Deccan Education Society's

Kirti M. Doongursee College of Arts, Science and Commerce
[AUTONOMOUS]

M.Sc. [Computer Science]

Practical Journal

Course Name: Natural Language Processing

Seat Number [ ]

(Academic Year 2024-2025)

Department of Computer Science and Information Technology

Department of Computer Science and Information Technology Deccan Education Society's

Kirti M. Doongursee College of Arts, Science and Commerce [AUTONOMOUS]

C E R T I F I C A T E

This is to certify that Mr./Miss _____of M.Sc. (C.S.) with Seat No. ___has complete_____Practical of Paper(Course Name- **Natural Language Processing**) under my supervision in this College during the Second Semester of academic year 2024-2025

Prof. Jaymala Deshpande

Lecturer-In-Charge

Date: / /2024

Dr. Apurva Yadav
H.O.D.
Department of
Computer Science & IT

Date:

Examined by:

Remarks:

Date

_____

Index

Sr.No	Practical Names	Sign
1	A. Convert the given text to speech.	
	B. Convert audio file to Text	
2	A. Study of Brown Corpus with various methods like fields, raw, words, sents ,categories.	
	B.Create and use your own corpora (plaintext, categorical)	
	C. Study of tagged corpora with method like tagged_sents , tagged_words.	
	D. Map Words to Properties Using Python Dictionaries	
3	A.Study Default tagger	
	B. Study Unigram Tagger	
4	A. Study of Wordnet Dictionary with methods as synsets, definitions, examples, antonyms	
	B.Write a program using python to find synonym and antonym of word "active" using Wordnet.	
	C. Compare two nouns	
5	A. Tokenization using Python's split() function	
	B. Tokenization using Regular Expressions (RegEx)	
	C. Tokenization using nltk	
	D. Tokenization using Spacy	
	E. Tokenization using Keras	
	F.Tokenization using Gensim	
6	A. Named Entity recognition using user defined text.	
	B. Named Entity recognition with diagram using NLTK corpus – treebank. Program:	
7	A. Define grammar using nltk. Analyze a sentence using the same	
	B. Implementation of Deductive Chart Parsing using context free grammar and a given sentence	
8	Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer Study WordNetLemmatizer	
9	Finite Automata	

Practical - 1(a)

Aim: Convert the given text to speech.

Requirements:

```
pip install nltk
pip install gtts
pip install --upgrade wheel
pip install playsound
```

Program:

```
from gtts import gTTS
from playsound import playsound

mytext = "Hello World"
lang = "en"

myobj = gTTS(text = mytext, lang = lang, slow = False)

myobj.save("./myFile.mp3")
playsound("./myFile.mp3")
```

Output:

myFile.mp3 audio file is getting created and it plays the file with playsound() method, while running the program

Practical - 1(b)

Aim: Convert audio file to Text.

Requirements:

```
pip install SpeechRecognition
```

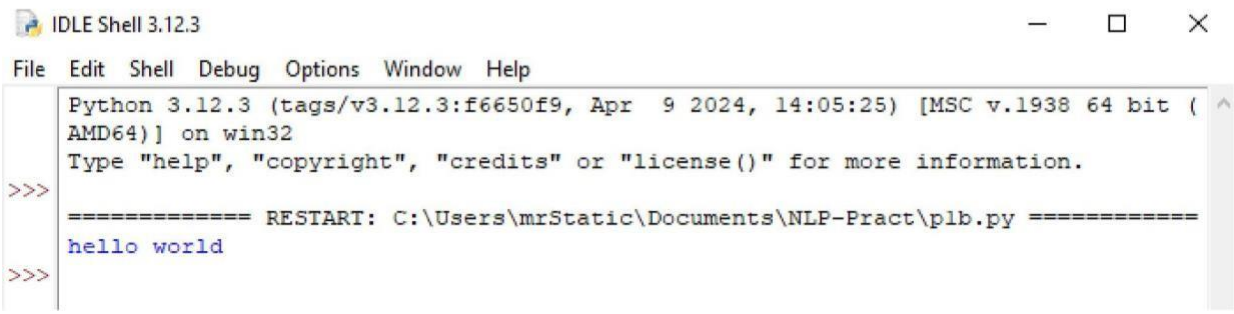
Program:

```
import speech_recognition as sr

filename = "./myFile.wav"
r = sr.Recognizer()

with sr.AudioFile(filename) as source:
    audio_data = r.record(source)
    text = r.recognize_google(audio_data)
    print(text)
```

Output:



Practical - 2(a)

Aim: Study of Brown Corpus with various methods like filelds, raw, words, sents, categories.

