Feature-Based Grammar

Ling571
Deep Processing Techniques for NLP
February 4, 2015

Features in CFGs: Agreement

- Goal:
  - Support agreement of NP/VP, Det Nominal
- Approach:
  - Augment CFG rules with features
  - Employ head features
    - Each phrase: VP, NP has head
      - Head: child that provides features to phrase
        - Associates grammatical role with word
        - VP – V; NP – Nom, etc

Simple Feature Grammars

- S -> NP VP

Simple Feature Grammars

- S -> NP[NUM=?n] VP[NUM=?n]
- NP -> N

Simple Feature Grammars

- S -> NP[NUM=?n] VP[NUM=?n]
- NP[NUM=?n] -> N[NUM=?n]
- NP-> PropN

Simple Feature Grammars

- S -> NP[NUM=?n] VP[NUM=?n]
- NP[NUM=?n] -> N[NUM=?n]
- NP[NUM=?n] -> PropN[NUM=?n]
- NP-> Det N

Simple Feature Grammars

- S -> NP[NUM=?n] VP[NUM=?n]
- NP[NUM=?n] -> N[NUM=?n]
- NP[NUM=?n] -> PropN[NUM=?n]
- NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
- Det-> 'this' | 'every’

Simple Feature Grammars

- S -> NP[NUM=?n] VP[NUM=?n]
- NP[NUM=?n] -> N[NUM=?n]
- NP[NUM=?n] -> PropN[NUM=?n]
- NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
- Det[NUM=sg] -> 'this' | 'every’
- Det-> 'these' | 'all'

Simple Feature Grammars

- S -> NP[NUM=?n] VP[NUM=?n]
- NP[NUM=?n] -> N[NUM=?n]
- NP[NUM=?n] -> PropN[NUM=?n]
- NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
- Det[NUM=sg] -> 'this' | 'every'
- Det[NUM=pl] -> 'these' | 'all'
- N-> 'dog' | 'girl' | 'car' | 'child'

Simple Feature Grammars

- S -> NP[NUM=?n] VP[NUM=?n]
- NP[NUM=?n] -> N[NUM=?n]
- NP[NUM=?n] -> PropN[NUM=?n]
- NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
- Det[NUM=sg] -> 'this' | 'every'
- Det[NUM=pl] -> 'these' | 'all'
- N[NUM=sg] -> 'dog' | 'girl' | 'car' | 'child'
- N[NUM=pl] -> 'dogs' | 'girls' | 'cars' | 'children'

Simple Feature Grammars

- S -> NP[NUM=?n] VP[NUM=?n]
- NP[NUM=?n] -> N[NUM=?n]
- NP[NUM=?n] -> PropN[NUM=?n]
- NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
- Det[NUM=sg] -> 'this' | 'every’
- Det[NUM=pl] -> 'these' | 'all’
- N[NUM=sg] -> 'dog' | 'girl' | 'car' | 'child’
- N-> 'dogs' | 'girls' | 'cars' | 'children'

Parsing with Features

- >>> cp = load_parser('grammars/book_grammars/feat0.fcfg’)
- >>> for tree in cp.parse(tokens):
  - ... print(tree)
- (S[] (NP[NUM='sg']
  - (PropN[NUM='sg'] Kim))
  - (VP[NUM='sg', TENSE='pres']
    - (TV[NUM='sg', TENSE='pres'] likes)
    - (NP[NUM='pl'] (N[NUM='pl'] children))))

Feature Applications

- Subcategorization:
  - Verb-Argument constraints
    - Number, type, characteristics of args (e.g. animate)
    - Also adjectives, nouns
- Long distance dependencies
  - E.g. filler-gap relations in wh-questions, rel

Unification and the Earley Parser

- Employ constraints to restrict addition to chart
- Actually pretty straightforward
  - Augment rules with feature structure
  - Augment state (chart entries) with DAG
    - Prediction adds DAG from rule
    - Completion applies unification (on copies)
      - Adds entry only if current DAG is NOT subsumed

Parsing with Features

- One strategy:
  - Parse as usual
  - Test completed parses for unification constraints

Parsing with Features

- One strategy:
  - Parse as usual
  - Test completed parses for unification constraints
- Pros:
  - Simple, requires little modification

Parsing with Features

- One strategy:
  - Parse as usual
  - Test completed parses for unification constraints
- Pros:
  - Simple, requires little modification
- Cons:
  - Wasted effort
  - Builds many partial parses that can't unify

Parsing with Features

- One strategy:
  - Parse as usual
  - Test completed parses for unification constraints

- Pros:
  - Simple, requires little modification
- Cons:
  - Wasted effort
  - Builds many partial parses that can't unify
- Integrate unification in parse construction

