Question 1.4
Nirupama Sharma

Common errors in Stanford Parser and Stanford Dependency Parser outputs:

1. **"that" is often considered as preposition in constituent parsing and dependency parsing using Stanford Parser while it is a demonstrative determiner**.

```
(ROOT
 (S
  (S
   (NP (PRP I))
   (VP (MD would)
    (VP (VB ask)
     (NP (PRP you))
     (, ,)
     (SBAR
      (ADVP (RB now))
      (IN that)
      (S
       (NP (PRP you))
       (VP (VBP are)
        (NP
         (NP (NNP President))
         (PP (IN of)
          (NP (DT the) (NNP Council))))))))))
```

2. **Homonymy causes errors in Stanford Parser**

Following is the example, where the Stanford POS tagger failed to identify right POS tags and the error was propagated in the later stages of the syntax analysis performed by the Stanford parser as parser uses tagger:

Example:

A customer books two items.

Constituent Parse Tree:

```
(ROOT
  (NP
    (NP (DT A) (NN customer) (NNS books))
    (NP (CD two) (NNS items))
    (. .)))
```

Typed Dependencies:

```
det(books-3, A-1)
nn(books-3, customer-2)
num(items-5, two-4)
dep(books-3, items-5)
```

token ‘books’ is misjudged as ‘NNS’ by the Stanford parser.

The token ‘books’ is verb and the correct POS tag is ‘VBZ’. Parse tree is also wrong as VP is missing. Similarly, the typed dependencies are also wrongly judged det(books-3, A-1) should be det(customer-2, A-1), nn(books-3, customer-2) should be nsubj(books-3, customer-2),and dep(books-3, items-5) should be nobj(books-3, item-5)
.
Another example of homonymy is “A customer can bank on manager”. In this example, word ‘bank’ is wrongly POS tagged ‘NN’ but the correct POS tag is ‘VB’.

3. **Attachment Ambiguities/Structural ambiguities**

English: The pay is given to all employees with bonus.

CFG Parse Tree:

```
(ROOT

 (S

  (NP (DT The) (NN pay))

   (VP (VBZ is)

     (VP (VBN given)

       (PP (TO to)

         (NP

             (NP (DT all) (NNS employees))

             (PP (IN with)
         (NP (NN bonus)))))))
 (. .)))
```

Typed Dependency:

det(pay-2, The-1)

nsubjpass(given-4, pay-2)

auxpass(given-4, is-3)

det(employees-7, all-6)

prep_to(given-4, employees-7)

prep_with(employees-7, bonus-9)

In this example, it is shown that the typed dependencies generated by the Stanford parser are wrong such as prep_with(employees-7, bonus-9). However, the correct typed dependency for this example should be prep_with(pay-2, bonus-9) to represent the actual meaning of the sentence -the pay with bonus is given to all the employees.

NP PP attachment ambiguities:

He was named chief executive officer of Applied in 1986.

Incorrect PP attachment:

```
(ROOT
  (S (NP (PRP He))
    (VP (VBD was)
      (VP (VBN named)
        (NP (NP (JJ chief)
            (NN executive)
            (NN officer))
          (PP (IN of)
            (NP (NP (NNP Applied))
              (PP (IN in)
                (NP
                    (CD 1986))))))))
   (. .)))
```

Correct PP Attachment

```
(ROOT
  (S (NP (PRP He))
    (VP (VBD was)
      (VP (VBN named)
        (NP (NP (JJ chief)
              (NN executive)
              (NN officer))
          (PP (IN of)
            (NP (NNP Applied))))
        (PP (IN in)
          (NP (CD 1986)))))
    (. .)))
```

NP Attachment Ambiguity:

Fidelity Investments placed new ads in newspapers yesterday, and wrote another new ad appearing today.

Incorrect NP attachment.

```
(ROOT
  (S (NP (NNP Fidelity)
        (NNPS Investments))
    (VP
      (VP (VBD placed)
           (NP (JJ new)
               (NNS ads))
           (PP (IN in)
               (NP (NNS newspapers)))
           (NP (NN yesterday)))
       (, ,)
       (CC and)
       (VP (VBD wrote)
           (NP (NP (DT another)
                   (JJ new)
                   (NN ad))
               (VBG appearing))
           (NP (NN today))))
     (. .)))
```

Correct NP Attachment:

```
(ROOT
  (S (NP (NNP Fidelity)
         (NNPS Investments))
     (VP
       (VP (VBD placed)
           (NP (JJ new)
               (NNS ads))
           (PP (IN in)
               (NP (NNS newspapers)))
           (NP (NN yesterday)))
       (, ,)
```

```
    (CC and)

    (VP (VBD wrote)

      (NP (NP (DT another)

          (JJ new)

          (NN ad))

        (VP (VBG appearing)

          (NP (NN today))))))

   (. .)))
```

There are a few ways to resolve these issues:

Methods utilizing probabilities for parse trees are better known to resolve ambiguity issues while parsing-whether it is attachment ambiguity or Homonymy.