Requirements:

```
pip install nltk
nltk.download("brown")
```

Program:

```
import nltk
nltk.download("brown")
from nltk.corpus import brown

print("File ids of brown corpus\n", brown. fileids())

ca01 = brown.words("ca01")
print("\nca01 has following words:\n", ca01)
print("\nca01 has ", len(ca01), " words")

print("\n\nCategories or file in brown corpus: \n")
print(brown.categories())

print("\n\nStatistics for each text:\n")
print("Avg-Word-Len\tAvg-Sentence-Len\tNo.-of-Times-Each-Word-Appears-On-
Avg\t\tFile-Name")
for fileid in brown.fileids():
        num_chars = len(brown.raw(fileid))
        num_words = len(brown.words(fileid))
        num_sents = len(brown.sents(fileid))
        num_vocab = len(set([w.lower() for w in brown.words(fileid)]))
        print(int(num_chars/num_words), "\t\t\t", num_words/num_sents,
"\t\t\t", int(num_words/num_vocab), "\t\t\t", fileid)
```

Output:

```
============ RESTART: C:/Users/mrStatic/Documents/NLP-Pract/p2a.py ===========
[nltk_data] Downloading package brown to
[nltk_data]     C:\Users\mrStatic\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\brown.zip.
File ids of brown corpus
 ['ca01', 'ca02', 'ca03', 'ca04', 'ca05', 'ca06', 'ca07', 'ca08', 'ca09', 'ca10', 'ca11', 'ca12', 'ca13', 'ca14', 'ca15', 'ca16', 'ca17', 'ca18', 'ca19', 'ca20', 'ca21', 'ca22', 'ca23', 'ca24'
, 'ca25', 'ca26', 'ca27', 'ca28', 'ca29', 'ca30', 'ca31', 'ca32', 'ca33', 'ca34', 'ca35', 'ca36', 'ca37', 'ca38', 'ca39', 'ca40', 'ca41', 'ca42', 'ca43', 'ca44', 'cb01', 'cb02', 'cb03', 'cb04'
, 'cb05', 'cb06', 'cb07', 'cb08', 'cb09', 'cb10', 'cb11', 'cb12', 'cb13', 'cb14', 'cb15', 'cb16', 'cb17', 'cb18', 'cb19', 'cb20', 'cb21', 'cb22', 'cb23', 'cb24', 'cb25', 'cb26', 'cb27', 'cc01'
, 'cc02', 'cc03', 'cc04', 'cc05', 'cc06', 'cc07', 'cc08', 'cc09', 'cc10', 'cc11', 'cc12', 'cc13', 'cc14', 'cc15', 'cc16', 'cc17', 'cd01', 'cd02', 'cd03', 'cd04', 'cd05', 'cd06', 'cd07', 'cd08'
, 'cd09', 'cd10', 'cd11', 'cd12', 'cd13', 'cd14', 'cd15', 'cd16', 'cd17', 'ce01', 'ce02', 'ce03', 'ce04', 'ce05', 'ce06', 'ce07', 'ce08', 'ce09', 'ce10', 'ce11', 'ce12', 'ce13', 'ce14', 'ce15'
, 'ce16', 'ce17', 'ce18', 'ce19', 'ce20', 'ce21', 'ce22', 'ce23', 'ce24', 'ce25', 'ce26', 'ce27', 'ce28', 'ce29', 'ce30', 'ce31', 'ce32', 'ce33', 'ce34', 'ce35', 'ce36', 'cf01', 'cf02', 'cf03'
, 'cf04', 'cf05', 'cf06', 'cf07', 'cf08', 'cf09', 'cf10', 'cf11', 'cf12', 'cf13', 'cf14', 'cf15', 'cf16', 'cf17', 'cf18', 'cf19', 'cf20', 'cf21', 'cf22', 'cf23', 'cf24', 'cf25', 'cf26', 'cf27'
, 'cf28', 'cf29', 'cf30', 'cf31', 'cf32', 'cf33', 'cf34', 'cf35', 'cf36', 'cf37', 'cf38', 'cf39', 'cf40', 'cf41', 'cf42', 'cf43', 'cf44', 'cf45', 'cf46', 'cf47', 'cf48', 'cg01', 'cg02', 'cg03'
, 'cg04', 'cg05', 'cg06', 'cg07', 'cg08', 'cg09', 'cg10', 'cg11', 'cg12', 'cg13', 'cg14', 'cg15', 'cg16', 'cg17', 'cg18', 'cg19', 'cg20', 'cg21', 'cg22', 'cg23', 'cg24', 'cg25', 'cg26', 'cg27'
, 'cg28', 'cg29', 'cg30', 'cg31', 'cg32', 'cg33', 'cg34', 'cg35', 'cg36', 'cg37', 'cg38', 'cg39', 'cg40', 'cg41', 'cg42', 'cg43', 'cg44', 'cg45', 'cg46', 'cg47', 'cg48', 'cg49', 'cg50', 'cg51'
, 'cg52', 'cg53', 'cg54', 'cg55', 'cg56', 'cg57', 'cg58', 'cg59', 'cg60', 'cg61', 'cg62', 'cg63', 'cg64', 'cg65', 'cg66', 'cg67', 'cg68', 'cg69', 'cg70', 'cg71', 'cg72', 'cg73', 'cg74', 'cg75'
, 'ch01', 'ch02', 'ch03', 'ch04', 'ch05', 'ch06', 'ch07', 'ch08', 'ch09', 'ch10', 'ch11', 'ch12', 'ch13', 'ch14', 'ch15', 'ch16', 'ch17', 'ch18', 'ch19', 'ch20', 'ch21', 'ch22', 'ch23', 'ch24'
, 'ch25', 'ch26', 'ch27', 'ch28', 'ch29', 'ch30', 'cj01', 'cj02', 'cj03', 'cj04', 'cj05', 'cj06', 'cj07', 'cj08', 'cj09', 'cj10', 'cj11', 'cj12', 'cj13', 'cj14', 'cj15', 'cj16', 'cj17', 'cj18'
, 'cj19', 'cj20', 'cj21', 'cj22', 'cj23', 'cj24', 'cj25', 'cj26', 'cj27', 'cj28', 'cj29', 'cj30', 'cj31', 'cj32', 'cj33', 'cj34', 'cj35', 'cj36', 'cj37', 'cj38', 'cj39', 'cj40', 'cj41', 'cj42'
, 'cj43', 'cj44', 'cj45', 'cj46', 'cj47', 'cj48', 'cj49', 'cj50', 'cj51', 'cj52', 'cj53', 'cj54', 'cj55', 'cj56', 'cj57', 'cj58', 'cj59', 'cj60', 'cj61', 'cj62', 'cj63', 'cj64', 'cj65', 'cj66'
, 'cj67', 'cj68', 'cj69', 'cj70', 'cj71', 'cj72', 'cj73', 'cj74', 'cj75', 'cj76', 'cj77', 'cj78', 'cj79', 'cj80', 'ck01', 'ck02', 'ck03', 'ck04', 'ck05', 'ck06', 'ck07', 'ck08', 'ck09', 'ck10'
, 'ck11', 'ck12', 'ck13', 'ck14', 'ck15', 'ck16', 'ck17', 'ck18', 'ck19', 'ck20', 'ck21', 'ck22', 'ck23', 'ck24', 'ck25', 'ck26', 'ck27', 'ck28', 'ck29', 'cl01', 'cl02', 'cl03', 'cl04', 'cl05'
, 'cl06', 'cl07', 'cl08', 'cl09', 'cl10', 'cl11', 'cl12', 'cl13', 'cl14', 'cl15', 'cl16', 'cl17', 'cl18', 'cl19', 'cl20', 'cl21', 'cl22', 'cl23', 'cl24', 'cm01', 'cm02', 'cm03', 'cm04', 'cm05'
, 'cm06', 'cn01', 'cn02', 'cn03', 'cn04', 'cn05', 'cn06', 'cn07', 'cn08', 'cn09', 'cn10', 'cn11', 'cn12', 'cn13', 'cn14', 'cn15', 'cn16', 'cn17', 'cn18', 'cn19', 'cn20', 'cn21', 'cn22', 'cn23'
, 'cn24', 'cn25', 'cn26', 'cn27', 'cn28', 'cn29', 'cp01', 'cp02', 'cp03', 'cp04', 'cp05', 'cp06', 'cp07', 'cp08', 'cp09', 'cp10', 'cp11', 'cp12', 'cp13', 'cp14', 'cp15', 'cp16', 'cp17', 'cp18'
, 'cp19', 'cp20', 'cp21', 'cp22', 'cp23', 'cp24', 'cp25', 'cp26', 'cp27', 'cp28', 'cp29', 'cr01', 'cr02', 'cr03', 'cr04', 'cr05', 'cr06', 'cr07', 'cr08', 'cr09']

ca01 has following words:
 ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

ca01 has  2242  words


