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

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.

Questions from Last Time

- Q: Do we model deep vs surface structure?

[Example: Johnson 02]

[Example: Johnson 02]
Ambiguities

Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes

Open class (lexical) words

<table>
<thead>
<tr>
<th>Nouns</th>
<th>Verbs</th>
<th>Adjectives</th>
<th>Adverbs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proper</td>
<td>Main</td>
<td>Adjectives</td>
<td>Adverbs</td>
</tr>
<tr>
<td>IBM</td>
<td>see</td>
<td>yellow</td>
<td>slowly</td>
</tr>
<tr>
<td>Italy</td>
<td>registered</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Closed class (functional)

<table>
<thead>
<tr>
<th>Determiners</th>
<th>Auxiliary</th>
<th>Conjunctions</th>
<th>Pronouns</th>
</tr>
</thead>
<tbody>
<tr>
<td>the some</td>
<td>can</td>
<td>and or</td>
<td>he its</td>
</tr>
</tbody>
</table>

Part-of-Speech Ambiguity

- Words can have multiple parts of speech

Fed raises interest rates 0.5 percent

Mrs./NNP, Saefer/NNP, never/RB, got/VBD around/TO, to/TO, joining/VBG
All/DT, we/PRP, gotta/VBN, do/VB, is/VBZ, go/VB, around/IN, the/DT, corner/NN
Chateau/NNP, Petrus/NNP, costs/VBZ, around/TO, 250/CD

- 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

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

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

Classical NLP: Parsing

- Write symbolic or logical rules:
  
<table>
<thead>
<tr>
<th>Grammar (CFG)</th>
<th>Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT → S</td>
<td>NN → interest</td>
</tr>
<tr>
<td>S → NP VP</td>
<td>VBP → interest</td>
</tr>
<tr>
<td>NP → DT NN</td>
<td>VBP → raises</td>
</tr>
<tr>
<td>NP → NN NNS</td>
<td>NNS → raises</td>
</tr>
</tbody>
</table>

- 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: PP Attachment

```
NP
DT NNP
The children ate the cake with a spoon
```

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

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

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.
    - 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-SBJ The move))
  (VP followed
    (NP (NP a round)
      (PP of
        (NP (NP similar increases)
          (PP by
            (NP other lenders))
          (PP against
            (NP Arizona real estate loans)))))))

(S-ADV (NP-SBJ 2))
(VP reflecting
  (NP (NP a continuing decline)
    (PP-LOC in
      (NP that market))))

Treebank Grammars

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

\[
\begin{align*}
\text{ROOT} & \rightarrow S & 1 \\
S & \rightarrow NP VP . & 1 \\
NP & \rightarrow PRP & 1 \\
VP & \rightarrow VBD ADJP & 1 \\
\end{align*}
\]

- 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
    - $N$-ary rules introduce new non-terminals
  - Unaries / 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!

A Recursive Parser

```
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:

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

A Bottom-Up Parser (CKY)

- Can also organize things bottom-up

```java
bestScore(s)
    for (i : [0,n-1])
        for (X : tags[s[i]])
            score[X][i][i+1] = tagScore(X,s[i])
    for (diff : [2,n])
        for (i : [0,n-diff])
            j = i + diff
            for (X->YZ : rule)
                for (k : [i+1, j-1])
                    score[X][i][j] = max score[X][i][j],
                        score(X->YZ) *
                        score[Y][i][k] *
                        score[Z][k][j]
```

Unary Rules

- Unary rules?

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

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

\[\text{bestScoreB}(X, i, j, s)\]
\[
\quad \text{return } \max \max \text{ score}(X \rightarrow Y) \times \\
\quad \text{bestScoreU}(Y, i, j) \times \\
\quad \text{bestScoreU}(Z, k, j)\
\]

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

Learning PCFGs

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

\[
\text{ROOT} \rightarrow S \quad 1 \\
S \rightarrow \text{NP VP} . \quad 1 \\
\text{NP} \rightarrow \text{PRP} \quad 1 \\
\text{VP} \rightarrow \text{VBD ADJP} \quad 1 \\
\]

\[
\begin{array}{|c|}
\hline
\text{Model} & F1 \\
\hline
\text{Baseline} & 72.0 \\
\hline
\end{array}
\]

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!

<table>
<thead>
<tr>
<th></th>
<th>NP PP</th>
<th>DT NN</th>
<th>PRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>All NPs</td>
<td>11%</td>
<td>9%</td>
<td>6%</td>
</tr>
<tr>
<td>NPs under S</td>
<td>9%</td>
<td>9%</td>
<td>21%</td>
</tr>
<tr>
<td>NPs under VP</td>
<td>25%</td>
<td>7%</td>
<td>4%</td>
</tr>
</tbody>
</table>

Grammar Refinement

- Example: PP attachment