1. **Probabilistic Context free Grammars PCFGs**

   Citation: As given in lecture slides-definition

- G = (T, N, S, R, P)
- T is a set of terminal symbols
- N is a set of nonterminal symbols
- S is the start symbol (S ∈ N)
- R is a set of rules/productions of the form X ® g
- P is a probability function
- P: R -> [0,1]

PCFGs give probabilities to each kind of attachment whether its NP, VP or PP. A grammar G generates a language model L.

For Example: Here the probabilities of two parse trees for the below sentence are calculated. The parse tree with the higher probability is considered. It removes homonymy related parse errors and attachment ambiguities.

Calvin imagined monsters in school

(S (NP Calvin)

```
  (VP (V imagined)
      (NP (NP monsters)
          (PP (P in)
              (NP school)))))
```

P(tree1) = P(S → NP VP) × P(NP → Calvin) × P(VP → V NP) × P(V → imagined) × P(NP → NP PP) × P(NP → monsters) × P(PP → P NP) × P(P → in) × P(NP → school) = 1 × 0.25 × 0.9 × 1 × 0.25 × 0.25 × 1 × 1 × 0.25 = .003515625

```
(S (NP Calvin)
 (VP (VP (V imagined)
        (NP monsters))
    (PP (P in)
        (NP school))))
```

P(tree2) = P(S → NP VP) × P(NP → Calvin) × P(VP → VP PP) × P(VP → V NP) × P(V → imagined) × P(NP → monsters) × P(PP → P NP) × P(P → in) × P(NP → school) = 1 × 0.25 × 0.1 × 0.9 × 1 × 0.25 × 1 × 1 × 0.25 = .00140625.

Most probable parse here is tree1 as P(tree1)>P(tree2). Hence the ambiguity is resolved.

2. **Probabilistic Cocke-Kasami-Younger (CKY) Constituency Parsing and Extended CKY Parsing-**It employs bottom up parsing and dynamic programming. Like PCFG it helps to get the most probable parse based on the probabilities. One example of CKY parsing is given in Question 2 PDF-attached in q2 folder.

3. **Lexicalized Parsing->Lexicalization models argument selection by sharpening rule expansion probabilities.**

Part of Speech ambiguity: saw → noun saw → verb

Structural ambiguity: Prepositional Phrases

I saw (the man) with the telescope

I saw (the man with the telescope)

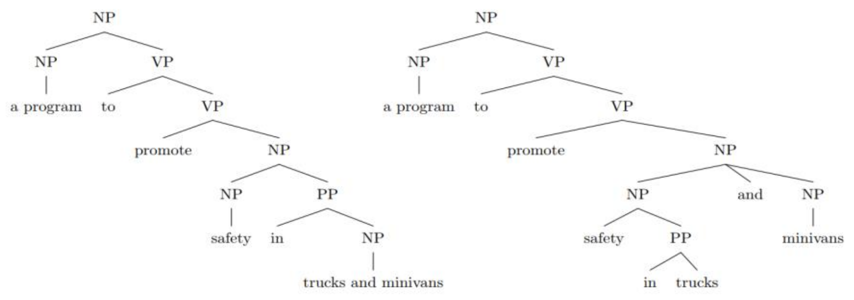
Structural ambiguity: Coordination

a program to promote safety in ((trucks) and (minivans))

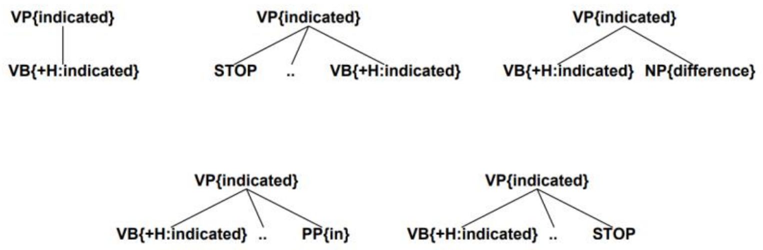
a program to promote ((safety in trucks) and (minivans))

((a program to promote safety in trucks) and (minivans))

Attachment choice in alternative parses:



Adding lexical information to PCFG:



Ph(VB | VP, indicated) × Pl(STOP | VP, VB, indicated)× Pr(NP(difference) | VP, VB, indicated)× Pr(PP(in) | VP, VB, indicated)× Pr(STOP | VP, VB, indicated)

To do better, it is necessary to condition probabilities on the actual words of the sentence. This makes the probabilities much tighter.

Evaluation: Consider a candidate parse to be evaluated against the truth (or gold-standard parse):

candidate: (S (A (P this) (Q is)) (A (R a) (T test))) gold: (S (A (P this)) (B (Q is) (A (R a) (T test))))

For evaluation, list all the constituents for candidate and gold.

Candidate	Gold
(0,4,S)	(0,4,S)
(0,2,A)	(0,1,A)
(2,4,A)	(1,4,B)
	(2,4,A)

Skip spans of length 1 which would be equivalent to part of speech tagging accuracy. Precision is defined as #correct/#proposed = 2/3 and recall as #correct/#in gold = 2/4. The parse with the better F1 score is the most probable parse.