Categories or file in brown corpus:

['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']


Statistics for each text:

Avg-Word-Len     Avg-Sentence-Len        No.-of-Times-Each-Word-Appears-On-Avg           File-Name
9                22.877551020408163                    2                                ca01
8                23.234694387755102                    2                                ca02
8                20.3125                    2                                ca03
9                25.193181818181817                    2                                ca04
8                26.714285714285715                    3                                ca05
8                22.858585858585858                    2                                ca06

8                16.828767123287673                    3                                cn19
8                14.817610062893081                    2                                cn20
8                16.37062937062937                    2                                cn21
8                16.288732394366196                    3                                cn22
8                18.209302325581394                    3                                cn23
7                14.094972067039107                    3                                cn24
8                12.4                    3                                cn25
8                18.147286821705425                    2                                cn26
8                19.283464566929133                    3                                cn27
8                18.85123966942149                    2                                cn28
7                13.333333333333334                    3                                cn29
8                17.801526717557252                    3                                cp01
8                23.673469387755l                    3                                cp02
8                16.589041095890412                    3                                cp03
8                19.142857142857142                    3                                cp04
8                17.908396946564885                    2                                cp05
8                13.945054945054945                    3                                cp06
8                16.5625                    3                                cp07
8                13.24                    3                                cp08
8                31.695652173913043                    3                                cp09
8                19.398373983739837                    3                                cp10
8                19.714285714285715                    3                                cp11
7                14.488235294117647                    3                                cp12
8                20.875                    3                                cp13
7                10.826839826839826                    3                                cp14
7                14.100558659217876                    3                                cp15
8                12.198067632850242                    3                                cp16
8                19.724137931034484                    3                                cp17
7                13.345945945945946                    3                                cp18
7                14.074712643678161                    3                                cp19
7                13.340659340659341                    3                                cp20
8                21.0                    3                                cp21
8                13.569318181818182                    3                                cp22
8                14.802325581395l5                    3                                cp23
7                10.740425531914894                    4                                cp24
8                23.333333333333332                    3                                cp25
7                21.61467889908257                    4                                cp26
8                11.879227053140097                    3                                cp27
8                14.993589743589743                    3                                cp28
8                16.113333333333333                    2                                cp29
8                20.628318584070797                    2                                cr01
8                21.168141592920353                    2                                cr02
8                28.119565217391305                    2                                cr03
8                19.115644047779661                    3                                cr04
8                17.475177730496454                    2                                cr05
8                20.373983739837l                    3                                cr06
8                13.347826086956522                    3                                cr07
8                33.871428571428574                    2                                cr08
8                23.555555555555557                    2                                cr09
```

