



**RAMNIRANJAN JHUNJHUNWALA COLLEGE**

**GHATKOPAR (W), MUMBAI - 400 086**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**2021 - 2022**

**M.Sc. ( I.T) SEM IV**

**Subject: Natural Language Processing**

**Name: Sneha Ramchandra Pawar**

**Roll No.: 18**



Hindi Vidya Prachar Samiti's  
**RAMNIRANJAN**  
**JHUNJHUNWALA COLLEGE**  
**(AUTONOMOUS)**

Opposite Ghatkopar Railway Station, Ghatkopar West, Mumbai-400086



## CERTIFICATE

This is to certify that Miss. **Sneha Ramchandra Pawar** with Roll No. **18** has successfully completed the necessary course of experiments in the subject of **Natural Language Processing** during the academic year **2021 – 2022** complying with the requirements of **RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ARTS, SCIENCE AND COMMERCE**, for the course of **M.Sc. (IT)** semester -IV.

---

Internal Examiner

---

External Examiner

---

Head of Department

---

College Seal

## INDEX

<b>Practical No.</b>	<b>Aim</b>	<b>Date</b>
1.	<p>A. Convert the given text to speech</p> <p>B. Convert audio file speech to text</p>	
2.	<p>A. Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like fields, raw, words, sents, categories,</p> <p>B. Create and use your own corpora (plaintext categorical)</p> <p>C. Study Conditional Frequency Distribution</p> <p>D. Study of Tagged Corpora with methods like tagged_sents, tagged_words</p> <p>E. Write a program to find the most frequent noun tags</p> <p>F. Map words to properties using Python Dictionaries</p> <p>G. Study i) Default tagger ii). Regular expression tagger iii) Unigram tagger</p> <p>H. Find different words from a given text without any space by comparing this text with a given corpus of words. Also find the score of words</p>	
3	<p>A. Study of WordNet dictionary with methods as synsets, definitions examples antonyms</p> <p>B. Study of lemmas hyponyms hypernyms</p> <p>C. Write a program using python to find synonyms and antonyms of word “active” using WordNet</p> <p>D. Compare two nouns Handling stopwords</p>	

	<p>i). Using nltk Adding or Removing Stop Words in NLTK's Default Stop Word List</p> <p>ii) Using Gensim Adding and Removing Stop Words in Default Gensim Stop Words List</p> <p>iii) Using Spacy Adding and Removing Stop Words in Default Spacy Stop Words List</p>	
4	<p>Text Tokenization</p> <p>A. Tokenization using python split() function</p> <p>B. Tokenization using Regular Expressions</p> <p>C. Tokenization using nltk</p> <p>D. Tokenization using spaCy library</p> <p>E. Tokenization using keras</p> <p>F. Tokenization using Gensim</p>	
5.	<p>Import NLP Libraries for indian language and perform</p> <p>A. Word tokenization in hindi</p> <p>B. Generate similar sentences from a given hindi text input</p> <p>C. Identify the indian language of a text</p>	
6.	<p>Illustrate parts of speech tagging</p> <p>A. Part of speech Tagging and chunking of user defined text.</p> <p>B. Named Entity recognition of user defined text.</p>	
7.	<p>Finite State Automata</p> <p>A. Define grammar using nltk. Analyze a sentence using the same.</p> <p>B. Accept the input string with regular expression of finite automation: <math>101^+</math></p> <p>C. Accept the input string with regular expression of FA: <math>(a+b)^*bba</math></p> <p>D. Implementation of Deductive Chart Parsing using context free grammar and a given sentence.</p>	

8.	Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer Study WordNetLemmatizer	
9.	Implement Naïve Bayes Classifier	
10.	<p>Speech Tagging</p> <p>A. i) Using spacy ii) Using nltk</p> <p>B. Statistical Parsing</p> <p>i) Using give and Gave in the Penn treebak sample ii) Probabilistic parser</p> <p>C. Malt Parsing: Parse a sentence and draw a tree using malt parsing</p>	
11.	<p>A. Multiword expression in NLP</p> <p>B. Normalized Web Distance and word similarity</p> <p>C. Word Sense Disambiguation</p>	

# Practical No. 01

## A) Convert the given text to speech.

### Code:

```
!pip install playsound

!pip install gtts

from playsound import playsound
from gtts import gTTS
mytext = "hello, this is Sneha Pawar. Welcome to Natural Language Programming"
language = "en"
myobj = gTTS(text=mytext, lang=language, slow=False)

myobj.save("myfile.mp3")
```

### Output:

The screenshot shows a Google Colab interface with two tabs: 'Welcome To Colaboratory - Colab' and 'Untitled0.ipynb - Colaboratory'. The 'Untitled0.ipynb' tab is active, displaying the following code and its execution results:

```
File Edit View Insert Runtime Tools Help All changes saved
Files + Code + Text
A) Convert the given text to speech.

[1] !pip install playsound
Collecting playsound
  Downloading playsound-1.3.0.tar.gz (7.7 kB)
Building wheels for collected packages: playsound
  Building wheel for playsound (setup.py): started
  Building wheel for playsound (setup.py): finished with status "done"
  Created wheel for playsound: filename=playsound-1.3.0-py3-none-any.whl.size=7035 sha256=9bcbabfe0998f2fb8dc0d09b4b247d8801a10844ed311176574049fb55893d0
  Stored in directory: /root/.cache/pip/wheels/ba/fb/e8/es7ce1486e64dc3aa8ada4199b0ec5f9dfc07b7e22b65b0az
Successfully built playsound
Installing collected packages: playsound
Successfully installed playsound-1.3.0

[2] !pip install gtts
Collecting gtts
  Downloading gtts-2.2.4-py3-none-any.whl (26 kB)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from gtts) (1.15.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from gtts) (2021.10.6)
Requirement already satisfied: requests>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from gtts) (2.23.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,>1.25.1 in /usr/local/lib/python3.7/dist-packages (from gtts) (1.24.3)
Requirement already satisfied: charsetet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from gtts) (2.23.0))
Requirement already satisfied: idna>=2.0,<3.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from gtts) (2.23.0))
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from gtts) (2.23.0))
Requirement already satisfied: idna>=2.5,<3 in /usr/local/lib/python3.7/dist-packages (from requests>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from gtts) (2.23.0))
Successfully installed gtts-2.2.4

[3] from playsound import playsound
playsound is relying on another python subprocess. Please use 'pip install pygobject' if you want playsound to run more efficiently.

[4] from gtts import gTTS

[5] mytext = "hello, this is Sneha Pawar. Welcome to Natural Language Programming"
language = "en"
myobj = gTTS(text=mytext,lang=language,slow=False)

[6] myobj.save("myfile.mp3")
```

The bottom status bar shows system information: 31°C Haze, 68.15 GB available, OS completed at 9:55 AM, ENG IN, 10:05 AM, 4/9/2022.

Welcome To Colaboratory - Colab x Untitled0.ipynb - Colaboratory x + colab.research.google.com/drive/1TUMjL0P6xhXbjNalx7HN2sMJIVQNY7E?hl=en#scrollTo=eKE4l8d0j9KA

Untitled0.ipynb ★

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[1] !pip install playsound

```
Collecting playsound
  Downloading playsound-1.3.0.tgz (7.7 kB)
    Building wheels for collected packages: playsound
      Download for playsound (setup.py) ... done
        for playsound (filename=playsound-1.3.0-py3-none-any.whl size=7035 sha256=0bd9cabfe0098f2fdb8cb0d0b4b247d80b1a10844ed3117657484fb55893d0
          ctory: /root/.cache/pip/wheels/ba/f8/bb/ea57c0146b664dc3a0ada4199b0ec5f9dfcb7b7e22b65ba2
        Rename file lt playsound
        Delete file cted packages: playsound
        Talled playsound-1.3.0
      Copy path
      Refresh
```

[2] \$

Collecting gtts

```
Downloaded gtts-2.2.4-py3-none-any.whl (26 kB)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from gtts) (1.15.0)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from gtts) (7.1.2)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from gtts) (2.23.0)
Requirement already satisfied: urllib3<1.25.1,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->gtts) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->gtts) (2021.10.8)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->gtts) (2.10)
Installing collected packages: gtts
Successfully installed gtts-2.2.4
```

[3] from playsound import playsound

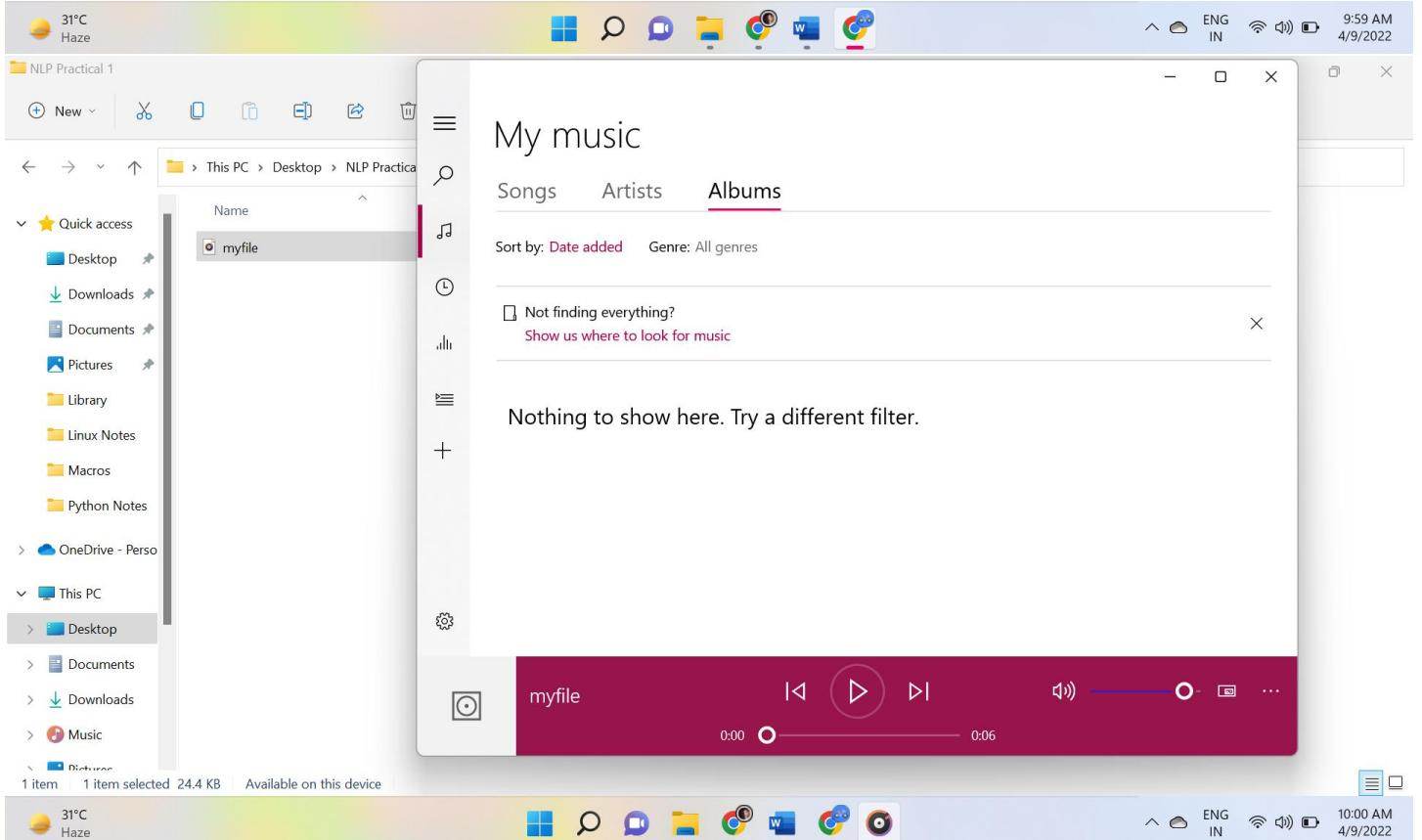
```
playsound is relying on another python subprocess. Please use `pip install pyobjc` if you want playsound to run more efficiently.
```

[4] from gtts import gTTS

[8] mytext = "hello, this is Sneha Pawar. Welcome to Natural Language Programming"
language = "en"
myobj = gTTS(text=mytext,lang=language,slow=False)

myfile.save("myfile.mp3")

Disk 68.15 GB available



## B) Convert the given text to speech

**Code:**

```
!pip3 install SpeechRecognition pydub
import speech_recognition as sr
filename = "myfileWAVExtension.wav"
r = sr.Recognizer()
with sr.AudioFile(filename) as source:
    audio_data = r.record(source)
    text = r.recognize_google(audio_data)
print(text)
```

**Output:**

The screenshot shows the Google Colab interface. The top bar displays 'Welcome To Colaboratory - Colab' and 'Untitled0.ipynb - Colaboratory'. The URL 'colab.research.google.com/drive/1TUMjL0P6xhXbxjNalx7HN2sMJIVQNY7E?hl=en#scrollTo=R2WbELiSnpDZ' is visible. The sidebar on the left shows files: 'sample\_data', 'myfile.mp3', and 'myfileWAVExtension.wav'. The main area contains the following code in cell [10]:

```
[10] import speech_recognition as sr
```

Cell [11] shows the assignment of filename:

```
[11] filename = "myfileWAVExtension.wav"
```

Cell [12] contains the speech recognition logic:

```
[12] r = sr.Recognizer()

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

The output of cell [12] is displayed in a text box:

```
hello this is not welcome to natural language programming
```

The bottom status bar shows '0s completed at 10:13 AM', system icons like battery and signal strength, and the date/time '10:15 AM 4/9/2022'.

# Practical No. 02

## A. Study of various corpus- brown, inaugural, reuters, udhr with various methods like fields, raw words, sents, categories

### Code & Output:

```
import nltk
nltk.download('brown')
from nltk.corpus import brown
print('File ids of brown corpus\n',brown.fileids())
```

```
ca01 = brown.words('ca01')
#display the first few words
print('\nca01 has following words:\n',ca01)
```

```
#total number of words in ca01
print('\nca01 has',len(ca01),'words')
```

```
#categories or files
print ('\n\nCategories or file in brown corpus:\n')
print (brown.categories())
```

The screenshot shows a Google Colab notebook titled "Untitled0.ipynb". The code cell contains four numbered steps. Step 1 imports nltk and downloads the 'brown' corpus. Step 2 defines 'ca01' as the words from the 'ca01' file in the brown corpus and prints the first few words. Step 3 prints the total number of words in 'ca01'. Step 4 prints the categories of files in the brown corpus. The output pane shows the results of each step, including the list of words for 'ca01' and the category names for the brown corpus.

