CS-626 Assignment 3
Parsing

Shubham Nemani, 203050011,
Anish M M, 203050066,
Pooja Verma, 203050072

November 17, 2020

1 CP to DP

1.1 Methodology Used

1. CP tree for a sentence is obtained using Berkeley Neural Parser , output is a string, converted to list of tokens.
2. Input to parser is list of tokens - (,) , tags , words for CP to DP conversion.
3. We are using a shift reduce parser to parse CP string into tree using stack.
4. It involves two steps:
   - Rule[1] based approach is used to find head of phrase using transforming dependencies into phrase structures [2]
   - Assigning dependency labels between head and modifier [3]

1.2 Error Analysis

1. In some sentences adjectives are tagged as NN by parser in original CP output , so wrong head chosen by rules leading to incorrect labeling.

   Sentence- Students played street football.

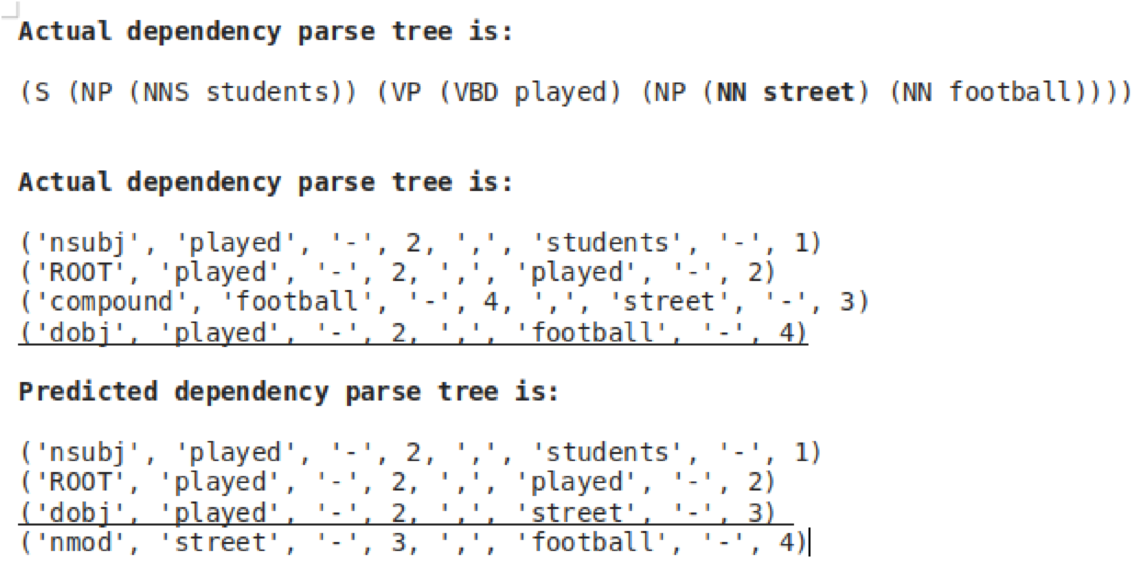
   CP Tree-(S (NP (NNS Students)) (VP (VBD Played) (NP (NN street) (NN Football))))

   Here , street is tagged as NN , so for phrase NP , street is selected as Head.
2. Sentences containing punctuations give very low accuracy , need to handle them.
3. Dependency relation labeling accuracy is very low for CASE , CCOMP , XCOMP.

[1] https://www.aclweb.org/anthology/W07-2416.pdf

[2] http://people.seas.harvard.edu/ srush/naacl15.pdf

[3] https://www.researchgate.net/publication/324940566_Guidelines_for_the_CLEAR_Style_Constituent_to_Dependency_



```
Actual dependency parse tree is:

(S (NP (NNS students)) (VP (VBD played) (NP (NN street) (NN football))))


Actual dependency parse tree is:

('nsubj', 'played', '-', 2, ',', 'students', '-', 1)
('ROOT', 'played', '-', 2, ',', 'played', '-', 2)
('compound', 'football', '-', 4, ',', 'street', '-', 3)
('dobj', 'played', '-', 2, ',', 'football', '-', 4)

Predicted dependency parse tree is:

('nsubj', 'played', '-', 2, ',', 'students', '-', 1)
('ROOT', 'played', '-', 2, ',', 'played', '-', 2)
('dobj', 'played', '-', 2, ',', 'street', '-', 3)
('nmod', 'street', '-', 3, ',', 'football', '-', 4)
```

1.3 Evaluation Metrics

We are using two types of evaluation metrics.

1. **Head Accuracy**: The first part is finding head for a given phrase.

2. **Label Accuracy**: Second part is labeling of arc between head and dependent correctly.

2 DP to CP

2.0.1 Methodology

- We follow a rule based bottom-up approach for this conversion.

- We use StanfordNLP parser to get the input dependency parse.

