Experiment 9

NAME	**Shreya Shetty**
UID	**2019140059**
CLASS	**TE IT**
BATCH	**B**
SUBJECT	**NLP Lab**

AIM: To build and perform neural model for parsing

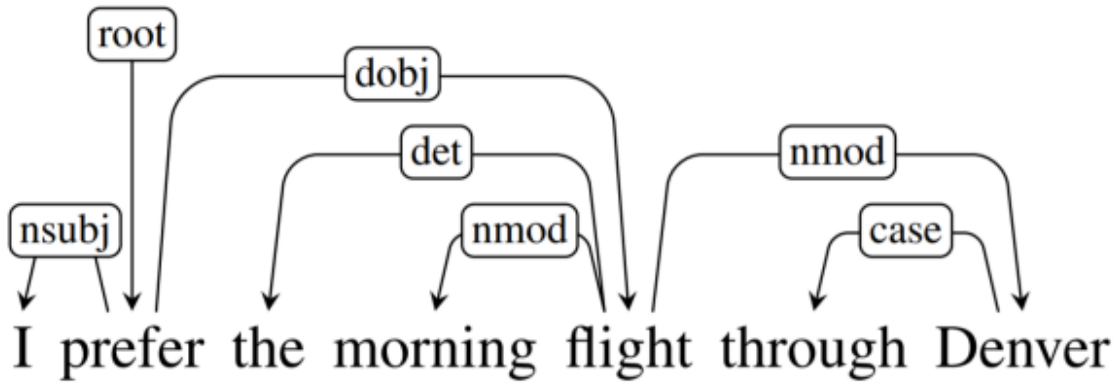
THEORY:

Dependency Parsing :

The term Dependency Parsing (DP) refers to the process of examining the dependencies between the phrases of a sentence in order to determine its grammatical structure. A sentence is divided into many sections based mostly on this. The process is based on the assumption that there is a direct relationship between each linguistic unit in a sentence. These hyperlinks are called dependencies.

Consider the following statement: "I prefer the morning flight through Denver."

The diagram below explains the sentence's dependence structure:



In a written dependency structure, the relationships between each linguistic unit, or phrase, in the sentence are expressed by directed arcs. The root of the tree "prefer" varies the pinnacle of the preceding sentence, as labeled within the illustration.

A dependence tag indicates the relationship between two phrases. For example, the word "flight" changes the meaning of the noun "Denver." As a result, you may identify a dependence from

flight -> Denver, where flight is the pinnacle and Denver is the kid or dependent. It's represented by nmod, which stands for the nominal modifier.

This distinguishes the scenario for dependency between the two phrases, where one serves as the pinnacle and the other as the dependent. Currently, the Common Dependency V2 taxonomy consists of 37 common syntactic relationships, as shown in the table below:

Dependency Tag	Description
acl	clausal modifier of a noun (adnominal clause)
acl:relcl	relative clause modifier
advcl	adverbial clause modifier
advmod	adverbial modifier
advmod:emph	emphasizing phrase, intensifier
advmod:lmod	locative adverbial modifier
amod	adjectival modifier
appos	appositional modifier
aux	auxiliary
aux:move	passive auxiliary
case	case-marking
cc	coordinating conjunction
cc:preconj	preconjunct
ccomp	clausal complement
clf	classifier
compound	compound
compound:lvc	gentle verb building
compound:prt	phrasal verb particle
compound:redup	reduplicated compounds

Dependency Parsing using NLTK :

The Pure Language Toolkit (NLTK) package deal will be used for Dependency Parsing, which is a set of libraries and codes used during statistical Pure Language Processing (NLP) of human language.

We may use NLTK to do dependency parsing in one of several ways:

1. Probabilistic, projective dependency parser: These parsers predict new sentences by using human language data acquired from hand-parsed sentences. They're known to make mistakes and work with a limited collection of coaching information.

