Batch: 1 **Roll No.: 16010420008** **Experiment No.:7**

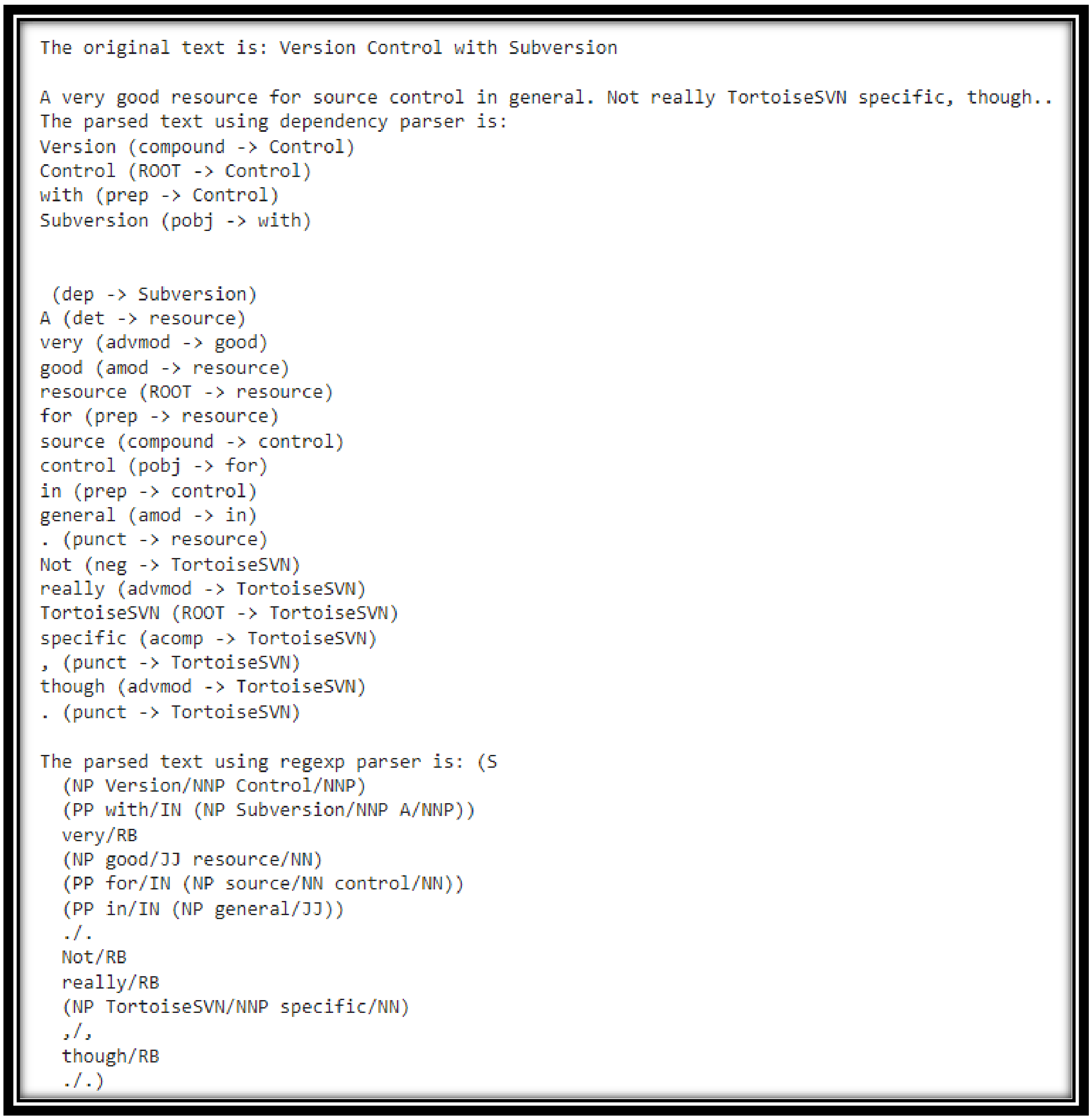
Aim: To implement any two Parsers

Resources needed: Text Editor, Python Interpreter

Code:

```
import pandas as pd
import re
import nltk
import spacy
spacy_model = spacy.load("en_core_web_sm")
from nltk.chunk.regexp import RegexpParser
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

csv_file = pd.read_csv('/content/Answers.csv')
csv_file_column = csv_file[csv_file['Body'].notna()]

answer_column = []
for i in range(0, 1):
    answer_column.append(csv_file_column['Body'][i])

def html_removal(text):
    pattern = r"<[^>]+>"
    html_removal_text = re.sub(pattern, "", text)
    return html_removal_text

def dependency_parser(text):
    parsed_text = spacy_model(text)
    return parsed_text

def regexp_parser(text):
    grammar = r"""
      NP: {<DT|JJ|NN.*>+}
      PP: {<IN><NP>}
      VP: {<VB.*><NP|PP|CLAUSE>+$}
      CLAUSE: {<NP><VP>}
    """

    tokens = nltk.word_tokenize(text)
    tagged = nltk.pos_tag(tokens)

    parser = RegexpParser(grammar)

    parsed_text = parser.parse(tagged)
    parsed_text = parsed_text.__str__()
    return parsed_text

for i in answer_column:
    html_removal_text = html_removal(i)
    print(f"The original text is: {html_removal_text}.")
    dependency_parser = dependency_parser(html_removal_text)
    print(f"The parsed text using dependency parser is:  ")
    for token in dependency_parser:
        print(f"{token.text} ({token.dep_} -> {token.head.text})")
    regexp_parser = regexp_parser(html_removal_text)
    print(f"\nThe parsed text using regexp parser is: {regexp_parser}")
```

Output:



```
The original text is: Version Control with Subversion