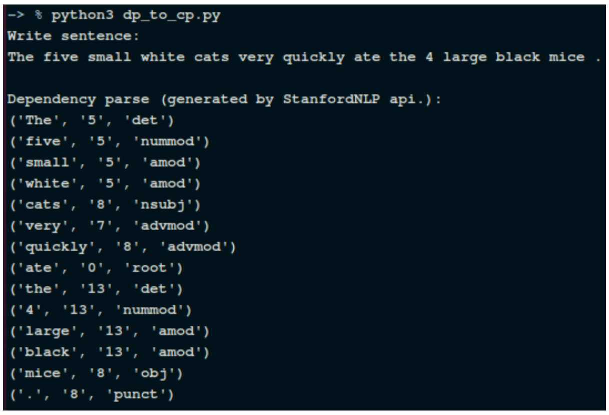
- We use the universal tagset for POS tags in the output CP.

- Our tool handles sentences with a verb, subject, object, one or more adjectives, one or more adverbs, numeral modifier for nouns, determiners.

2.1 Sample Input

Consider the sample input sentence **The five small white cats very quickly ate the 4 large black mice .**

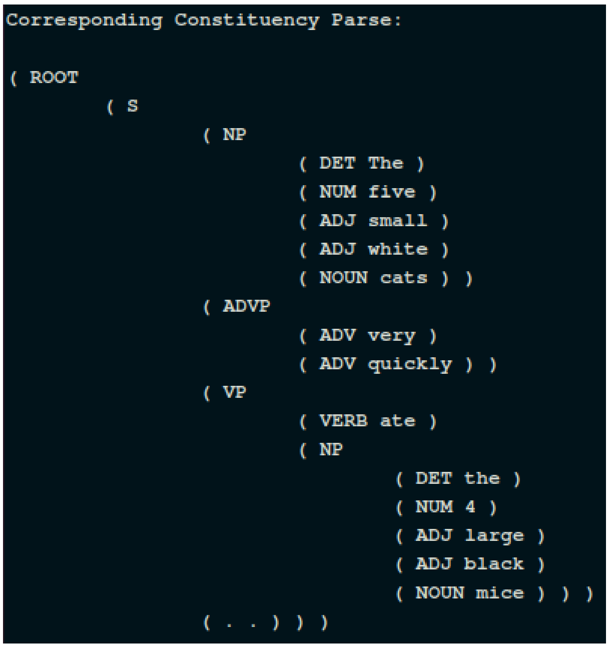
Figure 1 shows the input dependency parse that is created using the StanfordNLP parser. Figure 2 shows the output constituency parse our tool creates for that dependency parse.



```
-> % python3 dp_to_cp.py
Write sentence:
The five small white cats very quickly ate the 4 large black mice .

Dependency parse (generated by StanfordNLP api.):
('The', '5', 'det')
('five', '5', 'nummod')
('small', '5', 'amod')
('white', '5', 'amod')
('cats', '8', 'nsubj')
('very', '7', 'advmod')
('quickly', '8', 'advmod')
('ate', '0', 'root')
('the', '13', 'det')
('4', '13', 'nummod')
('large', '13', 'amod')
('black', '13', 'amod')
('mice', '8', 'obj')
('.', '8', 'punct')
```

Figure 1: Dependency parse generated by StanfordNLP parser (input)



```
Corresponding Constituency Parse:

( ROOT
        ( S
                ( NP
                        ( DET The )
                        ( NUM five )
                        ( ADJ small )
                        ( ADJ white )
                        ( NOUN cats ) )
                ( ADVP
                        ( ADV very )
                        ( ADV quickly ) )
                ( VP
                        ( VERB ate )
                        ( NP
                                ( DET the )
                                ( NUM 4 )
                                ( ADJ large )
                                ( ADJ black )
                                ( NOUN mice ) ) )
                ( . . ) ) )
```

Figure 2: Constituency parse generated by our code (output)

2.2 Analysis

- Methodology specifics:
  - Dependency parse does not distinguish between Nouns and Pronouns. But for constituency parse, we need this distinction. For this, we make use of the fact that pronouns are a closed set, and maintain a list of pronouns. We use this to determine if the given word tagged as *nsubj* or *obj* in the DP is a Noun or a Pronoun.
  - We follow a bottom up approach and link words to their heads in the dependecy relations, following some specific order to ensure that the final constituency parse is correct.
  - When handling sentences with multiple consecutive adjectives or consecutive adverbs, we parse the sentence in reverse order to ensure that in the final parse, the words follow the correct order.
- Shortcomings:
  - The Dependency parse doesn't distinguish between proper and improper nouns. So we are not able to distinguish them either, and tag both as Nouns.
  - We support only a subset of sentence types as mentioned in subsection 2.0.1.
  - The StanfordNLP parser fails to produce parse for some sentences. In those cases, we report the same.
  - Sometimes, the StanfordNLP parser keeps running without returning anything, and we have to manually kill the process. In that case too, our tool will fail.