Parsing, Unification, & Earley

- Augment existing Earley parser for unification
  - Fairly straightforward
- Modify representations:
  - Augment CFG rules with constraints
    - Use constraints to create feature structure as DAG
  - Add DAG to state representation
    - E.g., S -> • NP VP, [0,0],[],Dag

Integrating Unification

- Main change: Completer
  - Advances • in rules where next constituent matches a just-completed constituent
  - Now, unifies Dag from completed constituent with the part of the feature structure in rules advanced
    - If fails, no new entry in chart
- Second change:
  - Only add state if NOT subsumed by states in chart

Notes on Features

Ling 571
Deep Techniques for NLP
February 4, 2015

Feature Grammar in NLTK

- NLTK supports feature-based grammars
  - Includes ways of associating features with CFG rules
  - Includes readers for feature grammars
    - *.fcfg* files
  - Includes parsers
    - Nltk.parse.FeatureEarleyChartParser

Feature Structures

- >>> fs1 = nltk.FeatStruct("[NUM='pl']")
- >>> print fs1
- [NUM='pl']
- >>> print fs1['NUM']
- pl

- More complex structure
- >>> fs2 = nltk.FeatStruct("[POS='N',
- AGR=[NUM='pl',PER=3]]")

Reentrant Feature Structures

- First instance
  - Parenthesized integer: (1)
- Subsequent instances:
  - ‘Pointer’: -> (1)

  - >>> print nltk.FeatStruct("[A='a', B=(1)[C='c'], D->(1)]”
  - [A = ‘a’]
  - [B = (1) [C = ‘c’]]
  - [D -> (1)]

Augmenting Grammars

- Attach feature information to non-terminals, on
  - N[AGR=[NUM='pl']] -> 'students’
  - N[AGR=[NUM=’sg']] -> 'student’
- So far, all values are literal or reentrant
  - Variables allow generalization: ?a
    - Allows underspecification, e.g. Det[GEN=?a]
  - NP[AGR=?a] -> Det[AGR=?a] N[AGR=?a]

Mechanics

- >>> fs3 = nltk.FeatStruct(NUM='pl',PER=3)

- >>> fs4 = nltk.FeatStruct(NUM='pl')

- >>> print fs4.unify(fs3)
- [NUM = 'pl']
- [PER = 3]

Morphosyntactic Features

- Grammatical feature that influences morphological or syntactic behavior
  - English:
    - Number:
      - Dog, dogs
    - Person:
      - Am; are; is
    - Case:
      - I – me; he – him; etc
    - Countability:

Semantic Features

- Grammatical features that influence semantic(meaning) behavior of associated units
- E.g.:

Semantic Features

- Grammatical features that influence semantic(meaning) behavior of associated units
- E.g.:
  - ?The rocks slept.

Semantic Features

- Grammatical features that influence semantic(meaning) behavior of associated units
- E.g.:
  - ?The rocks slept.
  - ?Colorless green ideas sleep furiously.

Semantic Features

- Many proposed:
  - Animacy: +/-
  - Natural gender: masculine, feminine, neuter
  - Human: +/-
  - Adult: +/-
  - Liquid: +/-
  - Etc.
  - The milk spilled.
  - ?The cat spilled.

Examples

- The climber hiked for six hours.
- The climber hiked on Saturday.
- The climber reached the summit on Saturday.
- *The climber reached the summit for six hours.
- Contrast:

Examples

- The climber hiked for six hours.
- The climber hiked on Saturday.
- The climber reached the summit on Saturday.
- *The climber reached the summit for six hours.

- Contrast:
  - Achievement vs activity

Semantic features & Parsing

- Can filter some classes of ambiguity
  - Old men and women slept.
  - (Old men) and (women) slept.
  - (Old (men and women)) slept.

  - Sleeping people and books lie flat.
  - (Sleeping people) and (books) lie flat.
  - (Sleeping (people and books))lie flat.

Semantic features & Parsing

- Can filter some classes of ambiguity

  - Old men and women slept.
  - (Old men) and (women) slept.
  - (Old (men and women)) slept.

  - Sleeping people and books lie flat.
  - (Sleeping people) and (books) lie flat.
  - *(Sleeping (people and books))lie flat.

Summary

- Features
  - Enable compact representation of grammatical constraints
  - Capture basic linguistic patterns
- Unification
  - Creates and maintains consistency over features
- Integration with parsing allows filtering of ill-formed analyses