```
[1] import nltk
nltk.download('brown')
from nltk.corpus import brown
print('File ids of brown corpus\n',brown.fileids())

[nltk_data] Downloading package brown to /root/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']

[2] ca01 = brown.words('ca01')
#display the first few words
print('\nca01 has following words:\n',ca01)

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

[3] #total number of words in ca01
print('\nca01 has',len(ca01),'words')

ca01 has 2242 words

[4] #categories or files
print ('\n\nCategories or file in brown corpus:\n')
print (brown.categories())

Categories or file in brown corpus:
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']
```

```

print ('\n\nStatistics for each text:\n')
print('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\t\tFileName')
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', int(num_words/num_sents),'\t\t\t', int(num_
words/num_vocab),'\t\t\t', fileid)

print ('\n\nStatistics for each text:\n')
print('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\t\tFileName')
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', int(num_words/num_sents),'\t\t\t', int(num_
words/num_vocab),'\t\t\t', fileid)

```

The screenshot shows a Google Colab session with two code cells. The left cell runs a script to print statistics for the Brown corpus, and the right cell runs a similar script for a custom corpus named 'mytext'. Both cells output tables of statistics.

**Cell 1 (Left): Statistics for the Brown corpus**

AvgWordLen	AvgSentenceLen	no.ofTimesEachWordAppearsOnAvg	FileName
8	23	2	cr09

**Cell 2 (Right): Statistics for the 'mytext' corpus**

AvgWordLen	AvgSentenceLen	no.ofTimesEachWordAppearsOnAvg	FileName
9	22	2	ca01
8	23	2	ca02
8	20	2	ca03
9	25	2	ca04
8	26	3	ca05
8	22	2	ca06
9	18	2	ca07

## B. Create and use your own corpora (plaintext, categorical)

## C. Code & Output:

```
D. import nltk
E. nltk.download('punkt')
F. from nltk.corpus import PlaintextCorpusReader
G. corpus_root = '/root/nltk_data/corpora/corpus_root'
H. filelist = PlaintextCorpusReader(corpus_root, '.*')
I. print ('\n File list: \n')
J. print (filelist.fileids())
K. print (filelist.root)
L. print ('\n\nStatistics for each text:\n')
M. print('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\tFileName')
N. for fileid in filelist.fileids():
O.     num_chars = len(filelist.raw(fileid))
P.     num_words= len(filelist.words(fileid))
Q.     num_sents = len(filelist.sents(fileid))
R.     num_vocab = len(set([w.lower() for w in filelist.words(fileid)]))
S.     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)
```

The screenshot shows a Google Colab notebook titled "Untitled0.ipynb". The code cell contains the provided Python script for creating and using corpora. The output pane shows the execution of the code, including the download of the 'punkt' package, the creation of a file list, and the resulting statistics for two files: 'mytext.txt' and 'mytext1.txt'. The statistics table is as follows:

AvgWordLen	AvgSentenceLen	no.ofTimesEachWordAppearsOnAvg	FileName
5	31	2	mytext.txt
5	24	2	mytext1.txt

The bottom status bar indicates the session completed at 11:20 AM on 4/9/2022, with a system temperature of 33°C and a haze condition.

```
import nltk
nltk.download('punkt')
from nltk.corpus import PlaintextCorpusReader
corpus_root = '/root/nltk_data/corpora/corpus_root'
filelist = PlaintextCorpusReader(corpus_root, '.*')
print ('\n File list: \n')
print (filelist.fileids())
print (filelist.root)
print ('\n\nStatistics for each text:\n')
print('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\tFileName')
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)

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]  Unzipping tokenizers/punkt.zip.

File list:

['mytext.txt', 'mytext1.txt']
/root/nltk_data/corpora/corpus_root

Statistics for each text:

AvgWordLen      AvgSentenceLen      no.ofTimesEachWordAppearsOnAvg      FileName
5                  31                      2                  mytext.txt
5                  24                      2                  mytext1.txt
```

## C. Study Conditional Frequency Distribution

```
#processing a sequence of pairs
text = ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
pairs = [('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ...]
import nltk
nltk.download('brown')
from nltk.corpus import brown
```

```

fd = nltk.ConditionalFreqDist(
    (genre, word)
    for genre in brown.categories()
    for word in brown.words(categories=genre))
genre_word = [(genre, word)
    for genre in ['news', 'romance']
    for word in brown.words(categories=genre)]
print(len(genre_word))

print(genre_word[:4])

print(genre_word[-4:])

cfд = nltk.ConditionalFreqDist(genre_word)
print(cfд)
print(cfд.conditions())
print(cfд['news'])
print(cfд['romance'])
print(list(cfд['romance']))

nltk.download('inaugural')
nltk.download('udhr')
from nltk.corpus import inaugural
cfд = nltk.ConditionalFreqDist(genre_word)
print(cfд)
print(cfд.conditions())
print(cfд['news'])
print(cfд['romance'])
print(list(cfд['romance']))
cfд = nltk.ConditionalFreqDist(
    (target, fileid[:4])
    for fileid in inaugural.fileids()
    for w in inaugural.words(fileid)
    for target in ['america', 'citizen']
    if w.lower().startswith(target))
from nltk.corpus import udhr
languages = ['Chickasaw', 'English', 'German_Deutsch',
    'Greenlandic_Inuktikut', 'Hungarian_Magyar', 'Ibibio_Efik']
cfд = nltk.ConditionalFreqDist(
    (lang, len(word))
    for lang in languages
    for word in udhr.words(lang + '-Latin1'))
cfд.tabulate(conditions=['English', 'German_Deutsch'],
    samples=range(10), cumulative=True)

```

Colab Research Google Drive

NLP\_Practical 2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share RAM Disk Editing

Files

sample\_data

C. Study Conditional Frequency Distribution

```
[17] #processing a sequence of pairs
text = ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
pairs = [('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ...]
import nltk
nltk.download('brown')
from nltk.corpus import brown
fd = nltk.ConditionalFreqDist(
    (genre, word)
    for genre in brown.categories()
    for word in brown.words(categories=genre))
genre_word = [(genre, word)
    for genre in ['news', 'romance']
    for word in brown.words(categories=genre)]
print(len(genre_word))

[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data] Package brown is already up-to-date!
170576

[18] print(genre_word[:4])
[('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ('news', 'Grand')]

[19] print(genre_word[-4:])
[('romance', 'afraid'), ('romance', 'not'), ('romance', ''), ('romance', '.')]

[20] cfd = nltk.ConditionalFreqDist(genre_word)
print(cfd)
print(cfd.conditions())
print(cfd['news'])
print(cfd['romance'])
print(list(cfd['romance']))
```

Disk 68.07 GB available

Completed at 12:19 PM

>Welcome To Colaboratory | NLP\_Practical 1\_Covert Au | NLP\_Practical 2.ipynb - C | Python: TypeError: object | What is Natural Language | +

colab.research.google.com/drive/1MOzF6C-Ea-NvNiklyCES7\_fvs95\_E5rh?hl=en#scrollTo=8Z0OPYxc8qTh

NLP\_Practical 2.ipynb

All changes saved

Files

(x) sample\_data

+ Code + Text

```
[28] cfd = nltk.ConditionalFreqDist(genre_word)
print(cfd)
print(cfd.conditions())
print(cfd['news'])
print(cfd['romance'])
print(list(cfd['romance']))

<conditionalFreqDist with 2 conditions>
['news', 'romance']
<FreqDist with 14394 samples and 100554 outcomes>
<FreqDist with 8452 samples and 70022 outcomes>
['They', 'neither', 'liked', 'nor', 'disliked', 'the', 'old', 'Man', '.', 'To', 'them', 'he', 'could', 'have', 'been', 'broken', 'bell', 'in', 'church', 'tower', 'which', 'rang', 'before', 'and', 'after', 'mass', ',', 'at', 'noon', 'six', 'each', 'evening', '--', 'its', 'tone']

nltk.download('inaugural')
nltk.download('udhr')
from nltk.corpus import inaugural
cf = nltk.ConditionalFreqDist(genre_word)
print(cf)
print(cf.conditions())
print(cf['news'])
print(cf['romance'])
print(list(cfd['romance']))
fd = nltk.ConditionalFreqDist(
    (target, fileid,)
    for target in fd.conditions()
    for fileid in fd[target].keys()
    for w in fd[target][fileid]
    for target in ['america', 'citizen']
    if w.lower().startswith(target))
if w.lower().startswith('america'):
    languages = ['Chickasaw', 'English', 'German_Deutsch',
    'Greenlandic_Inuitut', 'Hungarian_Magyar', 'Ibibio_Efik']
    cf = nltk.ConditionalFreqDist(
        (lang, lang)
        for lang in languages
        for word in udhr.words(lang + '-latin1'))
    fd.tabulate(conditions=[english, 'German_Deutsch'],
    samples=range(10), cumulative=True)

[Ink_data] Downloading package inaugural to /root/nltk_data...
[Ink_data] Unzipping corpora/inaugural.zip...
[Ink_data] Downloading package udhr to /root/nltk_data...
[Ink_data] Unzipping corpora/udhr.zip...
<conditionalFreqDist with 2 conditions>
<FreqDist with 14394 samples and 100554 outcomes>
<FreqDist with 8452 samples and 70022 outcomes>
['They', 'neither', 'liked', 'nor', 'disliked', 'the', 'old', 'Man', '.', 'To', 'them', 'he', 'could', 'have', 'been', 'broken', 'bell', 'in', 'church', 'tower', 'which', 'rang', 'before', 'and', 'after', 'mass', ',', 'at', 'noon', 'six', 'each', 'evening', '--', 'its', 'tone']

English 0 185 625 883 997 1166 1281 1448 1558 1638
German_Deutsch 0 171 263 814 727 894 1013 1110 1213 1275
```

#### D. Study of Tagged Corpora with methods like tagged\_sents, tagged\_words

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

```

nltk.download('words')
para = "Hello! My name is Sneha Pawar. Today we'll be learning NLTK. It is one of the
powerful library of NLP"
sents = tokenize.sent_tokenize(para)
print("\nsentence tokenization\n=====\\n",sents)
# word tokenization
print("\nword tokenization\\n=====\\n")
for index in range(len(sents)):
    words = tokenize.word_tokenize(sents[index])
    print(words)

```

The screenshot shows a Google Colab interface with multiple tabs at the top. The active tab is 'NLP\_Practical 2.ipynb'. The code cell contains the provided NLTK tokenization code. Below the code, the output shows the download of 'punkt' and 'words' corpora, followed by the results of sentence and word tokenization.

