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

Practical No.	Aim	Date
1.		
A.	Convert the given text to speech	
B.	Convert audio file speech to text	
2.		
A.	Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like fields, raw, words, sents, categories,	
B.	Create and use your own corpora (plaintext categorical)	
C.	Study Conditional Frequency Distribution	
D.	Study of Tagged Corpora with methods like tagged_sents, tagged_words	
E.	Write a program to find the most frequent noun tags	
F.	Map words to properties using Python Dictionaries	
G.	Study i) Default tagger ii). Regular expression tagger iii) Unigram tagger	
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	
3		
A.	Study of WordNet dictionary with methods as syssets, definitions examples antonyms	
B.	Study of lemmas hyponyms hypernyms	
C.	Write a program using python to find synonyms and antonyms of word “active ” using WordNet	
D.	Compare two nouns	
	Handling stopwords	

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

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

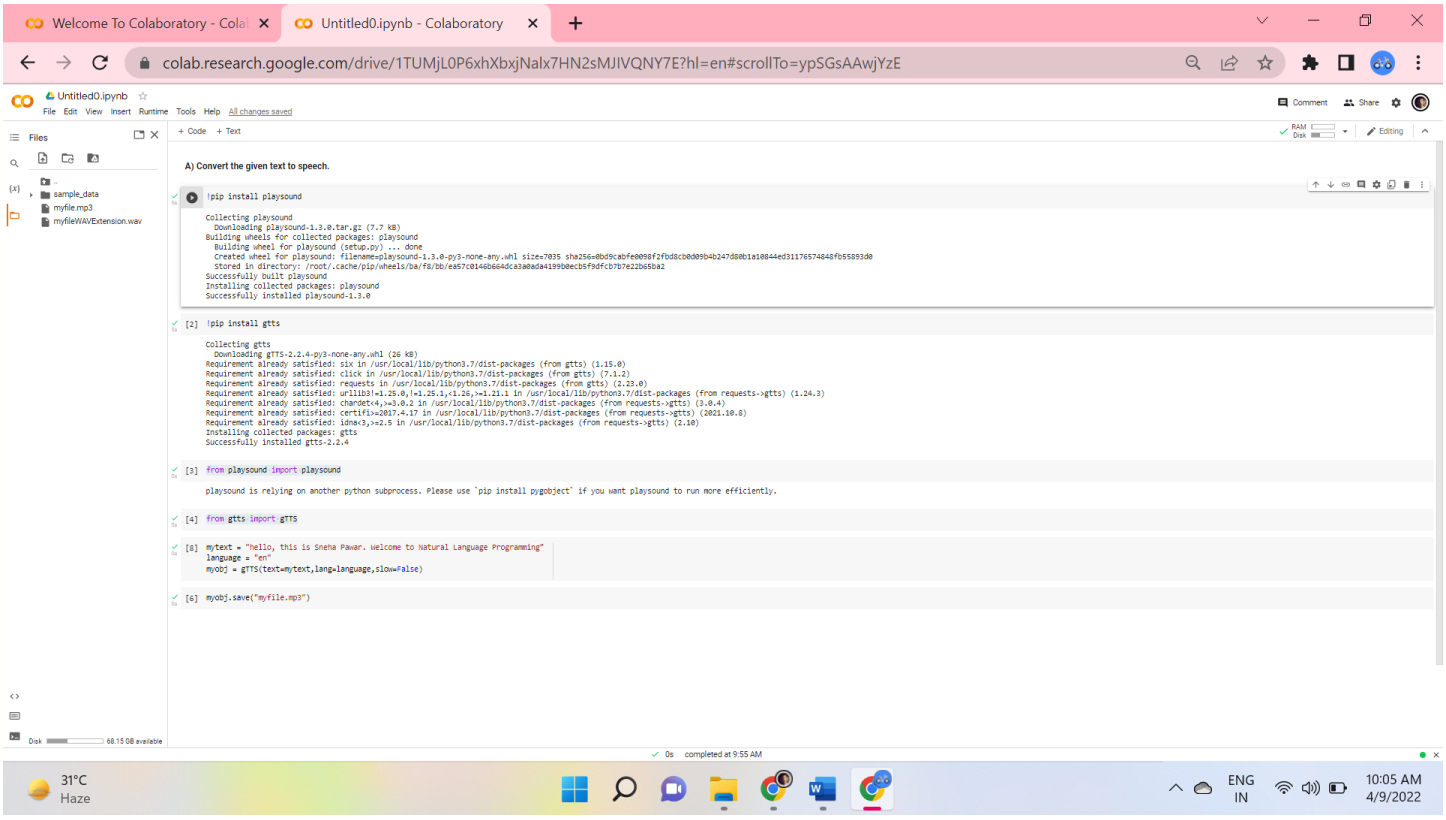
Practical No. 01

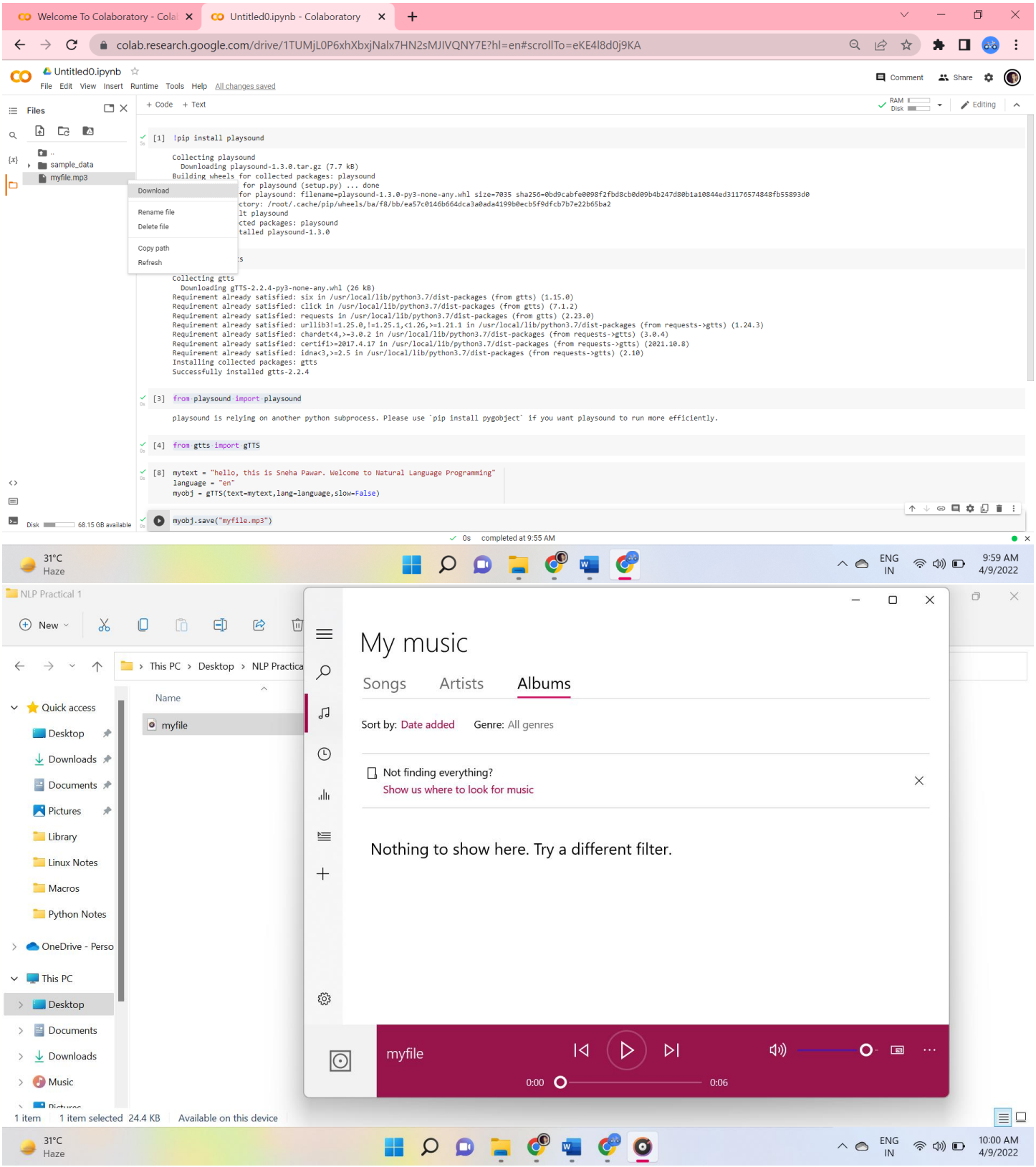
A) Convert the given text to speech.