Practical - 2(b)

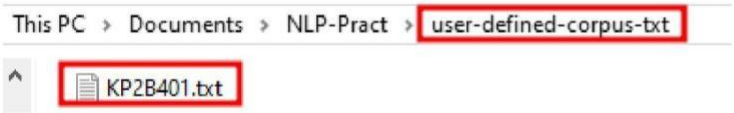
Aim: Create and use your own corpora (plaintext, categorical)

Requirements:

```
pip install nltk
nltk.download("punkt")
```

Creating user defined corpus

1. In same program directory create new folder with name `user-defined-corpus-txt`
2. Open `user-defined-corpus-txt` and create new `txt` file with any name here I have given `KP2B401.txt`
3. Open the `txt` file and enter any sentence here I have given `The Quick Brown Fox Jumpped Over The Lazy Dog`.



Program:

```
import nltk
nltk.download("punkt")
from nltk.corpus import PlaintextCorpusReader
corpus_root = "./user-defined-corpus-txt"
fileList = PlaintextCorpusReader(corpus_root, ".*")

print("\n File List \n")
print(fileList.fileids())
print(fileList.root)
print("\n\nStatistics for each text:\n")
print("Avg_Word_Len\tAvg_Sentence_Len\tno._of_Times_Each_Word_Appears_On_A
vg\tFile_Name")

for fileid in fileList.fileids():
        num_chars = len(fileList.raw(fileid))
        num_words = len(fileList.words(fileid))
        num_sents = len(fileList.sents(fileid))
        num_vocab = len(set([w.lower() for w in fileList.words(fileid)]))
        print(int(num_chars/num_words), "\t\t\t",
int(num_words/num_sents), "\t\t\t", int(num_words/num_vocab), "\t\t\t",
fileid)
```

Output:

```
==================== RESTART: C:/Users/mrStatic/Documents/NLP-Pract/p2b.py ===================
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\mrStatic\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!

 File List

['KP2B401.txt']
C:\Users\mrStatic\Documents\NLP-Pract\user-defined-corpus-txt


Statistics for each text:

Avg_Word_Len    Avg_Sentence_Len        no._of_Times_Each_Word_Appears_On_Avg   File_Name
5                       9                       1                       KP2B401.txt
```

Practical - 2(c)

Aim: Study of tagged corpora with methods like tagged_sents, tagged_words.

Requirements:

```
pip install nltk
import nltk
nltk.download('punkt')
nltk.download('words')
```

Program:

```
import nltk
nltk.download('punkt')
nltk.download('words')
from nltk import tokenize

para = "Hello World! From kirti college. Today we will be learning NLTK."
sents = tokenize.sent_tokenize(para)
print("\nsentence tokenization\n=========\n", sents)

print("\nword tokenization\n========\n")
for index in range(len(sents)):
    words = tokenize.word_tokenize(sents[index])
    print(words)
```

Output:

```
= RESTART: /home/mrportable/Documents/MiniHacker2/MSC-CS/P-2/NLP/Pract/Pract-2
/pract-2c.py
[nltk_data] Downloading package punkt to /home/mrportable/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package words to /home/mrportable/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.

sentence tokenization
=========
 ['Hello World!', 'From kirti college.', 'Today we will be learning NLTK.']

word tokenization
========

['Hello', 'World', '!']
['From', 'kirti', 'college', '.']
['Today', 'we', 'will', 'be', 'learning', 'NLTK', '.']
```

Practical - 2(d)

Aim: Map Words to Properties Using Python Dictionaries

Program:

```
thisdict={
    "brand":"Porsche",
    "model":"911 gt3 rs",
    "year":2003
    }

print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))
```

Output:

```
= RESTART: /home/mrportable/Documents/MiniHacker2/MSC-CS/P-2/NLP/Pract/Pract-2
/pract-2d.py
{'brand': 'Porsche', 'model': '911 gt3 rs', 'year': 2003}
Porsche
3
<class 'dict'>
```

Practical - 3(a)