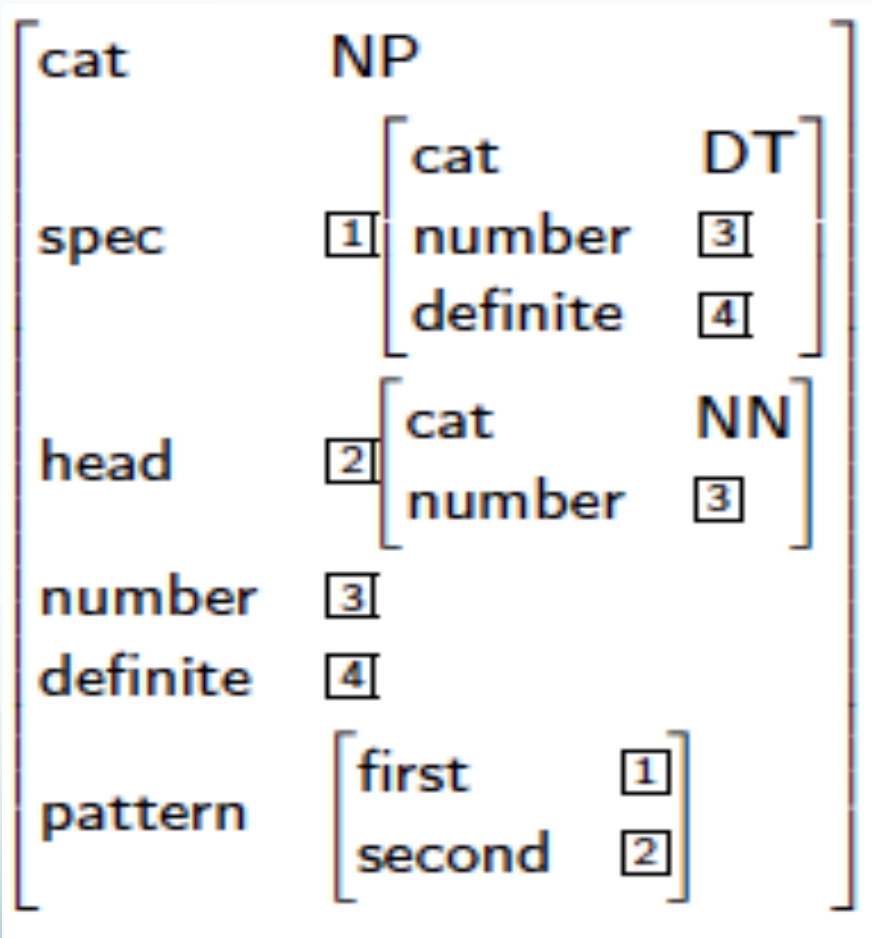
Unification Example

cat	S
voice	active
agent	[1] [cat NP, number [4]]
process	[2] [cat VB, number [4]]
patient	[3] [cat NP]
pattern	[subject [1], verb [2], object [3]]

Grammar entry for sentence

(From S.F., 2010)

Unification Example



Grammar entry for NP

(From S.F., 2010)

Unification Example

cat	DT
definite	yes
number	SG
form	"the"

cat	DT
definite	yes
number	PL
form	"these"

Lexical entries

(From S.F., 2010)

Unification Example

Unifying a noun phrase with a determiner

$$
\begin{bmatrix}
\text{cat} & \text{NP} \\
\text{spec} & \boxed{1}\begin{bmatrix} \text{cat} & \text{DT} \\ \text{number} & \boxed{3} \\ \text{definite} & \boxed{4} \end{bmatrix} \\
\text{head} & \boxed{2}\begin{bmatrix} \text{cat} & \text{NN} \\ \text{number} & \boxed{3} \end{bmatrix} \\
\text{number} & \boxed{3} \\
\text{definite} & \boxed{4} \\
\text{pattern} & \begin{bmatrix} \text{first} & \boxed{1} \\ \text{second} & \boxed{2} \end{bmatrix}
\end{bmatrix}
\sqcup
\begin{bmatrix}
\text{cat} & \text{DT} \\
\text{definite} & \text{yes} \\
\text{number} & \text{PL} \\
\text{form} & \text{"these"}
\end{bmatrix}
$$

(From S.F., 2010)

Unification Example

$$\begin{bmatrix} \text{cat} & \text{DT} \\ \text{number} & \boxed{3} \\ \text{definite} & \boxed{4} \end{bmatrix} \sqcup \begin{bmatrix} \text{cat} & \text{DT} \\ \text{definite} & \text{yes} \\ \text{number} & \text{PL} \\ \text{form} & \text{``these''} \end{bmatrix} = \begin{bmatrix} \text{cat} & \text{DT} \\ \text{definite} & \text{yes} \\ \text{number} & \text{PL} \\ \text{form} & \text{``these''} \end{bmatrix}$$

Unifying NP with Determiner

(From S.F., 2010)

Unification Example

Result of unification

```
[cat       NP                              ]
[          [cat       DT      ]            ]
[          [number    PL      ]            ]
[spec    1 [definite  yes     ]            ]
[          [form      "these" ]            ]
[                                          ]
[head    2 [cat       NN]                  ]
[          [number    PL]                  ]
[number    PL                              ]
[definite  yes                             ]
[pattern   [first   1]                     ]
[          [second  2]                     ]
```

(From S.F., 2010)