2. Stanford parser: It is a Java-based pure language parser. You would want the Stanford CoreNLP parser to perform dependency parsing. The parser supports a number of languages, including English, Chinese, German, and Arabic

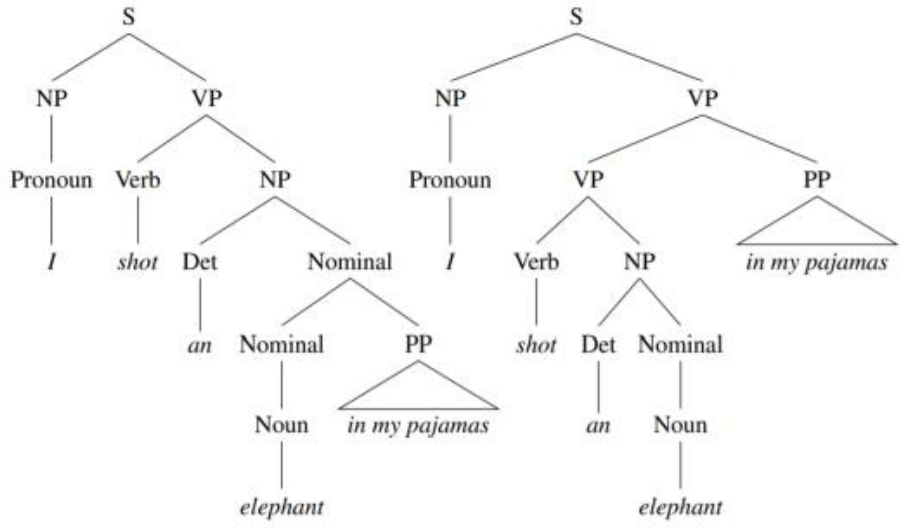
```
from nltk.parse.stanford import StanfordDependencyParser
path_jar = 'path_to/stanford-parser-full-2014-08-27/stanford-parser.jar'
path_models_jar = 'path_to/stanford-parser-full-2014-08-27/stanford-parser-3.4.1-models.jar'
dep_parser = StanfordDependencyParser(
   path_to_jar = path_jar, path_to_models_jar = path_models_jar
)
consequence = dep_parser.raw_parse('I shot an elephant in my sleep')
dependency = consequence.subsequent()
checklist(dependency.triples())
The following is the output of the above program:
[
   ((u'shot', u'VBD'), u'nsubj', (u'I', u'PRP')),
   ((u'shot', u'VBD'), u'dobj', (u'elephant', u'NN')),
   ((u'elephant', u'NN'), u'det', (u'an', u'DT')),
   ((u'shot', u'VBD'), u'prep', (u'in', u'IN')),
   ((u'in', u'IN'), u'pobj', (u'sleep', u'NN')),
   ((u'sleep', u'NN'), u'poss', (u'my', u'PRP$'))
]
```

Constituency Parsing :

Constituency Parsing is based on context-free grammars. Constituency Context-free grammars are used to parse text. Right here the parse tree includes sentences that have been broken down into sub-phrases, each of which belongs to a different grammar class. A terminal node is a linguistic unit or phrase that has a mother or father node and a part-of-speech tag.

For example, "A cat" and "a box beneath the bed", are noun phrases, while "write a letter" and "drive a car" are verb phrases.

Consider the following example sentence: "I shot an elephant in my pajamas." The constituency parse tree is shown graphically as follows:



The parse tree on the left represents catching an elephant carrying pyjamas, while the parse tree on the right represents capturing an elephant in his pyjamas.

The entire sentence is broken down into sub-phases till we've got terminal phrases remaining. VP stands for verb phrases, whereas NP stands for noun phrases.

Input: Text corpus containing multiple sentences.

Output: Parsed structure of every sentence.

IDE USED: Jupyter Notebook

LIBRARIES USED:

Nltk:

The Natural Language Toolkit (NLTK) is a Python package for natural language processing. NLTK requires Python 3.7, 3.8, 3.9 or 3.10. It can be installed as :

pip install nltk

Stanford Parser:

It is a Java-based pure language parser. The Stanford CoreNLP is used parser to perform dependency parsing. The parser supports a number of languages, including English, Chinese, German, and Arabic.

Spacy:

spaCy is a free, open-source library for NLP in Python. It's written in Python and Cython and is designed to build information extraction or natural language understanding systems. It's built for production use and provides a concise and user-friendly API.