Aim: Study Default Tagger

Requirements:

```
pip install nltk
import nltk
nltk.download('treebank')
```

Program:

```
import nltk
nltk.download('treebank')
from nltk.tag import DefaultTagger
from nltk.corpus import treebank
exptagger = DefaultTagger("NN")
testsentences=treebank.tagged_sents()[1000:]
print(exptagger.accuracy(testsentences))
print(exptagger.tag_sents([['Hey',','],['How', 'are', 'you','?']]))
```

Output:

```
= RESTART: /home/mrportable/Documents/MiniHacker2/MSC-CS/P-2/NLP/Pract/Pract-3
/pract-3a.py
[nltk_data] Downloading package treebank to
[nltk_data]     /home/mrportable/nltk_data...
[nltk_data]   Package treebank is already up-to-date!
0.13198749536374715
[[('Hey', 'NN'), (',', 'NN')], [('How', 'NN'), ('are', 'NN'), ('you', 'NN'), (
'?', 'NN')]]
```

Practical - 3(b)

Aim: Study Unigram Tagger

Requirements:

```
pip install nltk
import nltk
nltk.download('treebank')
```

Program:

```
from nltk.tag import UnigramTagger
from nltk.corpus import treebank

train_sents = treebank.tagged_sents()[:10]
tagger = UnigramTagger(train_sents)
print(treebank.sents()[0])
print('\n',tagger.tag(treebank.sents()[0]))

tagger.tag(treebank.sents()[0])
tagger = UnigramTagger(model={'Pierre':'NN'})
print('\n',tagger.tag(treebank.sents()[0]))
```

Output:

```
= RESTART: /home/mrportable/Documents/MiniHacker2/MSC-CS/P-2/NLP/Pract/Pract-3
/pract-3b.py
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the', 'b
oard', 'as', 'a', 'nonexecutive', 'director', 'Nov.', '29', '.']

 [('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'N
NS'), ('old', 'JJ'), (',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT')
, ('board', 'NN'), ('as', 'IN'), ('a', 'DT'), ('nonexecutive', 'JJ'), ('direct
or', 'NN'), ('Nov.', 'NNP'), ('29', 'CD'), ('.', '.')]

 [('Pierre', 'NN'), ('Vinken', None), (',', None), ('61', None), ('years', Non
e), ('old', None), (',', None), ('will', None), ('join', None), ('the', None),
 ('board', None), ('as', None), ('a', None), ('nonexecutive', None), ('directo
r', None), ('Nov.', None), ('29', None), ('.', None)]
```

Practical - 4(a)

Aim: Study of Wordnet Dictionary with methods as synsets, definitions, examples, antonyms

Requirements:

```
pip install nltk
nltk.download("wordnet")
```

Program:

```
import nltk
nltk.download("wordnet")
from nltk.corpus import wordnet
print(wordnet.synsets("computer"))
print(wordnet.synset("computer.n.01").definition())
print("Examples: ",wordnet.synset("computer.n.01").examples())
print(wordnet.lemma("buy.v.01.buy").antonyms())
```

Output:

```
======================== RESTART: E:/Prat-4/a/pract-4a.py ========================
[nltk_data] Downloading package wordnet to C:\Users\IT LAB PC NO
[nltk_data]     29\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[Synset('computer.n.01'), Synset('calculator.n.01')]
a machine for performing calculations automatically
Examples:  []
[Lemma('sell.v.01.sell')]
```

Practical - 4(b)

Aim: Write a program using python to find synonym and antonym of word "active" using Wordnet.

Requirements:

```
pip install nltk
nltk.download("wordnet")
```

Program:

```
from nltk.corpus import wordnet
print(wordnet.synsets("active"))
print(wordnet.lemma('active.a.01.active').antonyms())
```

Output:

```
======================= RESTART: E:/Pract-4/b/pract-4b.py =======================
[Synset('active_agent.n.01'), Synset('active_voice.n.01'), Synset('active.n.03')
, Synset('active.a.01'), Synset('active.s.02'), Synset('active.a.03'), Synset('a
ctive.s.04'), Synset('active.a.05'), Synset('active.a.06'), Synset('active.a.07'
), Synset('active.s.08'), Synset('active.a.09'), Synset('active.a.10'), Synset('
active.a.11'), Synset('active.a.12'), Synset('active.a.13'), Synset('active.a.14
')]
[Lemma('inactive.a.02.inactive')]
```

Practical - 4(c)

Aim: Compare two nouns

Requirements:

```
pip install nltk
nltk.download("wordnet")
```

Program:

```
import nltk
from nltk.corpus import wordnet
syn1 = wordnet.synsets("cricket")
syn2 = wordnet.synsets("hokey")
for s1 in syn1:
    for s2 in syn2:
        print("Path similarity of: ")
        print(s1, "(",s1.pos(),")","[",s1.definition(),"]")
        print(s1, "(",s2.pos(),")","[",s2.definition(),"]")
        print("is", s1.path_similarity(s2))
```