```

import nltk
from nltk import tokenize
nltk.download('punkt')
nltk.download('words')
para = "Hello! My name is Sneha Pawar. Today we'll be learning NLTK. It is one of the powerful library of NLP"
sents = tokenize.sent_tokenize(para)
print("\nsentence tokenization\n=====\\n",sents)
# word tokenization
print("\nword tokenization\\n=====\\n")
for index in range(len(sents)):
    words = tokenize.word_tokenize(sents[index])
    print(words)

```

```

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data] Package words is already up-to-date!

sentence tokenization
=====
['Hello!', 'My', 'name', 'is', 'Sneha', 'Pawar', '.']
[Today, 'we', "'ll", 'be', 'learning', 'NLTK', '.']
['It', 'is', 'one', 'of', 'the', 'powerful', 'library', 'of', 'NLP']

word tokenization
=====
['Hello', '!']
['My', 'name', 'is', 'Sneha', 'Pawar', '.']
[Today, 'we', "'ll", 'be', 'learning', 'NLTK', '.']
['It', 'is', 'one', 'of', 'the', 'powerful', 'library', 'of', 'NLP']

```

## E. Write a program to find the most frequent noun tags

```

import nltk
nltk.download('averaged_perceptron_tagger')
from collections import defaultdict
text = nltk.word_tokenize("Kane Williamson is a kiwi cricketer whereas MS Dhoni is an
Indian cricketer..")
tagged = nltk.pos_tag(text)
print(tagged)
# checking if it is a noun or not
addNounWords = []
count=0
for words in tagged:
    val = tagged[count][1]
    if(val == 'NN' or val == 'NNS' or val == 'NNPS' or val == 'NNP'):
        addNounWords.append(tagged[count][0])
    count+=1
print (addNounWords)
temp = defaultdict(int)
# memoizing count

```

```

for sub in addNounWords:
    for wrd in sub.split():
        temp[wrd] += 1
# getting max frequency
res = max(temp, key=temp.get)
# printing result
print("Word with maximum frequency : " + str(res))

```

```

import nltk
nltk.download('averaged_perceptron_tagger')
from collections import defaultdict
text = nltk.word_tokenize("Kane Williamson is a kiwi cricketer whereas MS Dhoni is an Indian cricketer..")
tagged = nltk.pos_tag(text)
print(tagged)
# checking if it is a noun or not
addNounWords = []
count=0
for words in tagged:
    val = tagged[count][1]
    if(val == 'NN' or val == 'NNS' or val == 'NNPS' or val == 'NNP'):
        addNounWords.append(tagged[count][0])
    count+=1
print (addNounWords)
temp = defaultdict(int)
# memoizing count
for sub in addNounWords:
    for wrd in sub.split():
        temp[wrd] += 1
# getting max frequency
res = max(temp, key=temp.get)
# printing result
print("Word with maximum frequency : " + str(res))

```

[nltk\_data] Downloading package averaged\_perceptron\_tagger to  
[nltk\_data] /root/nltk\_data...  
[nltk\_data] Unzipping taggers/averaged\_perceptron\_tagger.zip.  
[('Kane', 'NNP'), ('Williamson', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('kiwi', 'NN'), ('cricketer', 'NN'), ('whereas', 'IN'), ('MS', 'NNP'), ('Dhoni', 'NNP'), ('is', 'VBD'), ('Kane', 'NNP'), ('Kane', 'NNP'), ('Kane', 'NNP'), ('Kane', 'NNP'), ('Kane', 'NNP'), ('Kane', 'NNP'), ('Kane', 'NNP')]  
Word with maximum frequency : Kane

Os completed at 12:28 PM

## F. Map Words to Properties using Python Dictionaries

```

#creating and printing a dictionay by mapping word with its properties
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}
print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))

```

The screenshot shows a Google Colab interface. The top bar has tabs for 'Welcome To Colaboratory', 'NLP\_Practical 1\_Covert Au', 'NLP\_Practical 2.ipynb - C...', 'Python: TypeError: object', 'What is Natural Language...', and a plus sign for a new notebook. Below the tabs is a search bar and a toolbar with icons for file operations, sharing, and settings.

The main area has a 'File' menu with options like File, Edit, View, Insert, Runtime, Tools, Help, and a status message 'All changes saved'. There are tabs for '+ Code' and '+ Text'. On the right, there are buttons for 'Comment', 'Share', 'Edit', and a profile icon. A sidebar on the left shows 'Files' with a tree view: a folder 'sample\_data' under a root folder '{x}'.

The central workspace contains a code cell titled 'F. Map Words to Properties using Python Dictionaries'. The code creates a dictionary mapping car properties to values:

```
#creating and printing a dictionary by mapping word with its properties
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}
print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))

{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
Ford
3
<class 'dict'>
```

The code cell has a play button icon and a progress bar indicating it completed at 12:28 PM. The bottom of the screen shows a taskbar with various icons and system status: 34°C, Smoke, ENG IN, 12:31 PM, 4/9/2022.

## G. Study i) default Tagger ii) Regular expression tagger iii) Unigram tagger i) Default Tagger

### i) Default Tagger

```
import nltk
nltk.download('treebank')
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
from nltk.corpus import treebank
testsentences = treebank.tagged_sents() [1000:]
print(exptagger.evaluate (testsentences))
#Tagging a list of sentences
import nltk
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
print(exptagger.tag_sents([('Hi', ' ', ' '), ('How', ' ', 'are', ' ', 'you', ' ', '?')]))
```

The screenshot shows a Google Colaboratory notebook titled "NLP\_Practical 2.ipynb". The code cell contains Python code to demonstrate a Default Tagger. The output shows the package being downloaded and unzipped, followed by the tagged sentences. The status bar at the bottom indicates the command completed at 12:28 PM.

```

import nltk
nltk.download('treebank')
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
from nltk.corpus import treebank
testsentences = treebank.tagged_sents() [1000:]
print(exptagger.evaluate (testsentences))
#Tagging a list of sentences
import nltk
from nltk.tag import DefaultTagger
exptagger = DefaultTagger('NN')
print(exptagger.tag_sents([['Hi', ','], ['How', 'are', 'you', '?']]))

[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]  Unzipping corpora/treebank.zip.
0.13198749536374715
[ [('Hi', 'NN'), (',', 'NN')], [('How', 'NN'), ('are', 'NN'), ('you', 'NN'), ('?', 'NN')]]
```

## ii) Regular expression tagger

```

from nltk.corpus import brown
from nltk.tag import RegexpTagger
test_sent = brown.sents(categories='news')[0]
regexp_tagger = RegexpTagger(
    [(r'^-?[0-9]+(.?[0-9]+)?$', 'CD'), # cardinal numbers
     (r'(The|the|A|a|An|an)$', 'AT'), # articles
     (r'.*able$', 'JJ'), # adjectives
     (r'.*ness$', 'NN'), # nouns formed from adjectives
     (r'.*ly$', 'RB'), # adverbs
     (r'.*s$', 'NNS'), # plural nouns
     (r'.*ing$', 'VBG'), # gerunds
     (r'.*ed$', 'VBD'), # past tense verbs
     (r'.*', 'NN') # nouns (default)
    ])
print(regexp_tagger)
print(regexp_tagger.tag(test_sent))
```

The screenshot shows a Google Colab notebook titled "NLP\_Practical 2.ipynb". The code cell contains Python code for a Regular expression tagger using the NLTK library. The code imports `nltk.corpus` and `nltk.tag`, and defines a `RegexTagger` with various regular expression patterns for different parts of speech. The output of the code execution shows the tagger's size and a list of tagged words from a test sentence. The Colab interface includes a toolbar with file operations, a sidebar with a code editor, and a status bar at the bottom.

```
from nltk.corpus import brown
from nltk.tag import RegexpTagger
test_sent = brown.sents(categories='news')[0]
regexp_tagger = RegexpTagger([
    (r'^-[0-9]+(.?[0-9]+)$', 'CD'), # cardinal numbers
    (r'^(The|the|A|a|An|an)$', 'AT'), # articles
    (r'.*able$', 'JJ'), # adjectives
    (r'.*ness$', 'NN'), # nouns formed from adjectives
    (r'.*ly$', 'RB'), # adverbs
    (r'.*s$', 'NNS'), # plural nouns
    (r'.*ing$', 'VBG'), # gerunds
    (r'.*ed$', 'VBD'), # past tense verbs
    (r'.*', 'NN') # nouns (default)
])
print(regexp_tagger)
print(regexp_tagger.tag(test_sent))
```

### iii) Unigram Tagger

```
# Loading Libraries
from nltk.tag import UnigramTagger
from nltk.corpus import treebank
# Training using first 10 tagged sentences of the treebank corpus as data.
# Using data
train_sents = treebank.tagged_sents() [:10]

# Initializing
tagger = UnigramTagger(train_sents)

# Lets see the first sentence
# (of the treebank corpus) as list
print(treebank.sents()[0])
print('\n',tagger.tag(treebank.sents()[0]))
#Finding the tagged results after training.
tagger.tag(treebank.sents()[0])
#Overriding the context model
tagger = UnigramTagger(model ={'Pierre': 'NN'})
print('\n',tagger.tag(treebank.sents()[0]))
```

```

  Welcome To Colaboratory | NLP_Practical 1_Covert Au | NLP_Practical 2.ipynb - C | Python: TypeError: object | What is Natural Language | + | - | X
  colab.research.google.com/drive/1MOzF6C-Ea-NvNiklyCES7_fvs95_E5rh?hl=en#scrollTo=U104JUlw-JsV
  colab.research.google.com/drive/1MOzF6C-Ea-NvNiklyCES7_fvs95_E5rh?hl=en#scrollTo=U104JUlw-JsV

  NLP_Practical 2.ipynb
  File Edit View Insert Runtime Tools Help All changes saved
  + Code + Text
  iii) Unigram Tagger
  [x] [28] # Loading Libraries
  from nltk.tag import UnigramTagger
  from nltk.corpus import treebank
  # Training using first 10 tagged sentences of the treebank corpus as data.
  # Using data
  train_sents = treebank.tagged_sents()[:10]

  # Initializing
  tagger = UnigramTagger(train_sents)

  # Lets see the first sentence
  # (of the treebank corpus) as list
  print(treebank.sents()[0])
  print("\n",tagger.tag(treebank.sents()[0]))
  #Finding the tagged results after training.
  tagger.tag(treebank.sents()[0])
  #Overriding the context model
  tagger = UnigramTagger(model ={'Pierre': 'NN'})
  print("\n",tagger.tag(treebank.sents()[0]))

  ['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the', 'board', 'as', 'a', 'nonexecutive', 'director', 'Nov.', '29', '.']
  [('Pierre', 'NNP'), ('Vinken', 'NNP'), ('.', ','), ('61', 'CD'), ('years', 'NNS'), ('old', 'JJ'), ('.', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT'), ('board', 'NN'), ('as', 'IN'), ('a', None), ('.', None), ('61', None), ('years', None), ('old', None), ('.', None), ('will', None), ('join', None), ('the', None), ('board', None), ('as', None), ('.', None)]

```

34°C Smoke 0s completed at 12:28 PM 12:35 PM 4/9/2022

#### H. Find different words from a given plain text without any space by comparing this text with a given corpus of words. Also find score of words.

```

from __future__ import with_statement #with statement for reading file
import re # Regular expression
words = [] # corpus file words
testword = [] # test words
ans = [] # words matches with corpus
print("MENU")
print("-----")
print(" 1 . Hash tag segmentation ")
print(" 2 . URL segmentation ")
print("enter the input choice for performing word segmentation")

```

The screenshot shows a Google Colaboratory notebook titled "NLP\_Practical 2.ipynb". The code cell contains Python code for performing word segmentation. The terminal output shows the code execution completed at 12:28 PM. The system tray indicates a temperature of 34°C and a battery level of 0%.

```
[77]: from __future__ import with_statement #with statement for reading file
import re # Regular expression
words = [] # corpus file words
testword = [] # test words
ans = [] # words matches with corpus
print("MENU")
print("-----")
print(" 1 . Hash tag segmentation ")
print(" 2 . URL segmentation ")
print("enter the input choice for performing word segmentation")
```

MENU  
-----  
1 . Hash tag segmentation  
2 . URL segmentation  
enter the input choice for performing word segmentation

✓ 0s completed at 12:28 PM

34°C Smoke ENG IN 12:36 PM 4/9/2022

```
choice = int(input())
if choice == 1:
    text = "#whatismyname" # hash tag test data to segment
    print("input with HashTag",text)
    pattern=re.compile("[^\w']")
    a = pattern.sub('', text)
elif choice == 2:
    text = "www.whatismyname.com" # url test data to segment
    print("input with URL",text)
    a=re.split('\s|(?<!\d)[,\.](?!\\d)', text)
    splitwords = ["www","com","in"] # remove the words which is containg in the list
    a = ''.join([each for each in a if each not in splitwords])
else:
    print("wrong choice...try again")
    print(a)
```

The screenshot shows a Google Colab notebook titled "NLP\_Practical 2.ipynb". The code cell contains Python code for text segmentation based on user input. The output shows the user input "1" followed by "input with HashTag #whatismyname". The status bar at the bottom indicates the cell completed at 12:28 PM. The system tray shows a temperature of 34°C and a battery level of 12:36 PM on 4/9/2022.

