sensitive to specific words
Problems with PCFGs

- What’s different between basic PCFG scores here?
- What (lexical) correlations need to be scored?

Lexicalized Trees

- Add “head words” to each phrasal node
  - Syntactic vs. semantic heads
  - Headship not in (most) treebanks
- Usually use head rules, e.g.
  - NP: Take leftmost NP
  - Take rightmost N*
  - Take right child
  - VP: Take leftmost VB*
  - Take rightmost VP
  - Take left child

Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like
  \[ TP(\text{sex}) \rightarrow TP(\text{sex}) \rightarrow TP(\text{sex}) \rightarrow TP(\text{today}) \]
- Never going to get these atomically off of a treebank
- Solution: break up derivation into smaller steps

Lexical Derivation Steps

- A derivation of a local tree [Collins 99]
  - Choose a head tag and word
  - Choose a complement bag
  - Generate sub-trees (incl. adjuncts)
  - Recursively derive sub-trees
Lexicalized CKY

```plaintext
bestScore(X, i, j, h)
if (j = i+1)
    return tagScore(X, s[i])
else
    return max max
           score(X[h] - Y[h] Z[h']) * 
           bestScore(Y, i, k, h) * 
           bestScore(Z, k, j, h')
```

Results

- Some results
  - Collins 99 – 88.6 F1 (generative lexical)
  - Charniak and Johnson 05 – 89.7 / 91.3 F1 (generative lexical / reranked)
  - Petrov et al 06 – 90.7 F1 (generative unlexical)
  - McClosky et al 06 – 92.1 F1 (gen + rerank + self-train)

- However
  - Bilexical counts rarely make a difference (why?)
  - Gildea 05 – Removing bilexical counts costs < 0.5 F1

Latent Variable PCFGs

The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Parent annotation [Johnson '98]
  - Head lexicalization [Collins '99, Charniak '00]
  - Automatic clustering?
Latent Variable Grammars

Parse Tree
Sentence \( \mathcal{T} \)
Derivations \( \mathcal{D} \) Parameters \( \theta \)

Learning Latent Annotations

EM algorithm:
- Brackets are known
- Base categories are known
- Only induce subcategories

Refinement of the DT tag

Hierarchical refinement
Hierarchical Estimation Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat Training</td>
<td>87.3</td>
</tr>
<tr>
<td>Hierarchical Training</td>
<td>88.4</td>
</tr>
</tbody>
</table>

Refinement of the , tag

- Splitting all categories equally is wasteful.

Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful

Adaptive Splitting Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>88.4</td>
</tr>
<tr>
<td>With 50% Merging</td>
<td>89.5</td>
</tr>
</tbody>
</table>
Learned Splits

- Proper Nouns (NNP):
  - NNP-12: John, Robert, James
  - NNP-2: J., E., L.
  - NNP-1: Bush, Noriega, Peters
  - NNP-16: New, San, Wall
  - NNP-3: York, Francisco, Street

- Personal pronouns (PRP):
  - PRP-0: It, He, I
  - PRP-1: It, He, They
  - PRP-2: It, Them, Him

Learned Splits

- Relative adverbs (RBR):
  - RBR-0: further, lower, higher
  - RBR-1: more, less, More
  - RBR-2: earlier, Earlier, later

- Cardinal Numbers (CD):
  - CD-7: one, two, Three
  - CD-11: million, billion, trillion
  - CD-6: 1, 50, 100
  - CD-3: 1, 30, 31
  - CD-6: 78, 58, 34
Coarse-to-Fine Inference

- Example: PP attachment

Hierarchical Pruning

- Coarse: 
- Split in two: 
- Split in four: 
- Split in eight: 

Bracket Posteriors

Other Syntactic Models
**Parse Reranking**
- Assume the number of parses is very small.
- We can represent each parse $T$ as a feature vector $\varphi(T)$.
- Typically, all local rules are features.
- Also non-local features, like how right-branching the overall tree is.
- [Charniak and Johnson 05] gives a rich set of features.

**Dependency Parsing**
- Lexicalized parsers can be seen as producing dependency trees.
- Each local binary tree corresponds to an attachment in the dependency graph.

**Dependency Parsing**
- Pure dependency parsing is only cubic [Eisner 99].
- Some work on non-projective dependencies:
  - Common in, e.g., Czech parsing.
  - Can do with MST algorithms [McDonald and Pereira 05].

**Shift-Reduce Parsers**
- Another way to derive a tree:
  - No useful dynamic programming search.
  - Can still use beam search [Ratnaparkhi 97].
Data-oriented parsing:
- Rewrite large (possibly lexicalized) subtrees in a single step
- Formally, a tree-insertion grammar
- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable parse is NP-complete

Tree-adjoining grammars
- Start with local rules
- Can insert structure with adjunction operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- ... as well as other weird stuff that CFGs don’t capture well (e.g., cross-serial dependencies)
CCG Parsing

- Combinatory Categorial Grammar
- Fully lexicalized grammar
- Categories encode argument sequences
- Very closely related to the lambda calculus (more later)
- Can have spurious ambiguities (why?)

```
John => NP
shares => NP

boys => (S\NP)/NP

sleeps => S\NP

well => (S\NP)\(S\NP)
```

```
S
  /                
NP    S\NP
     /                 
John (S\NP)/NP   NP
       \                  
         boys    shares
```