Feature-Based Grammar

Ling571
Deep Processing Techniques for NLP
February 3, 2016
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

- $S \rightarrow NP[\text{NUM}=?n] \ VP[\text{NUM}=?n]$  
- $NP[\text{NUM}=?n] \rightarrow N[\text{NUM}=?n]$  
- $NP[\text{NUM}=?n] \rightarrow \text{PropN}[\text{NUM}=?n]$  
- $NP[\text{NUM}=?n] \rightarrow \text{Det}[\text{NUM}=?n] \ N[\text{NUM}=?n]$  
- $\text{Det}[\text{NUM}=\text{sg}] \rightarrow \text{'}this' | \text{'}every'$  
- $\text{Det}[\text{NUM}=\text{pl}] \rightarrow \text{'}these' | \text{'}all'$  
- $N[\text{NUM}=\text{sg}] \rightarrow \text{'}dog' | \text{'}girl' | \text{'}car' | \text{'}child'$  
- $N[\text{NUM}=\text{pl}] \rightarrow \text{'}dogs' | \text{'}girls' | \text{'}cars' | \text{'}children'$
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
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.:
  - ?The rocks slept.

- Many proposed:
  - Animacy: +/−
  - Natural gender: masculine, feminine, neuter
  - Human: +/−
  - Adult: +/−
  - Liquid: +/−
Aspect (J&M 17.4.2)

- 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 (in an instant) vs activity (for a time)
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
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**
HW #5
Ling 571
Deep Techniques for NLP
February 3, 2016
Feature-based Parsing

- Goals:
  - Explore the role of features in implementing linguistic constraints.
  - Identify some of the challenges in building compact constraints to define a precise grammar.
  - Gain some further familiarity with NLTK.
  - Apply feature-based grammars to perform grammar checking.

- Individual work
Task

- Create grammar rules with features
  - Produce a single parse for grammatical sentences
    - Single parse per line
  
- Reject ungrammatical sentences
  - Print blank line

- Homework includes sentences and “key”
Feature Grammar in NLTK

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

- Nice discussion, examples in NLTK book CH. 9 (Ch. 8, ed1)
- NOTE: HPSG-style comps list <NP,PP,..> NOT built into NLTK
  - Can be approximated with pseudo-list: e.g. [FIRST=?a, REST=?b]
  - For Extra-credit
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]]”)

- More complex structure
Reentrant Feature Structures

- First instance
  - Parenthesized integer: (1)

- Subsequent instances:
  - ‘Pointer’: -> (1)

```python
>>> 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

  - $N[\text{AGR}=[\text{NUM}='pl']]] \rightarrow 'students'$
  - $N[\text{AGR}=[\text{NUM}='sg']]] \rightarrow 'student'$

- So far, all values are literal or reentrant
  - Variables allow generalization: $?a$
    - Allows underspecification, e.g. $\text{Det}[\text{GEN}=?a]$
    - $\text{NP}[\text{AGR}=?a] \rightarrow \text{Det}[\text{AGR}=?a] N[\text{AGR}=?a]$
Mechanics

- >>> fs3 = nltkFeatStruct(NUM='pl', PER=3)

- >>> fs4 = nltkFeatStruct(NUM='pl')

- >>> print fs4.unify(fs3)

  [NUM = 'pl']
  [PER  =  3  ]