```
choice = int(input())
if choice == 1:
    text = "#whatismyname" # hash tag test data to segment
    print("input with HashTag",text)
    pattern=re.compile("[^w']")
    a = pattern.sub('', text)
elif choice == 2:
    text = "www.whatismyname.com" # url test data to segment
    print("input with URL",text)
    a=re.split('\s|(?<!\\d)[,\\.](?!\\d)', text)
    splitwords = ["www","com","in"] # remove the words which is containg in the list
    a = "".join([each for each in a if each not in splitwords])
else:
    print("wrong choice...try again")
    print(a)
```

```
choice = int(input())
if choice == 1:
    text = "#whatismyname" # hash tag test data to segment
    print("input with HashTag",text)
    pattern=re.compile("[^w']")
    a = pattern.sub('', text)
elif choice == 2:
    text = "www.whatismyname.com" # url test data to segment
    print("input with URL",text)
    a=re.split('\s|(?<!\\d)[,\\.](?!\\d)', text)
    splitwords = ["www","com","in"] # remove the words which is containg in the list
    a = "".join([each for each in a if each not in splitwords])
else:
    print("wrong choice...try again")
    print(a)
```

```

choice = int(input())
if choice == 1:
    text = "#whatismyname" # hash tag test data to segment
    print("input with HashTag",text)
    pattern=re.compile("[^\w']")
    a = pattern.sub('', text)
elif choice == 2:
    text = "www.whatismyname.com" # url test data to segment
    print("input with URL",text)
    a=re.split('s|(?<!d)[.,](?!\d)', text)
    splitwords = ["www","com","in"] # remove the words which is containg in the list
    a = "".join([each for each in a if each not in splitwords])
else:
    print("wrong choice...try again")
    print(a)

```

```

1
input with HashTag #whatismyname

```

```

[80] for each in a:

```

0s completed at 12:28 PM

34°C Smoke ENG IN 12:37 PM 4/9/2022

```

for each in a:
    testword.append(each) #test word
test_lenth = len(testword) # lenght of the test data
# Reading the corpus
with open('/SampleText.txt', 'r') as f:
    lines = f.readlines()
words =[e.strip() for e in lines]
def Seg(a,lenth):
    ans = []
    for k in range(0,lenth+1): # this loop checks char by char in the corpus
        if a[0:k] in words:
            print(a[0:k],"-appears in the corpus")
            ans.append(a[0:k])
            break
    if ans != []:
        g = max(ans,key=len)
        return g
test_tot_itr = 0 #each iteration value
answer = [] # Store the each word contains the corpus
Score = 0
N = 37 # total no of corpus
M = 0
C = 0
while test_tot_itr < test_lenth:
    ans_words = Seg(a,test_lenth)
    if ans_words != 0:
        test_itr = len(ans_words)
        answer.append(ans_words)
        a = a[test_itr:test_lenth]
        test_tot_itr += test_itr
Aft_Seg = " ".join([each for each in answer])

```

```

# print segmented words in the list
print("output")
print("-----")
print(Aft_Seg) # print After segmentation the input
# Calculating Score
C = len(answer)
score = C * N / N # Calculate the score
print("Score",score)

```

The screenshot shows a Google Colab notebook titled "NLP\_Practical 2.ipynb". The code cell contains the provided Python script. The output pane shows the execution results:

```

what -appears in the corpus
is -appears in the corpus
my -appears in the corpus
name -appears in the corpus
output
-----
what is my name
Score 4.0

```

The status bar at the bottom indicates the notebook was completed at 12:28 PM on 4/9/2022.

what -appears in the corpus  
 is -appears in the corpus  
 my -appears in the corpus  
 name -appears in the corpus  
 output  
 -----  
 what is my name  
 Score 4.0

# Practical No. 03

## A) Study of WordNet dictionary with methods as synsets, definitions, examples and antonyms

```
import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet
print(wordnet.synsets("computer"))
# definition and example of the word 'computer'
print(wordnet.synset("computer.n.01").definition())
#examples
print("Examples:", wordnet.synset("computer.n.01").examples())
#get Antonyms
print(wordnet.lemma('buy.v.01.buy').antonyms())
```

NLP Practical 3.ipynb - Colaboratory

NLP Practical 3.ipynb - Colaboratory

colab.research.google.com/drive/1WHqUNQAW0D7LM6cgTX3KXFnSSYVEKqoz#scrollTo=lw9m\_GExzPNk

NLP Practical 3.ipynb

File Edit View Insert Runtime Tools Help Last saved at 3:56 PM

Comment Share Settings User

+ Code + Text

RAM Disk Editing

A) Study of WordNet dictionary with methods as synsets, definitions, examples and antonyms

```
import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet
print(wordnet.synsets("computer"))
# definition and example of the word 'computer'
print(wordnet.synset("computer.n.01").definition())
#examples
print("Examples:", wordnet.synset("computer.n.01").examples())
#get Antonyms
print(wordnet.lemma('buy.v.01.buy').antonyms())
```

[nltk\_data] Downloading package wordnet to /root/nltk\_data...
[nltk\_data] Unzipping corpora/wordnet.zip.
[Synset('computer.n.01'), Synset('calculator.n.01')]
a machine for performing calculations automatically
Examples: []
[Lemma('sell.v.01.sell')]

B. Study lemmas, hyponyms and hypernyms

32°C Haze

ENG IN 3:57 PM 4/9/2022

## B. Study lemmas, hyponyms and hypernyms

```
import nltk
nltk.download('wordnet')
```

```
from nltk.corpus import wordnet
print(wordnet.synsets("computer"))
print(wordnet.synset("computer.n.01").lemma_names())
#all lemmas for each synset.
for e in wordnet.synsets("computer"):
    print(f'{e} --> {e.lemma_names() }')
#print all lemmas for a given synset
print(wordnet.synset('computer.n.01').lemmas())
#get the synset corresponding to lemma
print(wordnet.lemma('computer.n.01.computing_device').synset())
#Get the name of the lemma
print(wordnet.lemma('computer.n.01.computing_device').name())
#Hyponyms give abstract concepts of the word that are much more specific
#the list of hyponyms words of the computer
syn = wordnet.synset('computer.n.01')
print(syn.hyponyms())
print([lemma.name() for synset in syn.hyponyms() for lemma in synset.lemmas()])
#the semantic similarity in WordNet
vehicle = wordnet.synset('vehicle.n.01')
car = wordnet.synset('car.n.01')
print(car.lowest_common_hypernyms(vehicle))
```

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** NLP Practical 3.ipynb - Colaboratory
- URL:** colab.research.google.com/drive/1WHqUNQAW0D7LM6cgTX3KXFnSSYVEKqoz#scrollTo=2ZsD1SCj0ruT
- Header:** NLP Practical 3.ipynb, File, Edit, View, Insert, Runtime, Tools, Help, Last saved at 3:56 PM, Comment, Share, RAM Disk, Editing.
- Code Cell:** B. Study lemmas, hyponyms and hypernyms
- Code Content:**

```
import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet
print(wordnet.synsets("computer"))
print(wordnet.synset("computer.n.01").lemma_names())
#all lemmas for each synset.
for e in wordnet.synsets("compute"):
    print(f'{e} --> {e.lemma_names()}' )
#print all lemmas for a given synset
print(wordnet.synset('computer.n.01').lemmas())
#get the synset corresponding to lemma
print(wordnet.lemma('computer.n.01.computing_device').synset())
#Get the name of the lemma
print(wordnet.lemma('computer.n.01.computing_device').name())
#Hyponyms give abstract concepts of the word that are much more specific
#the list of hyponyms words of the computer
syn = wordnet.synset('computer.n.01')
print(syn.hyponyms())
print([lemma.name() for synset in syn.hyponyms() for lemma in synset.lemmas()])
#the semantic similarity in WordNet
vehicle = wordnet.synset('vehicle.n.01')
car = wordnet.synset('car.n.01')
print(car.lowest_common_hypernyms(vehicle))
```
- Output Cell:** [nltk\_data] Downloading package wordnet to /root/nltk\_data...  
[nltk\_data] Package wordnet is already up-to-date!  
[Synset('computer.n.01'), Synset('calculator.n.01')]  
['computer', 'computing\_machine', 'computing\_device', 'data\_processor', 'electronic\_computer', 'information\_processing\_system']  
Synset('computer.n.01') --> ['computer', 'computing\_machine', 'computing\_device', 'data\_processor', 'electronic\_computer', 'information\_processing\_system']  
Synset('calculator.n.01') --> ['calculator', 'reckoner', 'figurine', 'estimator', 'computer']  
[Lemma('computer.n.01.computer'), Lemma('computer.n.01.computing\_machine'), Lemma('computer.n.01.computing\_device'), Lemma('computer.n.01.data\_processor'), Lemma('computer.n.01.electronic\_computer'), Lemma('computer.n.01.information\_processing\_system')]  
Synset('computer.n.01')  
computing\_device  
<bound method \_WordNetObject.hyponyms of Synset('computer.n.01')>  
['analog\_computer', 'analogue\_computer', 'digital\_computer', 'home\_computer', 'node', 'client', 'guest', 'number\_cruncher', 'pari-mutuel\_machine', 'totalizer', 'totaliser', 'totalizator', 'totalisator', 'predictor', 'server', 'host', [Synset('vehicle.n.01')]]
- Bottom Bar:** 32°C, Haze, ENG IN, 3:58 PM, 4/9/2022

C. Write a program using Python to find synonyms, and antonyms of word “active” using wordnet

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

```
[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('active.s.04'), Synset('active.a.05'), Synset('active.a.06'), Synset('active.s.05'), Lemma('inactive.a.02.inactive')]
```

## D. Compare two nouns

```
#Using NLTK adding or removing stopwords
from nltk.corpus import wordnet
syn1 = wordnet.synsets('football')
syn2 = wordnet.synsets('soccer')
# A word may have multiple synsets, so need to compare each synset of word1 with syns
et of word2
for s1 in syn1:
    for s2 in syn2:
        print("Path similarity of: ")
        print(s1, '(', s1.pos(), ')', '[', s1.definition(), ']')
        print(s2, '(', s2.pos(), ')', '[', s2.definition(), ']')
        print(" is", s1.path_similarity(s2))
    print()
```

```
Path similarity of:
Synset('football.n.01') ( n ) [ any of various games played with a ball (round or oval) in which two teams try to kick or carry or propel the ball into each other's goal ]
Synset('soccer.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 American football ]
Synset('soccer.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 0.05
```

## E. Handling Stopwords

### i) Using nltk adding or removing stopwords in nltk stop wod list

```
import nltk
nltk.download('punkt')
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]  Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]  Unzipping corpora/stopwords.zip.
```

```
text = "Yashesh likes to play football, however he is not too fond of tennis."
text_tokens = word_tokenize(text)
```

```
tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]
print(tokens without sw)
```

```
['Yashesh', 'likes', 'play', 'football', ',', 'however', 'fond', 'tennis', '.']
```

```
#add the word play to the NLTK stop word collection
all_stopwords = stopwords.words('english')
all_stopwords.append('play')
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
print(tokens_without_sw)
```

```
#remove 'not' from stop word collection
all_stopwords.remove('not')
text_tokens = word_tokenize(text)
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
print(tokens without sw)
```

```
['Yashesh', 'likes', 'football', ',', 'however', 'not', 'fond', 'tennis', '.']
```

## ii) Using Gensim adding and removing stop words in default Gensim stop words lists

```
pip install gensim
```

```
import gensim
from gensim.parsing.preprocessing import remove_stopwords
text = "Yashesh likes to play football, however he is not too fond of tennis."
filtered_sentence = remove_stopwords(text)
print(filtered_sentence)
all_stopwords = gensim.parsing.preprocessing.STOPWORDS
print(all_stopwords)
```

Yashesh likes play football, fond tennis.  
frozensest({'meanwhile', 'these', 'fill', 'made', 'few', 'your', 'someone', 'eleven', 'whole', 'anyhow', 'beforehand', 'perhaps', 'per', 'whenever', 'only', 'call', 'six', 'fifteen', 'but', 'herself', 'anywhere', 'hereupon', 'fifty', 'a'})

```
from gensim.parsing.preprocessing import STOPWORDS
all_stopwords_gensim = STOPWORDS.union(set(['likes', 'play']))
```

```
text = "Yashesh likes to play football, however he is not too fond of tennis."  
text_tokens = word_tokenize(text)  
tokens_without_sw = [word for word in text_tokens if not word in  
all_stopwords_gensim]  
print(tokens_without_sw)
```

```
['Yashesh', 'football', ',', 'fond', 'tennis', '.']
```

```
from gensim.parsing.preprocessing import STOPWORDS  
all_stopwords_gensim = STOPWORDS  
sw_list = {"not"}  
all_stopwords_gensim = STOPWORDS.difference(sw_list)  
text = "Yashesh likes to play football, however he is not too fond of tennis."  
text_tokens = word_tokenize(text)  
tokens_without_sw = [word for word in text_tokens if not word in  
all_stopwords_gensim]  
print(tokens_without_sw)
```

```
['Yashesh', 'likes', 'play', 'football', ',', 'not', 'fond', 'tennis', '.']
```

### iii) Using Spacy adding and removing stop words in Default spacy stop word list

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

```
import spacy  
import nltk  
from nltk.tokenize import word_tokenize  
sp = spacy.load('en_core_web_sm')  
#add the word play to the NLTK stop word collection  
all_stopwords = sp.Defaults.stop_words  
all_stopwords.add("play")  
text = "Yashesh likes to play football, however he is not too fond of tennis."  
text_tokens = word_tokenize(text)  
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]  
print(tokens_without_sw)
```

```
[Yashesh', 'likes', 'football', ',', 'not', 'fond', 'tennis', '.']
```

```
#remove 'not' from stop word collection
all_stopwords.remove('not')
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
print(tokens_without_sw)
```

```
[Yashesh', 'likes', 'football', ',', 'not', 'fond', 'tennis', '.']
```

## Practical No. 04

### Text Tokenization

#### A) Tokenization using Python split() function

```

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)

```

The screenshot shows a Jupyter Notebook interface. The title bar says 'NLP Practical 4 - Text Tokenization.ipynb'. The notebook contains a single cell with the following Python code:

```

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)

```

The output of the code is displayed below the code cell, showing the tokens:

```

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

```

## B. Tokenization using RegEx function