Output:

```
======================== RESTART: E:/Pract-4/c/pract4c.py ========================
Path similarity of:
Synset('football.n.01') ( n ) [ any of various games played with a ball (round o
r oval) in which two teams try to kick or carry or propel the ball into each oth
er's goal ]
Synset('football.n.01') ( n ) [ a football game in which two teams of 11 players
 try to kick or head a ball into the opponents' goal ]
is 0.5
Path similarity of:
Synset('football.n.02') ( n ) [ the inflated oblong ball used in playing America
n football ]
Synset('football.n.02') ( n ) [ a football game in which two teams of 11 players
 try to kick or head a ball into the opponents' goal ]
is 0.05
```

Practical - 5(a)

Aim: Tokenization using Python's split() function

Program:

```
text = """ This tool is an a beta stage. Alexa developers can use Get
Metrics API to seamlessly analyse
metric. It also supports custom skill model, prebuilt Flash Briefing
model, and the Smart Home Skill API.
You can use this tool for creation of monitors, alarms, and dashboards
that spotlight changes. The
release of these three tools will enable developers to create visual rich
skills for Alexa devices with
screens. Amazon describes these tools as the collection of tech and tools
for creating visually rich and
interactive voice experiences. """
data = text.split('.')
for i in data:
    print (i)
```

Output:

```
======================== RESTART: E:/Pract-5/a/pract-5a.py ========================
 This tool is an a beta stage
 Alexa developers can use Get Metrics API to seamlessly analyse
metric
 It also supports custom skill model, prebuilt Flash Briefing model, and the Smart
Home Skill API

You can use this tool for creation of monitors, alarms, and dashboards that spotli
ght changes
 The
release of these three tools will enable developers to create visual rich skills f
or Alexa devices with
screens
 Amazon describes these tools as the collection of tech and tools for creating vis
ually rich and
interactive voice experiences
```

Practical - 5(b)

Aim: Tokenization using Regular Expressions (RegEx)

Requirements:

```
pip install nltk
```

Program:

```
import nltk
from nltk.tokenize import RegexpTokenizer
tk = RegexpTokenizer('\s+',gaps = True)
str = "Let's use RegexpTokenizer to split"
tokens = tk.tokenize(str)
print(tokens)
```

Output:

```
======================== RESTART: E:/Pract-5/b/pract-5b.py ========================
["Let's", 'use', 'RegexpTokenizer', 'to', 'split']
```

Practical - 5(c)

Aim: Tokenization using NLTK

Requirements:

```
pip install nltk
```

Program:

```
import nltk
from nltk.tokenize import word_tokenize
str = "Let's use word_tokenize to split"
print(word_tokenize(str))
```

Output:

```
======================== RESTART: E:/Pract-5/e/Pract-5e.py ========================
['Let', "'s", 'use', 'word_tokenize', 'to', 'split']
```

Practical - 5(d)

Aim:

Requirements:

```
pip install spacy
```

Program:

```
import spacy
nlp = spacy.blank("en")
str = "Let's use spacy to split"
doc = nlp(str)
words = [word.text for word in doc]
print(words)
```

Output:

```
======================== RESTART: E:/Pract-5/d/pract-5d.py ========================
['Let', "'s", 'use', 'spacy', 'to', 'split']
```

Practical - 5(e)

Aim: Tokenization using Keras

Requirements:

```
pip install tensorflow
pip install keras
pip install Keras-Preprocessing
```

Program:

```
import keras
from keras.preprocessing.text import text_to_word_sequence
str = "Let's use keras to split"
tokens = text_to_word_sequence(str)
print(tokens)
```

Output:

```
["let's", 'use', 'keras', 'to', 'split']
```

Practical - 5(f)

Aim: Tokenization using Gensim

Requirements:

```
pip install gensim

gensim requires C++
```

Program:

```
from gensim.utils import tokenize
str = "I love to study Natural Language Processing in Python"
list(tokenize(str))
```

Output:

```
['I',
 'love',
 'to',
 'study',
 'Natural',
 'Language',
 'Processing',
 'in',
 'Python']
```

Practical - 6(a)

Aim: Named Entity recognition using user defined text.

Requirements:

```
pip install spacy
python -m spacy download en_core_web_sm
```

Program:

```
import spacy
nlp = spacy.load("en_core_web_sm")
text = ("When Sebastian Thrun started working on self-driving cars at "
"Google in 2007, few people outside of the company took him "
"seriously. “I can tell you very senior CEOs of major American "
"car companies would shake my hand and turn away because I wasn’t "
"worth talking to,” said Thrun, in an interview with Recode earlier "
"this week.")

doc = nlp(text)
print("Noun phrases: ",[chunk.text for chunk in doc.noun_chunks])
print("Verbs: ",[token.lemma_ for token in doc if token.pos_=="VERB"])
```

Output:

```
= RESTART: /home/mrportable/Documents/MiniHacker2/MSC-CS/P-2/NLP/Pract/Pract-6/p
ract-6a.py
Noun phrases:  ['Sebastian Thrun', 'self-driving cars', 'Google', 'few people',
'the company', 'him', 'I', 'you', 'very senior CEOs', 'major American car compan
ies', 'my hand', 'I', 'Thrun', 'an interview', 'Recode']
Verbs:  ['start', 'work', 'drive', 'take', 'tell', 'shake', 'turn', 'talk', 'say
']
```

Practical - 6(b)

Aim: Named Entity recognition with diagram using NLTK corpus – treebank.