```
They raised VP NP
a point of order
```

Grammar Refinement

- 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

- 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

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 → 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 leftmost VP
      - Take left child

Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like
  \[ \text{VP(saw)} \rightarrow \text{VBD(saw) NP-C(her) NP(today)} \]

- 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 children (incl. adjuncts)
  - Recursively derive children
Lexicalized CKY

```python
bestScore(X, i, j, h)
  if (j = i+1)
    return tagScore(X, s[i])
  else
    return max
      max
        score(X[h] \to Y[h]\ Z[h']) *
        bestScore(Y, i, k, h) *
        bestScore(Z, k, j, h')
      max
        score(X[h] \to 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 01 – 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

$$\begin{align*}
S &
\rightarrow NP \ VBD \ ADJP \\
S &
\rightarrow NP \ VBD \ ADJP \\
\end{align*}$$

Grammar G

$$\begin{align*}
S &
\rightarrow NP \ VBD \ ADJP \\
S &
\rightarrow NP \ VBD \ ADJP \\
\end{align*}$$

Learning Latent Annotations

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

Just like Forward-Backward for HMMs.

Refinement of the DT tag

DT

- the (0.50)
- a (0.24)
- The (0.08)

Hierarchical refinement

- the (0.54)
- a (0.25)
- The (0.08)

- the (0.54)
- a (0.25)
- The (0.08)

- the (0.54)
- a (0.25)
- The (0.08)
Hierarchical Estimation Results

<table>
<thead>
<tr>
<th>Total Number of grammar symbols</th>
<th>Parsing accuracy (F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>74</td>
</tr>
<tr>
<td>300</td>
<td>88</td>
</tr>
<tr>
<td>500</td>
<td>88</td>
</tr>
<tr>
<td>700</td>
<td>88</td>
</tr>
<tr>
<td>900</td>
<td>88</td>
</tr>
<tr>
<td>1100</td>
<td>90</td>
</tr>
</tbody>
</table>

<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-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  |

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-4     | 1989    | 1990   | 1988   |
  | CD-11    | million | billion | trillion |
  | CD-0     | 1       | 50     | 100    |
  | CD-3     | 1       | 30     | 31     |
  | CD-9     | 78      | 58     | 34     |
Coarse-to-Fine Inference

- Example: PP attachment

![Diagram showing S, NP, VP, PRP, and PP with examples of VP attachments and splitting into two, four, and eight.]

Hierarchical Pruning

- Coarse: NP, VP
- Split in two: NP1, NP2; VP1, VP2
- Split in four: NP1, NP2, VP1, VP2
- Split in eight:...

Bracket Posteriors

- Diagram showing bracket posteriors for various word types and syntactic structures.

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

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:

  Parsing
  - 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 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)
CCG Parsing

- 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?)

\[
\begin{align*}
John & \rightarrow \text{NP} \\
shares & \rightarrow \text{NP} \\
buys & \rightarrow (S\text{\ NP})/\text{NP} \\
sleeps & \rightarrow S\text{\ NP} \\
well & \rightarrow (S\text{\ NP})/(S\text{\ NP})
\end{align*}
\]