```

import nltk
# import RegexpTokenizer() method from nltk
from nltk.tokenize import RegexpTokenizer
# Create a reference variable for Class RegexpTokenizer
tk = RegexpTokenizer('\s+', gaps = True)
# Create a string input
str = "I love to study Natural Language Processing in Python"

# Use tokenize method
tokens = tk.tokenize(str)

```

```
print(tokens)
```

#### B. Tokenization using RegEx function

```
import nltk
# import RegexpTokenizer() method from nltk
from nltk.tokenize import RegexpTokenizer
# Create a reference variable for Class RegexpTokenizer
tk = RegexpTokenizer('\s+', gaps = True)
# Create a string input
str = "I love to study Natural Language Processing in Python"
# Use tokenize method
tokens = tk.tokenize(str)
print(tokens)
```

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

## C. Tokenization using NLTK

---

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

# Create a string input
str = "I love to study Natural Language Processing in Python"

# Use tokenize method
print(word_tokenize(str))
```

#### C. Tokenization using NLTK

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

# Create a string input
str = "I love to study Natural Language Processing in Python"
# Use tokenize method
print(word_tokenize(str))

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]  Unzipping tokenizers/punkt.zip.
['I', 'love', 'to', 'study', 'Natural', 'Language', 'Processing', 'in', 'Python']
```

## D. Tokenization using spaCy library

```
import spacy
nlp = spacy.blank("en")
# Create a string input
str = "I love to study Natural Language Processing in Python"
# Create an instance of document;
# doc object is a container for a sequence of Token objects.
doc = nlp(str)
# Read the words; Print the words
#
words = [word.text for word in doc]
```

```
print(words)
```

#### D. Tokenization using spaCy library



```
import spacy
nlp = spacy.blank("en")
# Create a string input
str = "I love to study Natural Language Processing in Python"
# Create an instance of document;
# doc object is a container for a sequence of Token objects.
doc = nlp(str)
# Read the words; Print the words
#
words = [word.text for word in doc]
print(words)

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

## E. Tokenization using Keras

```
!pip install keras
!pip install tensorflow
```

```
import keras
from keras.preprocessing.text import text_to_word_sequence
#creating a string input
str = "I love to study Natural Language Processing in Python"
# tokenizing the text
tokens = text_to_word_sequence(str)
print(tokens)
```



```
import keras
from keras.preprocessing.text import text_to_word_sequence
#creating a string input
str = "I love to study Natural Language Processing in Python"
# tokenizing the text
tokens = text_to_word_sequence(str)
print(tokens)

['i', 'love', 'to', 'study', 'natural', 'language', 'processing', 'in', 'python']
```

## F. Tokenization using Gensim

```
!pip install gensim
from gensim.utils import tokenize
# Create a string input
str = "I love to study Natural Language Processing in Python"
# tokenizing the text
list(tokenize(str))
```

```
✓ 1s pip install gensim
from gensim.utils import tokenize
# Create a string input
str = "I love to study Natural Language Processing in Python"
# tokenizing the text
list(tokenize(str))

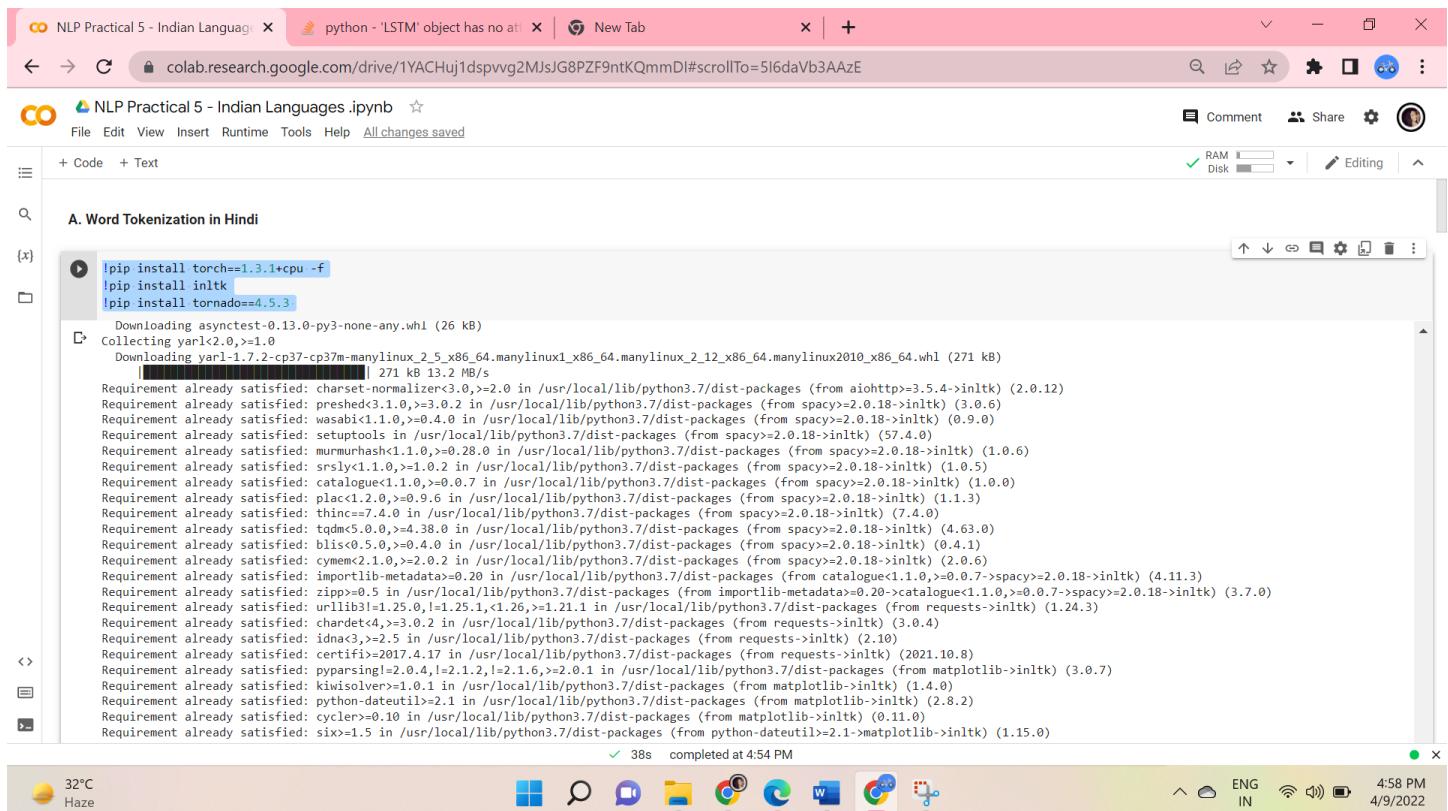
Requirement already satisfied: gensim in /usr/local/lib/python3.7/dist-packages (3.6.0)
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist-packages (from gensim) (5.2.1)
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.7/dist-packages (from gensim) (1.21.5)
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.7/dist-packages (from gensim) (1.4.1)
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.7/dist-packages (from gensim) (1.15.0)
['I',
 'love',
 'to',
 'study',
 'Natural',
 'Language',
 'Processing',
 'in',
 'Python']
```

## Practical No. 05

# Import NLP Libraries for Indian Languages and perform

## A. Word Tokenization in Hindi

```
!pip install torch==1.3.1+cpu -f  
!pip install inltk  
!pip install tornado==4.5.3
```



```
!pip install torch==1.3.1+cpu -f  
!pip install inltk  
!pip install tornado==4.5.3
```

Downloading asynctest-0.13.0-py3-none-any.whl (26 kB)  
Collecting yaml<2.0,>=1.0  
 Downloading yaml-1.7.2-cp37-cp37m-manylinux1\_x86\_64.manylinux1\_x86\_64.manylinux2010\_x86\_64.whl (271 kB)  
 [██████████] 271 kB 13.2 MB/s

Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp>3.5.4->inltk) (2.0.12)  
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (3.0.6)  
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (0.9.0)  
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (57.4.0)  
Requirement already satisfied: nummurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (1.0.6)  
Requirement already satisfied: srslv<1.1.0,>=1.0.2 in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (1.0.5)  
Requirement already satisfied: catalogue<1.1.0,>=0.7 in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (1.0.0)  
Requirement already satisfied: plac<1.2.0,>=0.9.6 in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (1.1.3)  
Requirement already satisfied: thinc<7.4.0,>=4.38.0 in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (7.4.0)  
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (4.63.0)  
Requirement already satisfied: blis<0.5.0,>=0.4.0 in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (0.4.1)  
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from spacy>=2.0.18->inltk) (2.0.6)  
Requirement already satisfied: importlib\_metadata<0.2. in /usr/local/lib/python3.7/dist-packages (from catalogue<1.1.0,>=0.7->spacy>=2.0.18->inltk) (4.11.3)  
Requirement already satisfied: zipp=<0.5. in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=0.20->catalogue<1.1.0,>=0.7->spacy>=2.0.18->inltk) (3.7.0)  
Requirement already satisfied: urllib3<1.25.0,>=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->inltk) (1.24.3)  
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->inltk) (3.0.4)  
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->inltk) (2.10)  
Requirement already satisfied: certifi<2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->inltk) (2021.10.8)  
Requirement already satisfied: pyparsing!=2.0.4,>=2.1.2,<2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->inltk) (3.0.7)  
Requirement already satisfied: kiwisolver<=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->inltk) (1.4.0)  
Requirement already satisfied: python-dateutil<=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->inltk) (2.8.2)  
Requirement already satisfied: cyclere<0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->inltk) (0.11.0)  
Requirement already satisfied: six<=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib->inltk) (1.15.0)

✓ 38s completed at 4:54 PM

32°C Haze ENG IN 4:58 PM 4/9/2022

```
from inltk.inltk import setup  
setup('hi')
```

```
from inltk.inltk import tokenize  
hindi_text = """प्राकृतिक भाषा सीखना बहुत दिलचस्प है। """  
# tokenize(input text, language code)  
tokenize(hindi_text, "hi")
```

The screenshot shows a Google Colab notebook titled "NLP Practical 5 - Indian Languages.ipynb". The notebook has three tabs open: "NLP Practical 5 - Indian Languages.ipynb", "python - 'LSTM' object has no attribute 'rnn\_is\_cuda'", and "New Tab". The main code cell contains the following Python code:

```
from inltk.inltk import setup
setup('hi')

from inltk.inltk import tokenize
hindi_text = """प्राक्तिक भाषा सीखना बहुत दिलचस्प है।"""
# tokenize(input text, language code)
tokenize(hindi_text, "hi")

['प्रा', 'कि', 'तिक', 'भाषा', 'सीखना', 'बहुत', 'दिलचस्प', 'है', '!']
```

The output of the code shows the tokens for the Hindi sentence "प्राक्तिक भाषा सीखना बहुत दिलचस्प है!".

## B) Generate similar sentences from a given Hindi text input

```
pip install torch==1.2.0+cu92 torchvision==0.4.0+cu92 -f https://download.pytorch.org/whl/torch_stable.html

from inltk.inltk import get_similar_sentences
# get similar sentences to the one given in hindi
output = get_similar_sentences('Mai aaj bahut khush hu.', 5, 'hi')
print(output)

!pip install torch==1.3.1+cpu -f https://download.pytorch.org/whl/torch_stable.html
!pip install inltk
!pip install tornado==4.5.3

from inltk.inltk import setup
setup('hi')
from inltk.inltk import get_similar_sentences
# get similar sentences to the one given in hindi
output = get_similar_sentences('मैंआज बहुत खुश हूं', 5, 'hi')
print(output)
```

NLP Practical 5 - Indian Languages x python - 'LSTM' object has no attribute x New Tab x | +

colab.research.google.com/drive/1YACHuj1dspvg2MJsJG8PZF9ntKQmmDI#scrollTo=lbxYMGpZBdl1

**NLP Practical 5 - Indian Languages.ipynb**

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] pip install torch==1.2.0+cu92 torchvision==0.4.0+cu92 -f https://download.pytorch.org/whl/torch_stable.html
Looking in links: https://download.pytorch.org/whl/torch_stable.html
Collecting torch==1.2.0+cu92
  Downloading https://download.pytorch.org/whl/cu92/torch-1.2.0%2Bcu92-cp37-cp37m-manylinux1_x86_64.whl (663.1 MB)
[ ] 663.1 MB 1.7 kB/s
Collecting torchvision==0.4.0+cu92
  Downloading https://download.pytorch.org/whl/cu92/torchvision-0.4.0%2Bcu92-cp37-cp37m-manylinux1_x86_64.whl (8.8 MB)
[ ] 8.8 MB 2.2 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from torch==1.2.0+cu92) (1.21.5)
Requirement already satisfied: pillow==4.1.1 in /usr/local/lib/python3.7/dist-packages (from torchvision==0.4.0+cu92) (7.1.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from torchvision==0.4.0+cu92) (1.15.0)
Installing collected packages: torch, torchvision
Attempting uninstall: torch
  Found existing installation: torch 1.10.0+cu111
  Uninstalling torch-1.10.0+cu111:
    Successfully uninstalled torch-1.10.0+cu111
Attempting uninstall: torchvision
  Found existing installation: torchvision 0.11.1+cu111
  Uninstalling torchvision-0.11.1+cu111:
    Successfully uninstalled torchvision-0.11.1+cu111
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
torchtext 0.11.0 requires torch==1.10.0, but you have torch 1.2.0+cu92 which is incompatible.
torchaudio 0.10.0+cu111 requires torch==1.10.0, but you have torch 1.2.0+cu92 which is incompatible.
Successfully installed torch-1.2.0+cu92 torchvision-0.4.0+cu92
```