PrettyTable:

A simple Python library for easily displaying tabular data in a visually appealing ASCII table format. It can be installed as :

pip install prettytable

CODE & OUTPUT:

Using Stanford parser

```
[1] from nltk.parse.stanford import StanfordDependencyParser
    from google.colab import drive
    drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[2] path_jar = '/content/drive/MyDrive/content/stanford-parser-3.5.2.jar'
    path_models_jar = '/content/drive/MyDrive/content/stanford-parser-3.5.2-models.jar'
    dep_parser = StanfordDependencyParser(
       path_to_jar = path_jar, path_to_models_jar = path_models_jar
    )
```

```
[3] result1 = dep_parser.raw_parse('I prefer the morning flight through Denver')
    dependency = result1.__next__()
    list(dependency.triples())
```

```
[(('prefer', 'VBP'), 'nsubj', ('I', 'PRP')),
 (('prefer', 'VBP'), 'dobj', ('flight', 'NN')),
 (('flight', 'NN'), 'det', ('the', 'DT')),
 (('flight', 'NN'), 'compound', ('morning', 'NN')),
 (('prefer', 'VBP'), 'nmod', ('Denver', 'NNP')),
 (('Denver', 'NNP'), 'case', ('through', 'IN'))]
```

```
[4] result2 = dep_parser.raw_parse('I shot an elephant in my sleep')
    dependency = result2.__next__()
    list(dependency.triples())
```

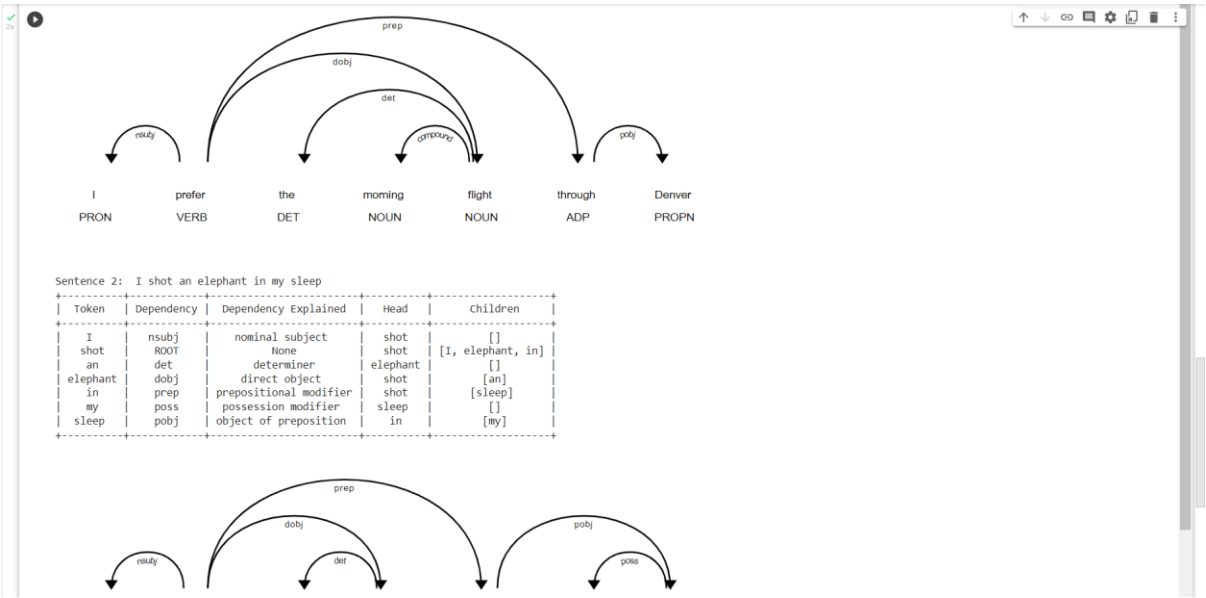
```
[(('shot', 'VBD'), 'nsubj', ('I', 'PRP')),
 (('shot', 'VBD'), 'dobj', ('elephant', 'NN')),
 (('elephant', 'NN'), 'det', ('an', 'DT')),
 (('shot', 'VBD'), 'nmod', ('sleep', 'NN')),
 (('sleep', 'NN'), 'case', ('in', 'IN')),
 (('sleep', 'NN'), 'nmod:poss', ('my', 'PRP$'))]
```

