Today's Topics

- Homework 13 out today
- Let's practice writing a natural language grammar...
Homework 13

• Cognate Object Construction (COC) examples:
  • He **fought** a heroic **fight**
  • She **lived** a happy **life**
  • They **died** a gruesome **death**
  • You **smiled** a happy **smile**

• According to Levin's EVCA (1993), CO-taking verbs include:
  • Verbs of Nonverbal Expression (some), e.g. **laugh**, **smile**, **yawn**;
  • Waltz Verbs, e.g. **boogie**, **dance**, **waltz**;
  • Other Verbs: e.g. **dream**, **flight**, **live**, **sing**, **sleep**, **think**;
Homework 13

• Question 1: write a predicate co_verb(V, N) that databases the CO-taking verbs, e.g.
  1. co_verb(fight, fight).
  2. co_verb(live, life).

• Question 2: write a predicate verb_form(V, POS_Tag, Form) that lists the verb forms for the CO-taking verbs, e.g.
  • verb_form(fight, vbd, fought).
  • verb_form(fight, vbz, fights).
Homework 13

• Question 3: use Prolog's grammar rules to write a sentence grammar that allows the intransitive form or take cognate objects only.
  • HINT 0: normally we write lexical rules directly as follows, e.g.
    • dt --> [a].
    • dt --> [the].
    • jj --> [happy].
  • HINT 1: use {...} in conjunction with your answers to Questions 1 and 2, e.g.
    • co_verb --> [Form], {verb_form(_Verb,_Tag,Form)}.
      (_Variable: means don't warn me about this variable not being used here.)
  • HINT 2: use an extra argument to restrict your sentential object to be a CO, e.g.
    • vp --> co_verb(CO_V), np(CO_N), {co_verb(CO_V,CO_N)}. 
Test your grammar, document your code, and submit plenty of examples, e.g.

**Grammatical:**
- `s([she,lived],[]).`
- `s([she,lived,a,life],[]).`
- `s([she,lived,a,happy,life],[]).`

**Ungrammatical:**
- `s([she,lived,a,death],[]).`
Homework 13

• Question 4: modify your answer to Question 3 to compute parses for the grammatical cases, e.g.

• Use the Penn Treebank POS and syntax tags

• HINT: you can have multiple extra arguments in a grammar rule

• Submit your runs
Homework 13

• Usual rules:
  • one PDF file, code attachments, due by next Monday midnight
Phrase Structure Grammars

• Let's write a phrase structure grammar (PSG) using Prolog

• Mechanisms:
  1. Extra argument for Prolog term representation of a parse (*you already know this*)
  2. Extra arguments for feature value agreement
  3. Dealing with left recursive rules: grammar transformation
Prolog Parse

- Let's write a simple grammar returning appropriate parse trees for sentences like:
  - *John kicked the ball*
  - *the men kicked the ball*
  - *a man kicked the ball*

- Stanford Parser:
  \[
  \text{(ROOT} \quad \quad \text{(S}} \\
  \quad \quad \quad \quad \quad \text{(NP} \quad \text{(DT the) (NNS men)}) \\
  \quad \quad \quad \quad \quad \quad \quad \quad \text{(VP} \quad \text{(VBD kicked) }} \\
  \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{(NP} \quad \text{(DT the) (NN ball))})\]

- Berkeley Parser:
  \[
  \text{(ROOT} \quad \quad \text{(S}} \\
  \quad \quad \quad \quad \quad \text{(NP} \quad \text{(DT the) (NNS men)}) \\
  \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{(VP} \quad \text{(VBD kicked) }} \\
  \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{(NP} \quad \text{(DT the) (NN ball))))\]


Prolog Parse

?- s(Parse, [a, men, kicked, the, ball], []).  
   false.

?- s(Parse, [the, men, kicked, the, ball], []).  
   Parse = s(np(dt(the)), nns(men)), vp(vbd(kicked), np(dt(the), nn(ball)))) ;  
   false.

?- s(Parse, [the, man, kicked, the, ball], []).  
   Parse = s(np(dt(the)), nn(man)), vp(vbd(kicked), np(dt(the), nn(ball)))) ;  
   false.

?- s(Parse, [a, man, kicked, the, ball], []).  
   Parse = s(np(dt(a)), nn(man)), vp(vbd(kicked), np(dt(the), nn(ball)))) ;  
   false.

?-
Topics

• Mechanisms:
  1. Extra argument for Prolog term representation of a parse
  2. Extra arguments for feature value agreement
  3. Dealing with left recursive rules: grammar transformation
Phrase Structure Grammars

• Berkeley Parser: *a men kicked the ball*

(Root
 (S
   (NP (DT a) (NNP men))
   (VP (VBD kicked)
     (NP (DT the) (NN ball))))

ROOT
  S
  NP  VP
    DT  NNP  VBD  NP
         a  men  kicked  DT  NN
              the  ball
Extra Arguments: Agreement

- **Idea:**
  - We can also use an extra argument to impose constraints between constituents within a DCG rule

- **Example:**
  - English determiner-noun number agreement
  - Data:
    - the man
    - the men
    - a man
    - *a men
  - Lexical Features:
    - man singular
    - men plural
Extra Arguments: Agreement