```
from inltk.inltk import get_similar_sentences
# get similar sentences to the one given in hindi
output = get_similar_sentences('Mai aaj bahut khush hu.', 5, 'hi')
print(output)
```

✓ 38s completed at 4:54 PM

32°C Haze ENG IN 5:00 PM 4/9/2022

NLP Practical 5 - Indian Languages x python - 'LSTM' object has no attribute x New Tab x | +

colab.research.google.com/drive/1YACHuj1dspvg2MJsJG8PZF9ntKQmmDI#scrollTo=W5kArgwQC5tV

**NLP Practical 5 - Indian Languages.ipynb**

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.loss.CrossEntropyLoss' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/torch/nn/modules/loss.html#CrossEntropyLoss
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.WeightDropout' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/fastai/text/models/awd_lstm.html#WeightDropout
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.rnn.LSTM' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/torch/rnn.html#LSTM
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.RNNDropout' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/fastai/text/models/awd_lstm.html#RNNDropout
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.LinearDecoder' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/fastai/text/models/awd_lstm.html#LinearDecoder
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.linear.Linear' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/torch/nn/modules/linear.html#Linear
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.loss.CrossEntropyLoss' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/torch/nn/modules/loss.html#CrossEntropyLoss
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.AWD_LSTM' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/fastai/text/models/awd_lstm.html#AWD_LSTM
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.sparse.Embedding' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/torch/nn/modules/sparse.html#Embedding
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.EmbeddingDropout' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/fastai/text/models/awd_lstm.html#EmbeddingDropout
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.container.ModuleList' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/torch/nn/modules/container.html#ModuleList
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.WeightDropout' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/fastai/text/models/awd_lstm.html#WeightDropout
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.rnn.LSTM' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/torch/rnn.html#LSTM
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.RNNDropout' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/fastai/text/models/awd_lstm.html#RNNDropout
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.LinearDecoder' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/fastai/text/models/awd_lstm.html#LinearDecoder
  warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.linear.Linear' has changed. you can retrieve the original source code at https://pytorch.org/docs/stable/_modules/torch/nn/modules/linear.html#Linear
  warnings.warn(msg, SourceChangeWarning)
[‘मैंआज बहुत खुश हूं’, ‘मैंअमृत बहुत खुश हूं’, ‘मैंआज उल्टि खुश हूं’, ‘मैंआज बहुत नाखुश हूं’]
```

✓ 38s completed at 4:54 PM

32°C Haze ENG IN 5:00 PM 4/9/2022

### C) Identify the Indian language from a text

```
!pip install torch==1.3.1+cpu -f https://download.pytorch.org/whl/torch_stable.html
```

```

from inltk.inltk import setup
setup('gu')
from inltk.inltk import identify_language
#Identify the language of given text
identify_language('અર્થાત કાપડિયા')

```

gujarati

The screenshot shows a Google Colab notebook titled "NLP Practical 5 - Indian Languages.ipynb". The code cell contains the following Python code:

```

from inltk.inltk import setup
setup('gu')
from inltk.inltk import identify_language
#Identify the language of given text
identify_language('ગુજરાતી કાપડિયા')

```

The output of the cell shows the following message:

```

Download Model. This might take time, depending on your internet connection. Please be patient.
We'll only do this for the first time.
Download Model. This might take time, depending on your internet connection. Please be patient.
We'll only do this for the first time.
Done!
Download Model. This might take time, depending on your internet connection. Please be patient.
We'll only do this for the first time.
Download Model. This might take time, depending on your internet connection. Please be patient.
We'll only do this for the first time.
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.loss.CrossEntropyLoss' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.learner.MultiBatchEncoder' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.AWD_LSTM' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.sparse.Embedding' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.EmbeddingDropout' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.container.ModuleList' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.WeightDropout' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.rnn.LSTM' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.models.awd_lstm.RNNDropout' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'fastai.text.learner.PoolingLinearClassifier' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.batchnorm.BatchNorm2d' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.dropout.Dropout' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.linear.Linear' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
/usr/local/lib/python3.7/dist-packages/torch/serialization.py:453: SourceChangeWarning: source code of class 'torch.nn.modules.activation.ReLU' has changed. you can retrieve the original source code by accessing the object's source attribute.
warnings.warn(msg, SourceChangeWarning)
'gujarati'

```

The status bar at the bottom indicates the cell completed at 4:54 PM.

### C) Named Entity recognition with diagram using NLTK corpus – treebank. Source code:

Note: It runs on Python IDLE

```

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()

```

Named Entity recognition with diagram using NLTK corpus – treebank.py - C:/Users/Sneha/OneDrive/Desktop/NLP Practical/NLP Practical 6/Named Entity recognition with diagram using NLTK corpus – treebank...

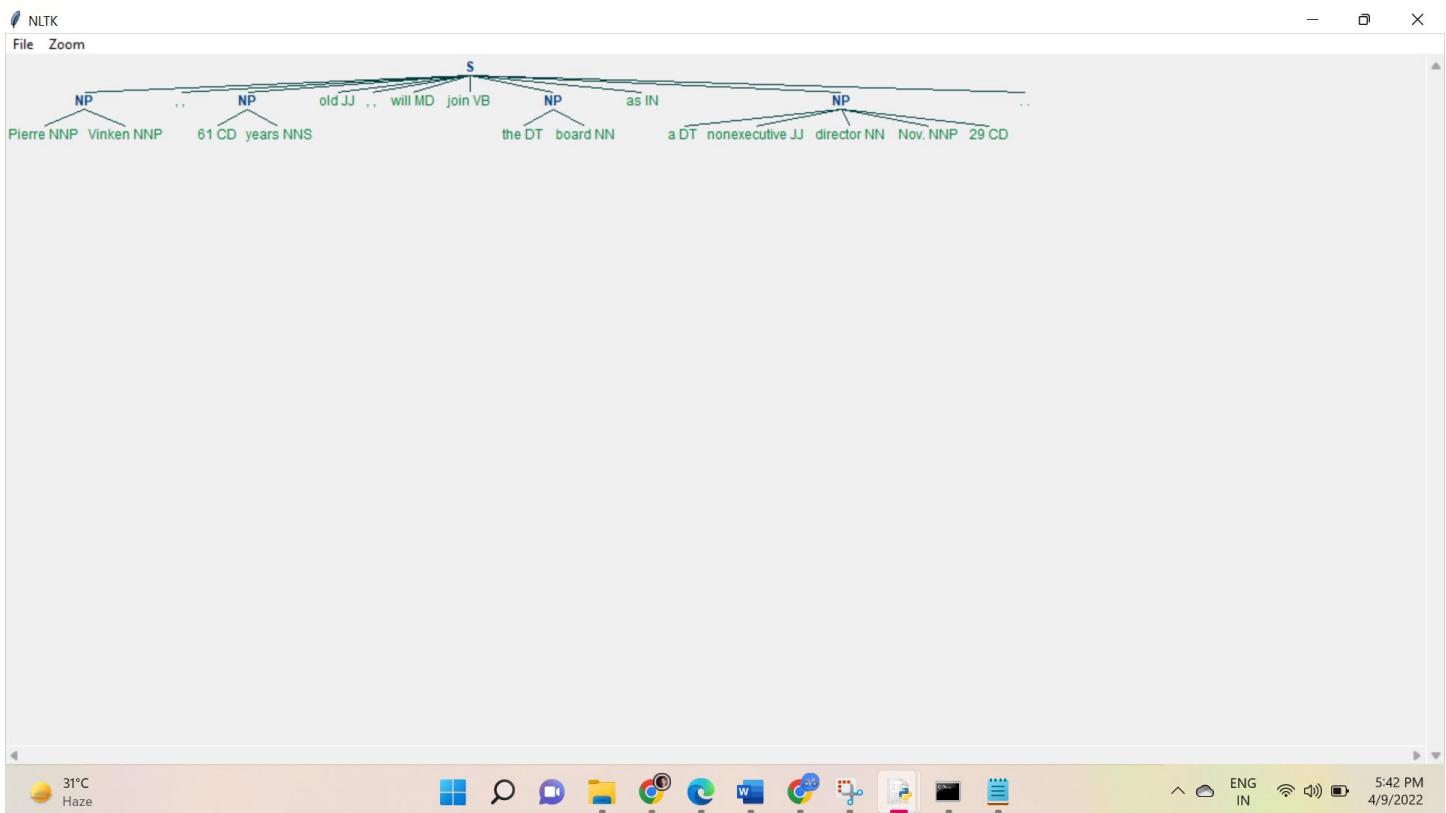
```
File Edit Format Run Options Window Help
import nltk
nltk.download('treebank')
from nltk.corpus import treebank_chunk
treebank_chunk.tagged_sents()[0]
treebank_chunk.chunked_sents()[0]
treebank_chunk.chunked_sents()[0].draw()
```

Ln: 7 Col: 0

31°C Haze

5:41 PM 4/9/2022

```
>>>
= RESTART: C:/Users/Sneha/OneDrive/Desktop/NLP Practical/NLP Practical 6/Named Entity recognition with diagram using NLTK corpus – treebank.py
[nltk_data]  Downloading package treebank to
[nltk_data]    C:\Users\Sneha\AppData\Roaming\nltk_data...
[nltk_data]    Unzipping corpora\treebank.zip.
```



# Practical No. 07

## Finite State Automata

A) Define grammar using nltk. Analyze a sentence using the same.

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

NLP Practical 7-A.py - C:/Users/Sneha/OneDrive/Desktop/NLP Practical/NLP Practical 7/NLP Practical 7-A.py (3.7.8)

File Edit Format Run Options Window Help

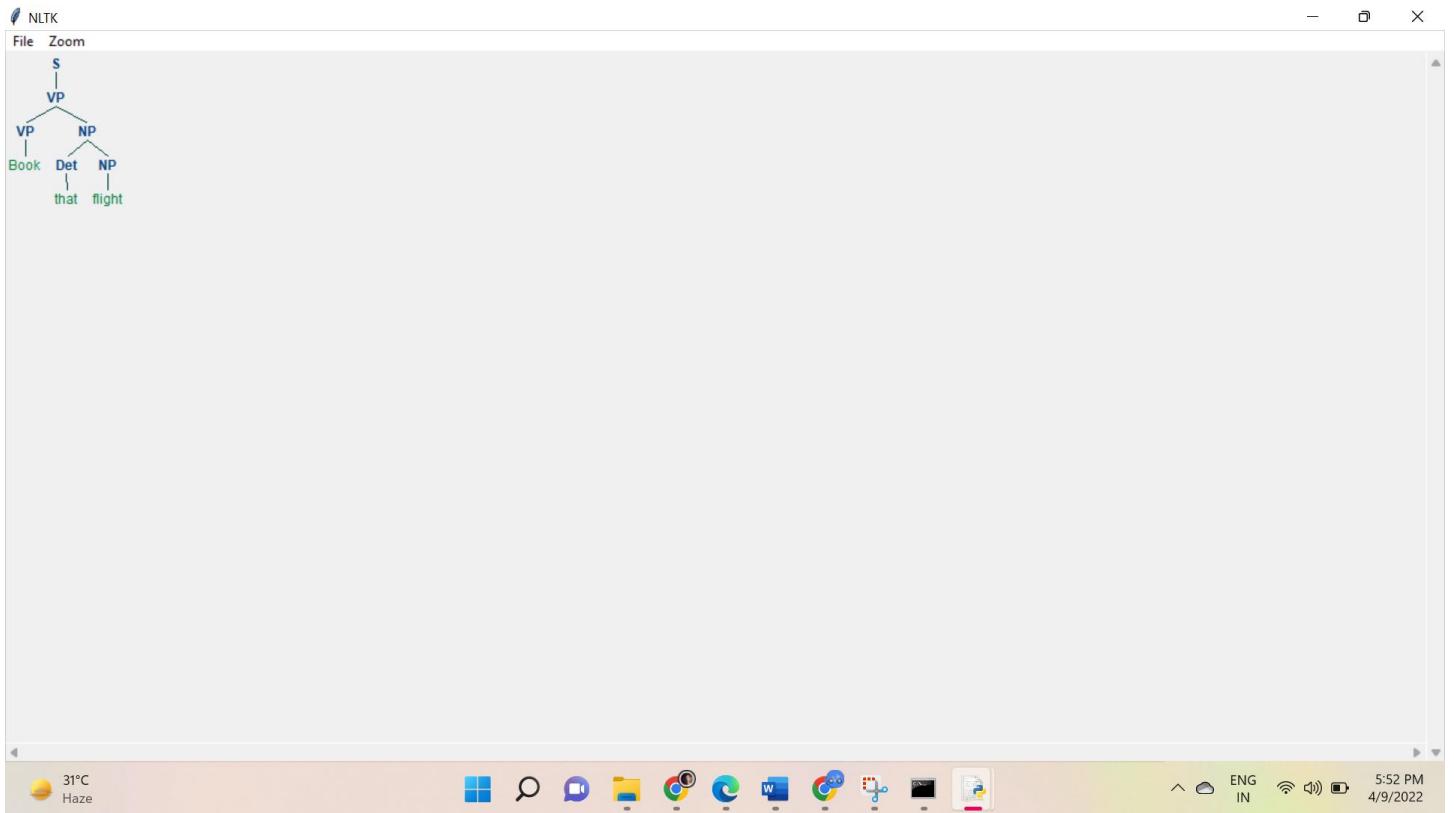
```
import nltk
nltk.download('punkt')
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
S -> VP
VP -> VP NP
NP -> Det NP
Det -> 'that'
NP -> singular Noun
NP -> 'flight'
VP -> 'Book'
""")
sentence = "Book that flight"
for index in range(len(sentence)):
    all_tokens = tokenize.word_tokenize(sentence)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

Ln: 14 Col: 34

31°C Haze

Windows Start button, Taskbar icons (File Explorer, Edge, Google Chrome, File Explorer, File Explorer, File Explorer, File Explorer, File Explorer, File Explorer, File Explorer), Network icon, ENG IN, WiFi icon, 5:52 PM, 4/9/2022

