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

**Parse Trees**

**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 “pensar a”
  - talk about “hablar de”
- Phonological reduction:
  - I will go → I’ll go
  - I want to go → I wanna go
  - a la 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: Cai et al 11]
```

Ambiguities

```
```

Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes

```
```

Part-of-Speech Ambiguity

- Words can have multiple parts of speech
  - Fed raises interest rates 0.5 percent
    - FBN: feeds [verb/feeds] as a verb
    - FBN: feeds [verb/feeds] as a verb
    - FBN: feeds [verb/feeds] as a verb

- 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[v] → see, saw[n] → saw
  - Quick-and-dirty NP-chunk detection: grep [JJ | 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  RP  NN  VBD
  The Georgia branch had taken on loan commitments ...
  DT  NN  IN  NN  VBD  NNS  VBD  VBD
  The average of interbank offered rates plummeted ...
```

Classical NLP: Parsing

- Write symbolic or logical rules:
  - Grammar (CFG) Lexicon
  - ROOT → S
  - NP → NP/NP
  - S → NP VP
  - VP → VBP/NP NNS → raises
  - NP → DT NN
  - VP → VBP/NP PP
  - VBP → raises
  - NP → NN NNS
  - PP → IN NP
  - This scaled very badly, didn’t yield broad-coverage tools

Ambiguities: PP Attachment

```
<table>
<thead>
<tr>
<th>Prepositional phrases:</th>
</tr>
</thead>
<tbody>
<tr>
<td>They cooked the beans in the pot on the stove with handles.</td>
</tr>
<tr>
<td>Particle vs. preposition:</td>
</tr>
<tr>
<td>The puppy tore up the staircase.</td>
</tr>
<tr>
<td>Complement structures</td>
</tr>
<tr>
<td>The tourists objected to the guide that they couldn’t hear.</td>
</tr>
<tr>
<td>She knows you like the back of her hand.</td>
</tr>
<tr>
<td>Gerund vs. participial adjective</td>
</tr>
<tr>
<td>Visiting relatives can be boring.</td>
</tr>
<tr>
<td>Changing schedules frequently confused passengers.</td>
</tr>
</tbody>
</table>
```

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

```
<table>
<thead>
<tr>
<th>Modifier scope within NPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>impractical design requirements</td>
</tr>
<tr>
<td>plastic cup holder</td>
</tr>
<tr>
<td>Multiple gap constructions</td>
</tr>
<tr>
<td>The chicken is ready to eat.</td>
</tr>
<tr>
<td>The contractors are rich enough to sue.</td>
</tr>
<tr>
<td>Coordination scope:</td>
</tr>
<tr>
<td>Small rats and mice can squeeze into holes or cracks in the wall.</td>
</tr>
</tbody>
</table>
```
**Inaccessible Ambiguities**

- **Inaccessible 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

**Probabilistic Context-Free Grammars**

- A context-free grammar is a tuple \( \langle N, T, S, R \rangle \)
  - \( N \): 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 \( X \rightarrow Y_1 Y_2 \ldots Y_k \), with \( X, Y_i \in N \)
    - Examples: \( S \rightarrow NP \ VP \), \( VP \rightarrow VP \ CC \ VP \)
    - Also called rewrites, productions, or local trees
  - A PCFG adds:
    - A top-down production probability per rule \( P(Y_1 Y_2 \ldots Y_k | X) \)

**Treebank Sentences**

```
( S (NP (SB) The move) (VP followed (PP of the) (NP (NP similar increases) (PP by (NP other lenders))) (PP against (NP Arizona real estate loans))))
( S (ADV:NP:SB) )
( VP (reflecting) (NP (NP a continuing decline) (PP LOC:SB (NP that market)))))
. )
```

**Treebank Grammars**

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

```
ROOT \rightarrow S 1
S \rightarrow NP VP . 1
NP \rightarrow PRP 1
VP \rightarrow VBD ADJP 1
```

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

Chomsky Normal Form

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

- Unaries/empties are "promoted"
- In practice it's kind of a pain:
  - Reconstructing unaries is easier
  - Reconstructing unaries is trickier
  - The straightforward transformations don't preserve tree scores
- Makes parsing algorithms simpler!