Using SPACY

```
[5] # Importing libraries
    import spacy
    from spacy import displacy
    from prettytable import PrettyTable

    nlp = spacy.load("en_core_web_sm") # Loading the language model

    # Sentence 1
    sentence1 = 'I prefer the morning flight through Denver'
    print("Sentence 1: ",sentence1)
    final_result = PrettyTable(['Token','Dependency','Dependency Explained','Head', 'Children'])
    # nlp returns an object with individual token information, linguistic features and relationships
    # Printing the token, dependency nature, head and all dependents of the token
    for token in nlp(sentence1):
      ch = str([child for child in token.children])
      final_result.add_row([str(token.text), str(token.dep_), str(spacy.explain(token.dep_)),str(token.head.text), ch])
    print(final_result)
    # Visualize the dependency
    displacy.render(nlp(sentence1), style='dep', jupyter=True, options={'distance': 120})

    # Sentence 2
    sentence2 = 'I shot an elephant in my sleep'
    print("Sentence 2: ",sentence2)
    final_result = PrettyTable(['Token','Dependency','Dependency Explained','Head', 'Children'])
    # nlp returns an object with individual token information, linguistic features and relationships
    # Printing the token, dependency nature, head and all dependents of the token
    for token in nlp(sentence2):
      ch = str([child for child in token.children])
      final_result.add_row([str(token.text), str(token.dep_), str(spacy.explain(token.dep_)),str(token.head.text), ch])
    print(final_result)
    # Visualize the dependency
    displacy.render(nlp(sentence2), style='dep', jupyter=True, options={'distance': 120})
```

```
Sentence 1:  I prefer the morning flight through Denver
+---------+------------+------------------------+---------+----------------------+
|  Token  | Dependency |  Dependency Explained  |   Head  |       Children       |
+---------+------------+------------------------+---------+----------------------+
|    I    |   nsubj    |    nominal subject     |  prefer |          []          |
|  prefer |    ROOT    |          None          |  prefer | [I, flight, through] |
|   the   |    det     |       determiner       |  flight |          []          |
| morning |  compound  |        compound        |  flight |          []          |
|  flight |    dobj    |     direct object      |  prefer |    [the, morning]    |
| through |    prep    | prepositional modifier |  prefer |       [Denver]       |
|  Denver |    pobj    | object of preposition  | through |          []          |
+---------+------------+------------------------+---------+----------------------+
```

```
Sentence 2:  I shot an elephant in my sleep
+---------+------------+------------------------+---------+----------------------+
|  Token  | Dependency |  Dependency Explained  |   Head  |       Children       |
+---------+------------+------------------------+---------+----------------------+
|    I    |   nsubj    |    nominal subject     |   shot  |          []          |
|   shot  |    ROOT    |          None          |   shot  | [I, elephant, sleep] |
|    an   |    det     |       determiner       | elephant |          []          |
| elephant |   dobj     |     direct object      |   shot  |        [an]          |
|    in   |    prep    | prepositional modifier |   shot  |        [sleep]       |
|    my   |  poss      |        possessive      |  sleep  |          []          |
|  sleep  |    pobj    | object of preposition  |    in   |        [my]          |
+---------+------------+------------------------+---------+----------------------+
```



Sentence 2: I shot an elephant in my sleep

Token	Dependency	Dependency Explained	Head	Children
I	nsubj	nominal subject	shot	[]
shot	ROOT	None	shot	[I, elephant, in]
an	det	determiner	elephant	[]
elephant	dobj	direct object	shot	[an]
in	prep	prepositional modifier	shot	[sleep]
my	poss	possession modifier	sleep	[]
sleep	pobj	object of preposition	in	[my]

REFERENCES:

1. https://www.analyticsvidhya.com/blog/2021/12/dependency-parsing-in-natural-language processing-with-examples/#h2_2
2. https://medium.com/mlearning-ai/dependency-parsing-with-neural-networks-e36f5166628d