```
>>>
=====
RESTART: C:/Users/Sneha/OneDrive/Desktop/NLP Practical/NLP Practical 7/NLP Practical 7-A.py =====
[nltk_data]  Downloading package punkt to
[nltk_data]      C:\Users\Sneha\AppData\Roaming\nltk_data...
[nltk_data]  Unzipping tokenizers\punkt.zip.
['Book', 'that', 'flight']
(S (VP (VP Book) (NP (Det that) (NP flight))))
```



## B) Accept the input string with regular expression of finite automation: 101+

```
def FA(s):
    #if the length is less than 3 then it can't be accepted, Therefore end the process.
    if len(s)<3:
        return "Rejected"
    #first three characters are fixed. Therefore, checking them using index
    if s[0]=='1':
        if s[1]=='0':
            if s[2]=='1':
                # After index 2 only "1" can appear. Therefore break the process if any other character is detected
                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(FA(i))
```

The screenshot shows a Google Colab notebook titled 'NLP Practical 7 Finite State Automata.ipynb'. The code cell contains the same Python function as above. Below the code, the output shows the results for each input string: 'Rejected', 'Rejected', 'None', 'None', 'None', 'None', 'Rejected', 'Rejected', and 'None'. The Colab interface includes tabs for other notebooks and a toolbar with various icons.

## C. Accept the input string with Regular expression of FA: $(a+b)^*bba$ .

```

def FA(s):
    size=0
    #scan complete string and make sure that it contains only 'a' & 'b'
    for i in s:
        if i=='a' or i=='b':
            size+=1
        else:
            return "Rejected"
    #After checking that it contains only 'a' & 'b'
    #check it's length it should be 3 atleast
    if size>=3:
        #check the last 3 elements
        if s[size-3]=='b':
            if s[size-2]=='b':
                if s[size-1]=='a':
                    return "Accepted"
                return "Rejected"
            return "Rejected"
        return "Rejected"
    return "Rejected"
inputs=['bba', 'ababbba', 'abba', 'abb', 'baba', 'bbb', '']
for i in inputs:
    print(FA(i))

```

NLP Practical 7 Finite State Automata.ipynb

C. Accept the input string with Regular expression of FA:  $(a+b)^*bba$ .

```

def FA(s):
    size=0
    #scan complete string and make sure that it contains only 'a' & 'b'
    for i in s:
        if i=='a' or i=='b':
            size+=1
        else:
            return "Rejected"
    #After checking that it contains only 'a' & 'b'
    #check it's length it should be 3 atleast
    if size>=3:
        #check the last 3 elements
        if s[size-3]=='b':
            if s[size-2]=='b':
                if s[size-1]=='a':
                    return "Accepted"
                return "Rejected"
            return "Rejected"
        return "Rejected"
    return "Rejected"
inputs=['bba', 'ababbba', 'abba', 'abb', 'baba', 'bbb', '']
for i in inputs:
    print(FA(i))

```

Output:

- Accepted
- Accepted
- Accepted
- Rejected
- Rejected
- Rejected
- Rejected

## D) Implementation of Deductive Chart Parsing using context free grammar and a given sentence.

```
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)

# all_tokens = ['I', 'saw', 'a', 'bird', 'in', 'my', 'balcony']

parser = nltk.ChartParser(grammar1)

for tree in parser.parse(all_tokens):

    print(tree)

    tree.draw()
```

NLP Practical 7-D.py - C:/Users/Sneha/OneDrive/Desktop/NLP Practical/NLP Practical 7/NLP Practical 7-D.py (3.7.8)

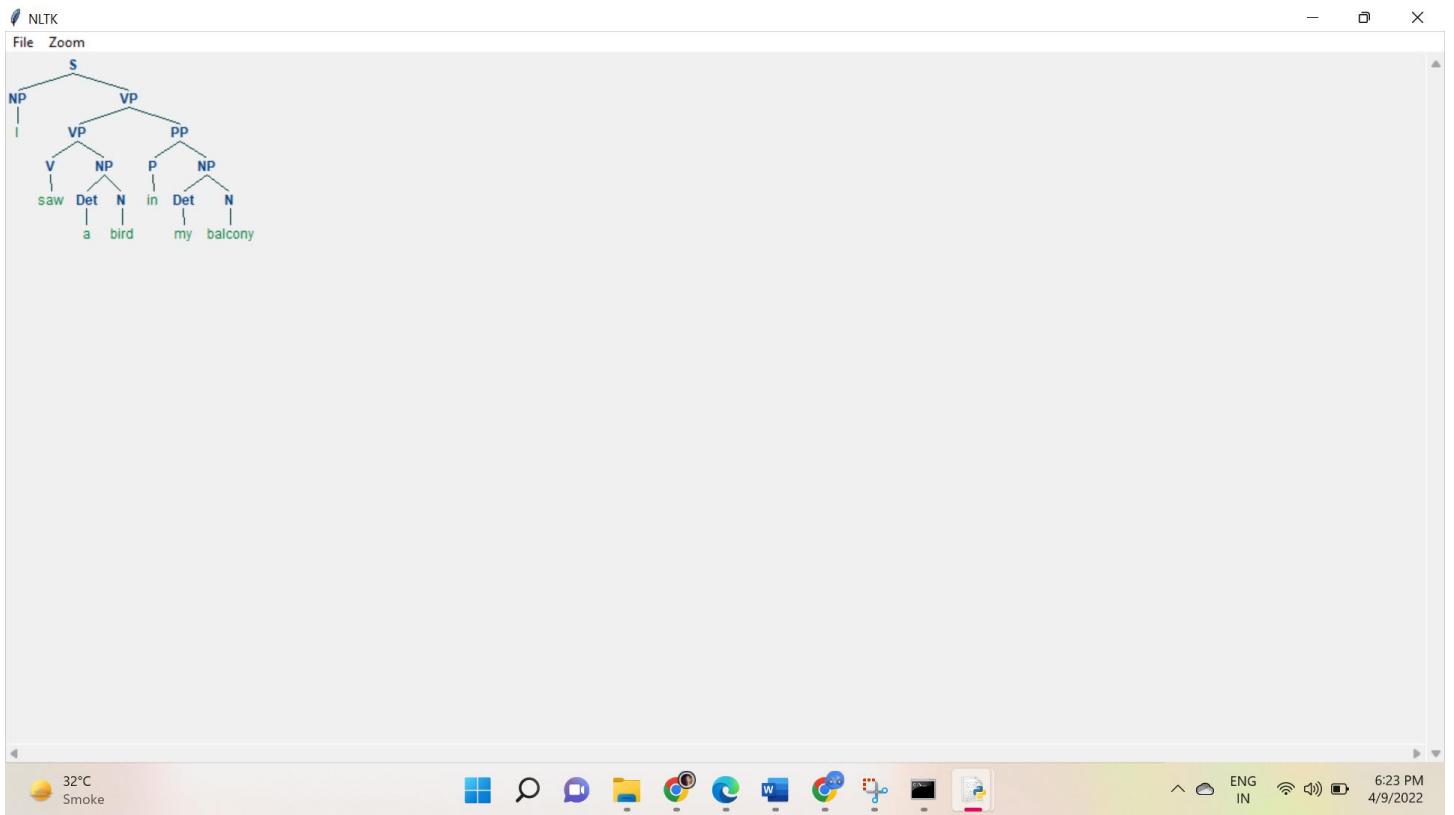
```
File Edit Format Run Options Window Help
import nltk
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
S -> NP VP
NP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'a' | 'my'
N -> 'bird' | 'balcony'
V -> 'saw'
P -> 'in'
""")
sentence = "I saw a bird in my balcony"
for index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

32°C  
Smoke

LN: 17 Col: 0

6:23 PM  
4/9/2022

```
>>>
=====
RESTART: C:/Users/Sneha/OneDrive/Desktop/NLP Practical/NLP Practical 7/NLP Practical 7-D.py =====
['I', 'saw', 'a', 'bird', 'in', 'my', 'balcony']
(S
  (NP I)
  (VP
    (VP (V saw) (NP (Det a) (N bird)))
    (PP (P in) (NP (Det my) (N balcony)))))
```



# Practical No. 08

## Study PorterStemmer, LancasterStemmer, RegexpStemmer, SnowballStemmer Study WordNetLemmatizer

```
#PorterStemmer

import nltk
from nltk.stem import PorterStemmer
word_stemmer = PorterStemmer()
print(word_stemmer.stem('writing'))

import nltk
from nltk.stem import LancasterStemmer
Lanc_stemmer = LancasterStemmer()
print(Lanc_stemmer.stem('writing'))

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

import nltk
from nltk.stem import SnowballStemmer
english_stemmer = SnowballStemmer('english')
print(english_stemmer.stem ('writing'))

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

The screenshot shows a Google Colab notebook titled "NLP Practical 8.ipynb". The code cell contains Python code demonstrating various stemming and lemmatization methods from the NLTK library:

```
#PorterStemmer
import nltk
from nltk.stem import PorterStemmer
word_stemmer = PorterStemmer()
print(word_stemmer.stem('writing'))

import nltk
from nltk.stem import LancasterStemmer
Lanc_stemmer = LancasterStemmer()
print(Lanc_stemmer.stem('writing'))

import nltk
from nltk.stem import RegexStemmer
Reg_stemmer = RegexStemmer('ing\$|s\$e\$able\$', min=4)
print(Reg_stemmer.stem('writing'))

import nltk
from nltk.stem import SnowballStemmer
english_stemmer = SnowballStemmer('english')
print(english_stemmer.stem ('writing'))

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

```
▷ write
write
write
write
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]  Unzipping corpora/wordnet.zip.
word : lemma
rocks : rock
corpora : corpus
```

# **Practical No. 09**

## **Implement Naïve Bayes Classifier**

**Code:**

```
import pandas as pd

import numpy as np

sms_data = pd.read_csv("/content/spam.csv", encoding='latin-1')

import re

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

nltk.download('stopwords')
```

The screenshot shows a Google Colab notebook titled "NLP Practical 9.ipynb". The code cell contains Python code to implement a Naive Bayes classifier. The code imports pandas, numpy, and the spam dataset from a CSV file. It then imports re, nltk, and PorterStemmer, and downloads the stopwords package. The execution cell shows the download progress: "Downloading package stopwords to /root/nltk\_data..." and "Unzipping corpora/stopwords.zip." The final step is to create a PorterStemmer object. The status bar at the bottom indicates the operation completed at 8:19 PM.

```
[1] import pandas as pd
import numpy as np
sms_data = pd.read_csv("/content/spam.csv", encoding='latin-1')

[2] import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
nltk.download('stopwords')

[3] stemming = PorterStemmer()
```

```
stemming = PorterStemmer()

corpus = []

for i in range (0,len(sms_data)):

    s1 = re.sub('^[a-zA-Z]',repl = ' ',string = sms_data['v2'][i])

    s1.lower()

    s1 = s1.split()

    s1 = [stemming.stem(word) for word in s1 if word not in
set(stopwords.words('english'))]

    s1 = ' '.join(s1)

    corpus.append(s1)

from sklearn.feature_extraction.text import CountVectorizer

countvectorizer =CountVectorizer()

x = countvectorizer.fit_transform(corpus).toarray()

print(x)
```

```
y = sms_data['v1'].values

print(y)

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,
stratify=y,random_state=2)

#Multinomial Naïve Bayes.

from sklearn.naive_bayes import MultinomialNB

multinomialnb = MultinomialNB()

multinomialnb.fit(x_train,y_train)

# Predicting on test data:

y_pred = multinomialnb.predict(x_test)

print(y_pred)

#Results of our Models

from sklearn.metrics import classification_report, confusion_matrix

from sklearn.metrics import accuracy_score

print(classification_report(y_test,y_pred))

print("accuracy_score: ",accuracy_score(y_test,y_pred))
```