NP

VP

A Recursive Parser

CKY Parsing

A Memoized Parser

A Bottom-Up Parser (CKY)
Unary Rules

- Unary rules?

```python
bestScore(X, i, j, s)
if (j = i+1)
    return tagScore(X, s[i])
else
    return max
        max score(X->Y) * bestScoreB(Y, i, j)
        bestScoreU(X, i, j, s)
```

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

```
bestScoreU(X, i, j, s)
if (j = i+1)
    return tagScore(X, s[i])
else
    return max
        max score(X->Y) * bestScoreB(Y, i, j)
        bestScoreU(X, i, j, s)
```

Alternating Layers

```
bestScoreB(X, i, j, s)
return max
    max score(X->Y) * bestScoreB(Y, i, j)
    bestScoreU(X, i, j, s)
```

Learning PCFGs

Treebank PCFGs

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

```
ROOT → S     1
S → NP VP    1
NP → PRP     1
VP → VBD ADJP 1
```

```
Model | F1
------|---
Baseline | 72.0
```

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!

```
All NPs | NPs under S | NPs under VP
NP PP  |   11%   |   23%   
 DT NN  |     8%  |     1%  
 DP PS  |     8%  |     4%  
```

Grammar Refinement

- Example: PP attachment

```
They VP NP
rased...
a point of order
```

Grammar Refinement

```
S
NP-ite
PRP VBD NP-ite
She heard DT NN
the noise
```

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

Structural Annotation

```
The Game of Designing a Grammar
```

```
The Game of Designing a Grammar
```

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

Lexicalization
The Game of Designing a Grammar

- 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:
  - $VP \rightarrow VP PP$
  - $NP \rightarrow NP PP$
- 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 head
  - Headship not in (most) treebanks
- Usually use head rules, e.g.:
  - NP:
    - Take leftmost NP
    - Take rightmost NP
    - Take rightmost JJ
  - VP:
    - Take leftmost VB
    - Take left child
    - Take rightmost NN
      - Take (presumably) left (or) right

Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like $VP(noun) \rightarrow VP(noun) VP(noun) VP(today)$
- 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 tag
  - Generate children (not adjuncts)
  - Recursively derive children
Lexicalized CKY

```latex
bestScore(X, i, j, h)
if (j = i+1)
    return tagScore(X, s[i])
else
    return max \max \max \text{score}(X[h] \rightarrow Y[h] Z[h']) \ast
    \text{bestScore}(Y, i, k, h) \ast \text{bestScore}(Z, k, j, h')
    \max \text{score}(X[h] \rightarrow Y[h'] Z[h]) \ast \text{bestScore}(Y, i, k, h') \ast \text{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 01 – Removing bilexical counts costs < 0.5 F1

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 PCFGs

Latent Variable Grammars

Learning Latent Annotations

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

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

\begin{align*}
K \{X_{1} X_{2} X_{3} X_{4} X_{5} X_{6} X_{7} \} \\
\text{NP} & \rightarrow \text{NP} \text{ VP} \\
\text{VP} & \rightarrow \text{PRP} \text{ VBD} \text{ ADJP} \\
\text{He} & \rightarrow \text{he} \text{ was} \text{ right} \\
\end{align*}
```

Parse Tree 
Sentence $T$
Derivations $t : T$
Parameters $\theta$

```
Forwadd
Backward
```

```
Just like Forward-Backward for HMMs.
```

Refinement of the DT tag

Hierarchical refinement

Hierarchical Estimation Results

Refinement of the , tag

Adaptive Splitting

Adaptive Splitting Results
Learned Splits

- **Proper Nouns (NNP):**
  - NNP-12: John Robert James
  - NNP-2: J. E. L.
  - NNP-1: Bush Noriega Peters
  - NNP-15: 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

- **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-0: 1 50 100
  - CD-3: 1 30 31
  - CD-9: 78 58 34

Natural Language Processing

Syntax and Parsing

Dan Klein – UC Berkeley

Latent Variable Grammars
Coarse-to-Fine Inference

- Example: PP attachment

Hierarchical Pruning

- 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 φ(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

Shift-Reduce Parsers

- Another way to derive a tree:
  - Parsing
    - No useful dynamic programming search
    - Can still use beam search [Ratnaparkhi 97]
Other Transformations

- Example: Left-Corner Transforms, Tetra-Tags

K-Best Parsing

- Lexicalized parsers can be seen as producing dependency trees

Dependency Parsing

- Each local binary tree corresponds to an attachment in the dependency graph

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
TIG: Insertion

- Start with local trees
- 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)

Tree-adjoining grammars

- TAG: Long Distance

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

- CCG Parsing

  - \( John \vdash NP \)
  - \( shares \vdash NP \)
  - \( hays \vdash (S/NP)/NP \)
  - \( sleeps \vdash S/NP \)
  - \( well \vdash (S/NP)/(S/NP) \)