A very good resource for source control in general. Not really TortoiseSVN specific, though..
The parsed text using dependency parser is:
Version (compound -> Control)
Control (ROOT -> Control)
with (prep -> Control)
Subversion (pobj -> with)


 (dep -> Subversion)
A (det -> resource)
very (advmod -> good)
good (amod -> resource)
resource (ROOT -> resource)
for (prep -> resource)
source (compound -> control)
control (pobj -> for)
in (prep -> control)
general (amod -> in)
. (punct -> resource)
Not (neg -> TortoiseSVN)
really (advmod -> TortoiseSVN)
TortoiseSVN (ROOT -> TortoiseSVN)
specific (acomp -> TortoiseSVN)
, (punct -> TortoiseSVN)
though (advmod -> TortoiseSVN)
. (punct -> TortoiseSVN)

The parsed text using regexp parser is: (S
  (NP Version/NNP Control/NNP)
  (PP with/IN (NP Subversion/NNP A/NNP))
  very/RB
  (NP good/JJ resource/NN)
  (PP for/IN (NP source/NN control/NN))
  (PP in/IN (NP general/JJ))
  ./.
  Not/RB
  really/RB
  (NP TortoiseSVN/NNP specific/NN)
  ,/,
  though/RB
  ./.)
```

Observations:

Dependency Parser:

- The dependency parser creates a tree structure that represents the grammatical relationships between words in the text.
- It identifies the following key relationships:
    - "Version" is a compound of "Control," indicating a compound noun relationship.
    - "Control" is the root of the sentence, meaning it is the main subject.
    - "with" is a prepositional word that relates to "Control."
    - "Subversion" is the object of the preposition "with."
    - "A" is a determiner related to "resource."
    - "very" is an adverb modifying "good."
    - "good" is an adjective describing "resource."
    - "resource" is the root of the sentence, acting as the main subject.