NLP Practical 9.ipynb - Colaboratory

colab.research.google.com/drive/1IR93b3aAeud5B0OJ03d3taW8SNM7m-D#scrollTo=XUfOhB5vyUV7

```
stemming = PorterStemmer()
corpus = []
for i in range (0,len(sms_data)):
    si = sms_data["text"][i].replace(' ','',string = sms_data['text'][i])
    si_low = si.lower()
    si = si_low.split()
    si = [stemming.stem(word) for word in si if word not in set(stopwords.words('english'))]
    si = ' '.join(si)
    corpus.append(si)
corpus.append(si)

from sklearn.feature_extraction.text import CountVectorizer
countvectorizer = CountVectorizer()
x = countvectorizer.fit_transform(corpus).toarray()
print(x)

y = sms_data['label'].values
print(y)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3,
stratify=y,random_state=42)

# Multinomial Naive Bayes.
from sklearn.naive_bayes import MultinomialNB
multinomial = MultinomialNB()
multinomial.fit(x_train,y_train)
# Predicting on test data:
y_pred = multinomial.predict(x_test)
print(y_pred)

# Evaluating our models
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import accuracy_score
print(classification_report(y_test,y_pred))
print("accuracy score:",accuracy_score(y_test,y_pred))

[[0 0 0 ... 0 0]
 [0 0 0 ... 0 0]
 [0 0 0 ... 0 0]
 ...
 [0 0 0 ... 0 0]
 [0 0 0 ... 0 0]
 [0 0 0 ... 0 0]
 [ham ham ham ... ham ham ham]
 [ham ham ham ham ham ham]
 precision recall f1-score support
ham 0.99 0.99 0.99 1446
spam 0.92 0.93 0.92 224

accuracy: 0.979666885649393
accuracy_score: 0.979666885649393
```



## Practical no 10

### A. Speech Tagging

#### i. Speech Tagging using spaCy

Code

```
import spacy

sp = spacy.load('en_core_web_sm')

sen = sp(u"I like to play football. I hated it in my childhood though")

print(sen.text)

print(sen[7].pos_)

print(sen[7].tag_)

print(spacy.explain(sen[7].tag_))

for word in sen:

    print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}} {spacy.explain(word.tag_)}')

sen = sp(u'Can you google it?')

word = sen[2]

print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}} {spacy.explain(word.tag_)}')

sen = sp(u'Can you search it on google?')

word = sen[5]

print(f'{word.text:{12}} {word.pos_:{10}} {word.tag_:{8}} {spacy.explain(word.tag_)}')

#Finding the Number of POS Tags

sen = sp(u"I like to play football. I hated it in my childhood though")

num_pos = sen.count_by(spacy.attrs.POS)
```

```

num_pos

for k,v in sorted(num_pos.items()):
    print(f'{k}: {sen.vocab[k].text}:{v}')

```

## #Visualizing Parts of Speech Tags

```
from spacy import displacy
```

```
sen = sp(u"I like to play football. I hated it in my childhood though")
```

```
displacy.serve(sen, style='dep', options={'distance': 120})
```

The screenshot shows a dependency parse visualization for the sentence "I like to play football. I hated it in my childhood though". The tokens are listed on the left, followed by their part-of-speech tags (POS), and then their dependencies. The dependencies are represented by arrows pointing from the head word to the dependent word, indicating the grammatical relationship. The POS tags include VERB, NOUN, PRON, ADP, and PUNCT. The dependencies show how words like 'like', 'hated', and 'play' act as verbs, while 'it', 'in', and 'my' act as dependents.

Token	POS	Depository	Relationship
I	PRON		pronoun, personal
like	VERB		verb, non-3rd person singular present
to	PART	like	infinitival "to"
play	VERB		verb, base form
football	NOUN		noun, singular or mass
.	PUNCT		punctuation mark, sentence closer
I	PRON		pronoun, personal
hated	VERB		verb, past tense
it	PRON	hated	pronoun, personal
in	ADP		conjunction, subordinating or preposition
my	DET		pronoun, possessive
childhood	NOUN		noun, singular or mass
though	SCONJ		conjunction, subordinating or preposition
google	VERB		verb, base form
google	PROPN		noun, proper singular
85. ADP	:	1	
90. DET	:	1	
92. NOUN	:	2	
94. PART	:	1	
95. PRON	:	3	
97. PUNCT	:	1	
98. SCONJ	:	1	
100. VERB	:	3	

## i. Speech Tagging using nltk

### Code

```

nltk.download('punkt')

nltk.download('averaged_perceptron_tagger')

import nltk

from nltk.corpus import state_union

from nltk.tokenize import PunktSentenceTokenizer

import nltk

```

```
nltk.download('state_union')
```

```
#create our training and testing data:
```

```
train_text = state_union.raw("2005-GWBush.txt")
```

```
sample_text = state_union.raw("2006-GWBush.txt")
```

```
#train the Punkt tokenizer like:
```

```
custom_sent_tokenizer = PunktSentenceTokenizer(train_text)
```

```
# tokenize:
```

```
tokenized = custom_sent_tokenizer.tokenize(sample_text)
```

```
def process_content():
```

```
    try:
```

```
        for i in tokenized[:2]:
```

```
            words = nltk.word_tokenize(i)
```

```
            tagged = nltk.pos_tag(words)
```

```
            print(tagged)
```

```
    except Exception as e:
```

```
        print(str(e))
```

```
process_content()
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]   /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package state_union to /root/nltk_data...
[nltk_data]   Package state_union is already up-to-date!
[('PRESIDENT', 'NNP'), ('GEORGE', 'NNP'), ('W.', 'NNP'), ('BUSH', 'NNP'), ("S", 'POS'), ('ADDRESS', 'NNP'), ('BEFORE', 'IN'), ('A', 'NNP'), ('J
[('Mr.', 'NNP'), ('Speaker', 'NNP'), ('', '',''), ('Vice', 'NNP'), ('President', 'NNP'), ('Cheney', 'NNP'), ('', '',''), ('members', 'NNS'), ('of'

```

## A. Statistical Parsing

### i. Usage of Give and Gave in the Penn treebank sample

## Code

```
import nltk

import nltk.parse.viterbi

import nltk.parse.pchart

def give(t):

    return t.label() == 'VP' and len(t) > 2 and t[1].label() == 'NP' \
           and (t[2].label() == 'PP-DTV' or t[2].label() == 'NP') \
           and ('give' in t[0].leaves() or 'gave' in t[0].leaves())

def sent(t):

    return ' '.join(token for token in t.leaves() if token[0] not in '*-0')

def print_node(t, width):

    output = "%s %s: %s / %s: %s" %(sent(t[0]), t[1].label(), sent(t[1]), t[2].label(), sent(t[2]))

    if len(output) > width:

        output = output[:width] + "..."

    print (output)

for tree in nltk.corpus.treebank.parsed_sents():

    for t in tree.subtrees(give):

        print_node(t, 72)
```

```
↳ [nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]  Unzipping corpora/treebank.zip.
gave NP: the chefs / NP: a standing ovation
give NP: advertisers / NP: discounts for maintaining or increasing ad sp...
give NP: it / PP-DTV: to the politicians
gave NP: them / NP: similar help
give NP: them / NP:
give NP: only French history questions / PP-DTV: to students in a Europe...
give NP: federal judges / NP: a raise
give NP: consumers / NP: the straight scoop on the U.S. waste crisis
gave NP: Mitsui / NP: access to a high-tech medical product
give NP: Mitsubishi / NP: a window on the U.S. glass industry
give NP: much thought / PP-DTV: to the rates she was receiving , nor to ...
give NP: your Foster Savings Institution / NP: the gift of hope and free...
give NP: market operators / NP: the authority to suspend trading in futu...
gave NP: quick approval / PP-DTV: to $ 3.18 billion in supplemental appr...
give NP: the Transportation Department / NP: up to 50 days to review any...
give NP: the president / NP: such power
give NP: me / NP: the heebie-jeebies
give NP: holders / NP: the right , but not the obligation , to buy a cal...
gave NP: Mr. Thomas / NP: only a `` qualified '' rating , rather than `` ...
give NP: the president / NP: line-item veto power
```

---

### i. Probabilistic parser

Code

```
import nltk

from nltk import PCFG

grammar = PCFG.fromstring(""

NP -> NNS [0.5] | JJ NNS [0.3] | NP CC NP [0.2]

NNS -> "men" [0.1] | "women" [0.2] | "children" [0.3] | NNS CC NNS [0.4]

JJ -> "old" [0.4] | "young" [0.6]

CC -> "and" [0.9] | "or" [0.1]

")

print(grammar)

viterbi_parser = nltk.ViterbiParser(grammar)
```

```
token = "old men and women".split()  
  
obj = viterbi_parser.parse(token)  
  
print("Output: ")  
  
for x in obj:  
  
    print(x)  
  


---



```
Grammar with 11 productions (start state = NP)  
NP -> NNS [0.5]  
NP -> JJ NNS [0.3]  
NP -> NP CC NP [0.2]  
NNS -> 'men' [0.1]  
NNS -> 'women' [0.2]  
NNS -> 'children' [0.3]  
NNS -> NNS CC NNS [0.4]  
JJ -> 'old' [0.4]  
JJ -> 'young' [0.6]  
CC -> 'and' [0.9]  
CC -> 'or' [0.1]  
Output:  
(NP (JJ old) (NNS (NNS men) (CC and) (NNS women))) (p=0.000864)
```



---


```

## C. Malt Parsing

**Parse a sentence and draw a tree using malt parsing**

Code

```
from nltk.parse import malt  
mp = malt.MaltParser('maltparser-1.7.2', 'engmalt.linear-1.7.mco')#file  
t = mp.parse_one('I saw a bird from my window.'.split()).tree()  
print(t)  
t.draw()
```

## Practical no. 11

### A. Multiword expression in NLP

Code

```
import nltk

nltk.download('punkt')

from nltk.tokenize import MWETokenizer

from nltk import sent_tokenize, word_tokenize

s = "Good cake cost Rs.1500\kg in Mumbai. Please buy me one of them.\n\nThanks."

mwe = MWETokenizer([('New', 'York'), ('Hong', 'Kong')], separator='_')

for sent in sent_tokenize(s):

    print(mwe.tokenize(word_tokenize(sent)))

→ [nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]  Unzipping tokenizers/punkt.zip.
['Good', 'cake', 'cost', 'Rs.1500\\kg', 'in', 'Mumbai', '.']
['Please', 'buy', 'me', 'one', 'of', 'them', '.']
['Thanks', '.']
```

### A. Normalized Web Distance and word similarity

Code

```
import numpy as np

import re

!pip install textdistance

import textdistance

# we will need scikit-learn>=0.21

import sklearn #pip install sklearn
```

```
from sklearn.cluster import AgglomerativeClustering

texts = ['Reliance supermarket', 'Reliance hypermarket', 'Reliance', 'Reliance', 'Reliance downtown', 'Reliance market', 'Mumbai', 'Mumbai Hyper', 'Mumbai dxb', 'mumbai airport', 'k.m trading', 'KM Trading', 'KM trade', 'K.M. Trading', 'KM.Trading']

def normalize(text):
    """ Keep only lower-cased text and numbers"""
    return re.sub('[^a-z0-9]+', ' ', text.lower())

def group_texts(texts, threshold=0.4):
    """ Replace each text with the representative of its cluster"""

    normalized_texts = np.array([normalize(text) for text in texts])

    distances = 1 - np.array([
        [textdistance.jaro_winkler(one, another) for one in normalized_texts]
        for another in normalized_texts
    ])

    clustering = AgglomerativeClustering(
        distance_threshold=threshold, # this parameter needs to be tuned carefully
        affinity="precomputed", linkage="complete", n_clusters=None
    ).fit(distances)

    centers = dict()

    for cluster_id in set(clustering.labels_):
        index = clustering.labels_ == cluster_id

        centrality = distances[:, index][index].sum(axis=1)

        centers[cluster_id] = normalized_texts[index][centrality.argmin()]

    return [centers[i] for i in clustering.labels_]

print(group_texts(texts))
```

```
Collecting textdistance
  Downloading textdistance-4.2.2-py3-none-any.whl (28 kB)
Installing collected packages: textdistance
Successfully installed textdistance-4.2.2
['reliance', 'reliance', 'reliance', 'reliance', 'reliance', 'reliance', 'mumbai', 'mumbai', 'mumbai', 'mumbai', 'km trading', 'km trading', 'km
```

## A. Word Sense Disambiguation

Code

```
#Word Sense Disambiguation
```

```
nltk.download('wordnet')

from nltk.corpus import wordnet as wn

def get_first_sense(word, pos=None):

    if pos:

        synsets = wn.synsets(word,pos)

    else:

        synsets = wn.synsets(word)

    return synsets[0]

best_synset = get_first_sense('bank')

print ('%s: %s' % (best_synset.name, best_synset.definition))

best_synset = get_first_sense('set','n')

print ('%s: %s' % (best_synset.name, best_synset.definition))

best_synset = get_first_sense('set','v')

print ('%s: %s' % (best_synset.name, best_synset.definition))
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]  Unzipping corpora/wordnet.zip.
<bound method Synset.name of Synset('bank.n.01')>: <bound method Synset.definition of Synset('bank.n.01')>
<bound method Synset.name of Synset('set.n.01')>: <bound method Synset.definition of Synset('set.n.01')>
<bound method Synset.name of Synset('put.v.01')>: <bound method Synset.definition of Synset('put.v.01')>
```