• Data:
  • the man/men
  • a man/*a men

• Grammar: (NP section)

```plaintext
np(np(Y)) --> pronoun(Y).
np(np(D,N)) --> det(D,Number), common_noun(N,Number).
det(dt(the),sg) --> [the].
det(dt(the),pl) --> [the].
det(dt(a),sg) --> [a].
common_noun(nn(ball),sg) --> [ball].
common_noun(nn(man),sg) --> [man].
common_noun(nns(men),pl) --> [men].
pronoun(prp(i)) --> [i].
pronoun(prp(we)) --> [we].
```

– Idea:
give determiners a number feature as well and make it agree with the noun

• Rules
  • `the` can combine with singular or plural nouns
  • `a` can combine only with singular nouns
Extra Arguments: Agreement

- **Simplifying the grammar:**
  
  \[
  \begin{align*}
  \text{det}(\text{dt}(\text{the}), \text{sg}) & \rightarrow [\text{the}]. \\
  \text{det}(\text{dt}(\text{the}), \text{pl}) & \rightarrow [\text{the}]. \\
  \text{det}(\text{dt}(\text{a}), \text{sg}) & \rightarrow [\text{a}].
  \end{align*}
  \]

- **Grammar is ambiguous:**
  
  - two rules for determiner *the*
  - Agreement Rule (revisited):
    - *the* can combine with singular or plural nouns
    - i.e. *the* doesn’t care about the number of the noun
  - DCG Rule:
    
    \[
    \begin{align*}
    \text{np}(\text{np}(\text{D}, \text{N})) & \rightarrow \text{det}(\text{D}, \text{Number}), \text{common_noun}(\text{N}, \text{Number}). \\
    \text{det}(\text{dt}(\text{the}), \_ ) & \rightarrow [\text{the}].
    \end{align*}
    \]

**Note:** \_ is a variable

*used underscore character because we don’t care about the value of the variable*
Note:

- Use of the extra argument for agreement here is basically “syntactic sugar” and lends no more expressive power to the grammar rule system.
- i.e. we can enforce the agreement without the use of the extra argument at the cost of more rules.

• Instead of
  \[ \text{np}(\text{np}(D,N)) \rightarrow \text{det}(D,\text{Number}), \]
  \[ \quad \text{common}_\text{noun}(N,\text{Number}). \]

we could have written:

\[ \text{np}(\text{np}(D,N)) \rightarrow \text{detsg}(D), \text{common}_\text{nounsg}(N). \]
\[ \text{np}(\text{np}(D,N)) \rightarrow \text{detpl}(D), \text{common}_\text{nounpl}(N). \]
\[ \text{detsg}(\text{dt}(a)) \rightarrow [a]. \]
\[ \text{detsg}(\text{dt}(\text{the})) \rightarrow [\text{the}]. \]
\[ \text{detpl}(\text{dt}(\text{the})) \rightarrow [\text{the}]. \]
\[ \text{common}_\text{nounsg}(\text{nn}(\text{ball})) \rightarrow [\text{ball}]. \]
\[ \text{common}_\text{nounsg}(\text{nn}(\text{man})) \rightarrow [\text{man}]. \]
\[ \text{common}_\text{nounpl}(\text{nn}(\text{men})) \rightarrow [\text{men}]. \]
Extra Arguments: Agreement

• English exhibits subject-verb agreement

• Examples:
  1. John kicked the ball
  2. The men kicked the ball
  3. John kicks the balls
  4. The men *kicks/kick the ball

Constraint:
  1. -s form of the verb is compatible with 3rd person singular only for the subject NP
  2. uninflected form is not compatible with 3rd person singular for the subject NP
Subject Verb Agreement

• We need feature percolation:

Subject and VP come together at this rule

Form Ending  |  Comment
-------------|------------------
eat          | uninflected

| eat          | -s               | 3rd person singular |
eaten        | -en              | past participle     |
eating       | -ing             | gerund              |

POS tags

VB - Verb, base form
VBD - Verb, past tense
VBG - Verb, gerund or present participle
VBN - Verb, past participle
VBP - Verb, non-3rd person singular present
VBZ - Verb, 3rd person singular present
Subject Verb Agreement

• Implementation: using POS tags
  \[ v(vb(eat),vb) \rightarrow [eat]. \]
  \[ v(vbd(ate),vbd) \rightarrow [ate]. \]
  \[ v(vbg(eating),vbg) \rightarrow [eating]. \]
  \[ v(vbn(eaten),vbn) \rightarrow [eaten]. \]
  \[ v(vbp(eat),vbp) \rightarrow [eat]. \]
  \[ v(vbz(eats),vbz) \rightarrow [eats]. \]

• Constraint table:
  - % table of Person Number Tag possible combinations
  - table(3,plural,vb).
  - table(3,plural,vbd).
  - table(3,singular,vbz).
  - table(3,singular,vbd).
Topics

• Mechanisms:
  1. Extra argument for Prolog term representation of a parse
  2. Extra arguments for feature value agreement
  3. **Dealing with left recursive rules: grammar transformation**
Left recursion and Prolog

Left recursive grammars:

• we know from an earlier lecture that left recursive rules are a no-no given Prolog’s left-to-right depth-first computation rule...

• Example:
  1. \( s \rightarrow a, [!] \).
  2. \( a \rightarrow ba, [a] \).
  3. \( a \rightarrow a, [a] \).
  4. \( ba \rightarrow b, [a] \).
  5. \( b \rightarrow [b] \).

?- s([b,a,!],[[]]).
ERROR: Out of local stack
Preposition Phrase (PP) Attachment

• The preferred syntactic analysis is a left recursive parse
• Examples:
  • *John saw the boy with a telescope*
  • (structural ambiguity: automatically handled by Prolog)