- "for" is a preposition that relates to "resource."
- "source" is a compound of "control," indicating a compound noun relationship.
- "control" is the object of the preposition "for."
- "in" is a preposition related to "control."
- "general" is an adjective describing "in."
- "Not" is a negation word related to "TortoiseSVN."
- "really" is an adverb modifying "TortoiseSVN."
- "TortoiseSVN" is the root of the sentence, acting as the main subject.
- "specific" is an adjective describing "TortoiseSVN."
- "though" is an adverb related to "TortoiseSVN."

Regexp Parser:

- The regexp parser uses regular expressions to define patterns for identifying phrases in the text.
- It structures the text into phrases and subphrases based on the defined patterns:
    - "(S" represents the main sentence.
    - "(NP Version/NNP Control/NNP)" identifies a noun phrase with "Version Control."
    - "(PP with/IN (NP Subversion/NNP A/NNP))" identifies a prepositional phrase with "Subversion A."
    - "very/RB" represents an adverb.
    - "(NP good/JJ resource/NN)" identifies a noun phrase with "good resource."
    - "(PP for/IN (NP source/NN control/NN))" identifies a prepositional phrase with "source control."
    - "(PP in/IN (NP general/JJ))" identifies a prepositional phrase with "general."
    - "./." represents a period.
    - "Not/RB" represents an adverb.
    - "really/RB" represents an adverb.
    - "(NP TortoiseSVN/NNP specific/NN)" identifies a noun phrase with "TortoiseSVN specific."
    - ",/," represents a comma.
    - "though/RB" represents an adverb.
    - "./.)" represents a closing parenthesis.

Questions:

1. Differentiate between deep and shallow parser.

A) Deep parsing and shallow parsing are two different approaches to syntactic parsing in natural language processing (NLP) and computational linguistics. They differ in terms of the level of detail and linguistic information they extract from a sentence.

a) Deep Parsing:

- **Granularity:** Deep parsing aims to provide a detailed analysis of a sentence's grammatical structure and the relationships between words. It goes beyond just identifying basic sentence components (e.g., words, phrases) and seeks to understand the hierarchical and functional relationships between them.
- **Linguistic Knowledge:** Deep parsing typically relies on a comprehensive understanding of linguistic rules and grammar. It considers aspects such as phrase structure, grammatical functions (subject, object, etc.), and syntactic roles.

- **Complexity:** Deep parsing can be computationally expensive and complex. It often involves extensive parsing algorithms that require a deep understanding of the language's grammar, which may vary from language to language.
- **Use Cases:** Deep parsing is valuable in applications where a detailed, fine-grained analysis of text is required, such as in machine translation, sentiment analysis, and question answering systems.

b) Shallow Parsing (Chunking):

Granularity: Shallow parsing, also known as chunking, provides a less detailed analysis of a sentence. Instead of breaking down the sentence into fine-grained grammatical components, it groups words into larger chunks or phrases, such as noun phrases or verb phrases.
Linguistic Knowledge: Shallow parsing is less concerned with the intricate details of grammar and focuses more on identifying meaningful units within the text. It does not attempt to establish the complete syntactic structure of the sentence.
Simplicity: Shallow parsing is computationally less intensive and easier to implement compared to deep parsing. It can be used for quick and efficient processing of large volumes of text.
Use Cases: Shallow parsing is often used in applications where a high-level understanding of text is sufficient. For instance, it is employed in information extraction, part-of-speech tagging, and named entity recognition.

Outcomes

CO3: Establish concept of Structure and Semantics

Conclusion

We understood the concept of parsing and the ways of performing parsing. We also implemented two types of parses on our corpus of data and observed the differences in them.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

- Allen.James, Natural Language Understanding, Benjamin Cumming, Second Edition, 1995.
- Jurafsky, Dan and Martin, James, Speech and Language Processing, Prentice Hall, 2008.
- Palash Goyal, Karan Jain, Sumit Pandey,Deep Learning for Natural Language Processing: Creating Neural Networks with Python, Apress, 2018.