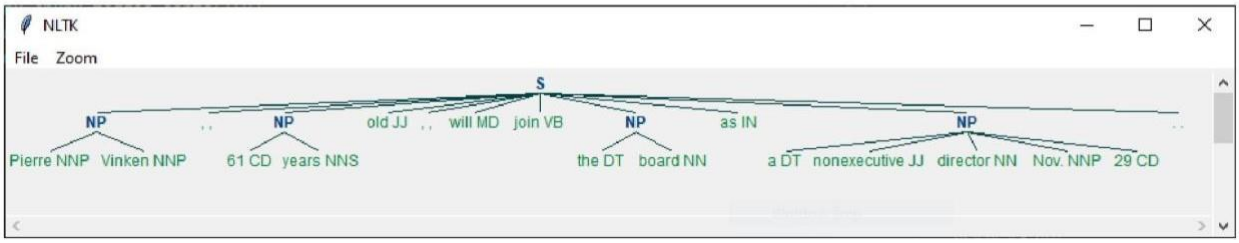
Requirements:

```
pip install nltk
nltk.download('treebank')
```

Program:

```
import nltk
nltk.download('treebank')
from nltk.corpus import treebank_chunk
treebank_chunk.tagged_sents()[0]
treebank_chunk.chunked_sents()[0]
treebank_chunk.chunked_sents()[0].draw()
```

Output:



Practical - 7(a)

Aim: Define grammar using nltk. Analyze a sentence using the same

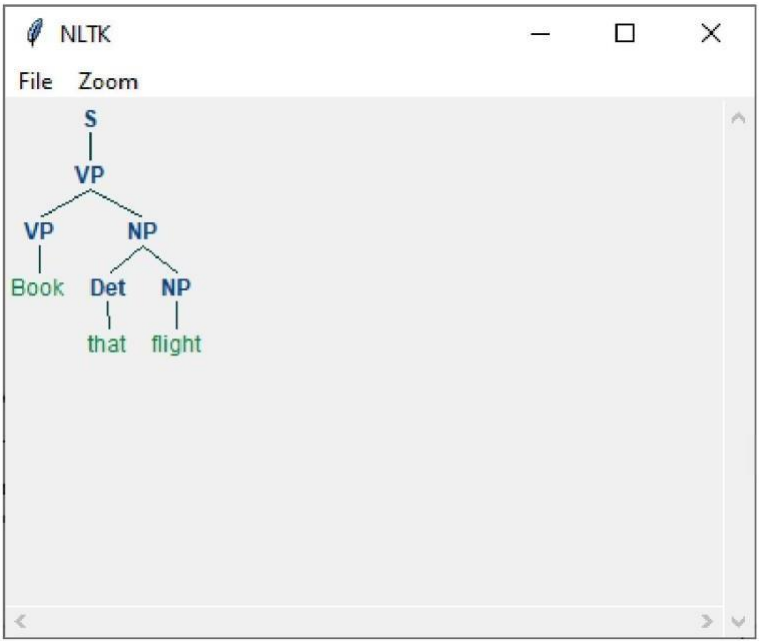
Requirements:

```
pip install nltk
import nltk
nltk.download("punkt")
nltk.download("treebank")
```

Program:

```
import nltk
from nltk import tokenize
nltk.download("punkt")
nltk.download("treebank")
grammar1 = nltk.CFG.fromstring("""
S -> VP
VP -> VP NP
NP -> Det NP
Det -> 'that'
NP -> singular Noun
NP -> 'flight'
VP -> 'Book'
""")
sentence = "Book that flight"
for index in range(len(sentence)):
    all_tokens = tokenize.word_tokenize(sentence)
print(all_tokens)
parser = nltk.ChartParser(grammar1)
for tree in parser.parse(all_tokens):
    print(tree)
    tree.draw()
```

Output:


NLTK
File
Zoom
S
VP
VP
NP
Book
Det
NP
that
flight

Practical - 7(b)

Aim: Implementation of Deductive Chart Parsing using context free grammar and a given sentence.

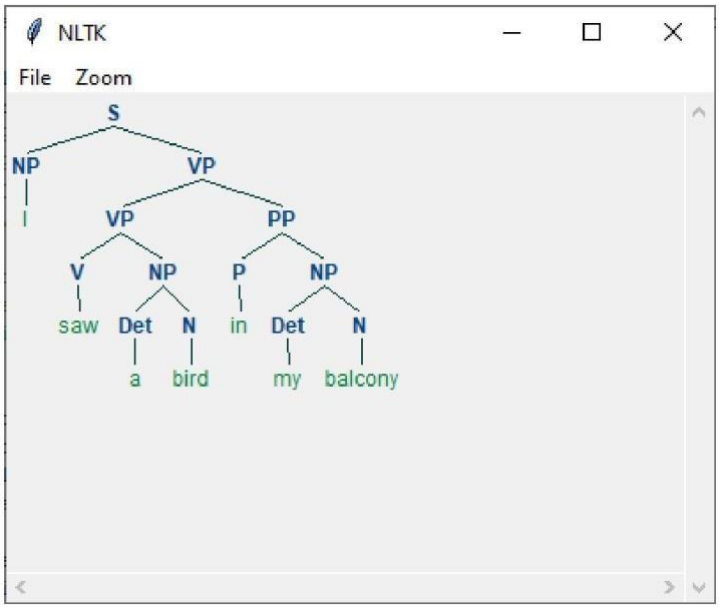
Requirements:

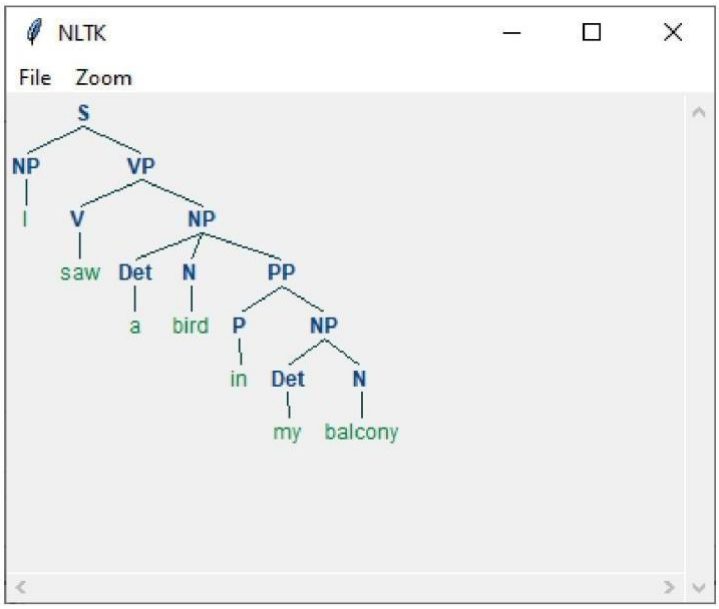
```
pip install nltk
```

Program:

```
import nltk
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'a' | 'my'
N -> 'bird' | 'balcony'
V -> 'saw'
P -> 'in'
""")
sentence = "I saw a bird in my balcony"
for index in range (len(sentence)):
    all_tokens = tokenize.word_tokenize(sentence)
print(all_tokens)
parser = nltk.ChartParser(grammar1)
for tree in parser.parse(all_tokens):
    print(tree)
    tree.draw()
```

Output:


NLTK
File Zoom
S
NP
VP
I
VP
PP
V
NP
P
NP
saw
Det
N
in
Det
N
a
bird
my
balcony


NLTK
File Zoom
S
NP
VP
I
V
NP
saw
Det
N
PP
a
bird
P
NP
in
Det
N
my
balcony

Practical - 8

Aim: Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer Study WordNetLemmatizer

Requirements:

```
pip install nltk
import nltk
nltk.download("wordnet")
```

Program:

```
import nltk
nltk.download("wordnet")
word = "running"
print("PoterStemmer")
from nltk.stem import PorterStemmer
word_stemmer = PorterStemmer()
print(word_stemmer.stem(word))

print("---------------------------------")
print("Lancaster Stemmer")
from nltk.stem import LancasterStemmer
Lanc_stemmer = LancasterStemmer()
print(Lanc_stemmer.stem(word))

print("---------------------------------")
print('RegexpStemmer')
import nltk
from nltk.stem import RegexpStemmer
Reg_stemmer = RegexpStemmer('ing$|s$|e$|able$', min=4)
print(Reg_stemmer.stem(word))


print("---------------------------------")
print('SnowballStemmer')
from nltk.stem import SnowballStemmer
english_stemmer = SnowballStemmer('english')
print(english_stemmer.stem (word))

print("---------------------------------")
print('WordNetLemmatizer')
```

```python
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print("word :\tlemma")
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))

print("better :", lemmatizer.lemmatize("better", pos ="a"))
```

Output:

```
========================= RESTART: E:/Pract-8/Pract-8.py =============
[nltk_data] Downloading package wordnet to C:\Users\IT LAB PC NO
[nltk_data]     29\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
PoterStemmer
run
--------------------------------
Lancaster Stemmer
run
--------------------------------
RegexpStemmer
runn
--------------------------------
SnowballStemmer
run
--------------------------------
WordNetLemmatizer
word :  lemma
rocks : rock
corpora : corpus
better : good
```

Practical - 9

Aim: Finite Automata

Program:

```
def FA(s):
    if len(s) < 3:
        return "Rejected"

    if s[0] == '1':
        if s[1] == '0':
            if s[2] == '1':
                for i in range(3, len(s)):
                    if s[i] != "1":
                        return "Rejected"
                return "Accepted"
            return "Rejected"
        return "Rejected"
    return "Rejected"

inputs=['1','10101','101','10111','01010','100','','10111101','1011111']
for i in inputs:
    print(i, FA(i))
```

Output:

```
========================= RESTART: E:/Pract-9/pract-9.py =========================
1 Rejected
10101 Rejected
101 Accepted
10111 Accepted
01010 Rejected
100 Rejected
 Rejected
10111101 Rejected
1011111 Accepted
```