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:



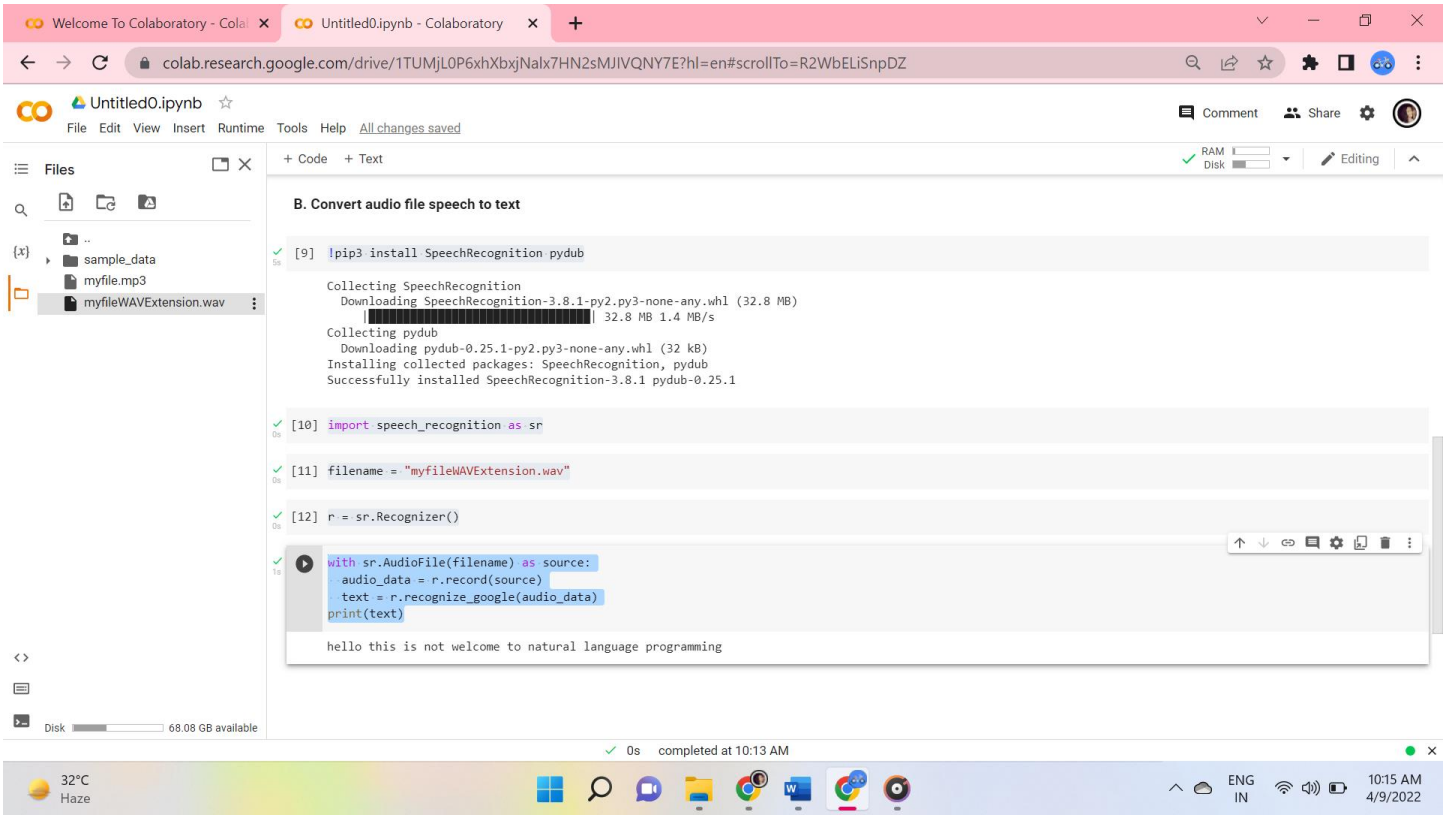

Welcome To Colaboratory - Cola
Untitled0.ipynb - Colaboratory
colab.research.google.com/drive/1TUMjL0P6xhXbxjNalx7HN2sMJIVQNY7E?hl=en#scrollTo=eKE4l8d0j9KA
Untitled0.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
sample_data
myfile.mp3
Download
Rename file
Delete file
Copy path
Refresh
+ Code + Text
Editing
[1] !pip install playsound
Collecting playsound
Downloading playsound-1.3.0.tar.gz (7.7 kB)
Building wheels for collected packages: playsound
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: 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!=1.25.0,!=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: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->gtts) (3.0.4)
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 pygobject` 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)
myobj.save("myfile.mp3")
Disk 68.15 GB available
0s completed at 9:55 AM
31°C Haze
ENG IN
9:59 AM 4/9/2022
NLP Practical 1
New
This PC › Desktop › NLP Practica
Quick access
Desktop
Downloads
Documents
Pictures
Library
Linux Notes
Macros
Python Notes
OneDrive - Perso
This PC
Desktop
Documents
Downloads
Music
Name
myfile
My music
Songs Artists Albums
Sort by: Date added Genre: All genres
Not finding everything?
Show us where to look for music
Nothing to show here. Try a different filter.
myfile
0:00 0:06
1 item 1 item selected 24.4 KB Available on this device
31°C Haze
ENG IN
10:00 AM 4/9/2022

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:

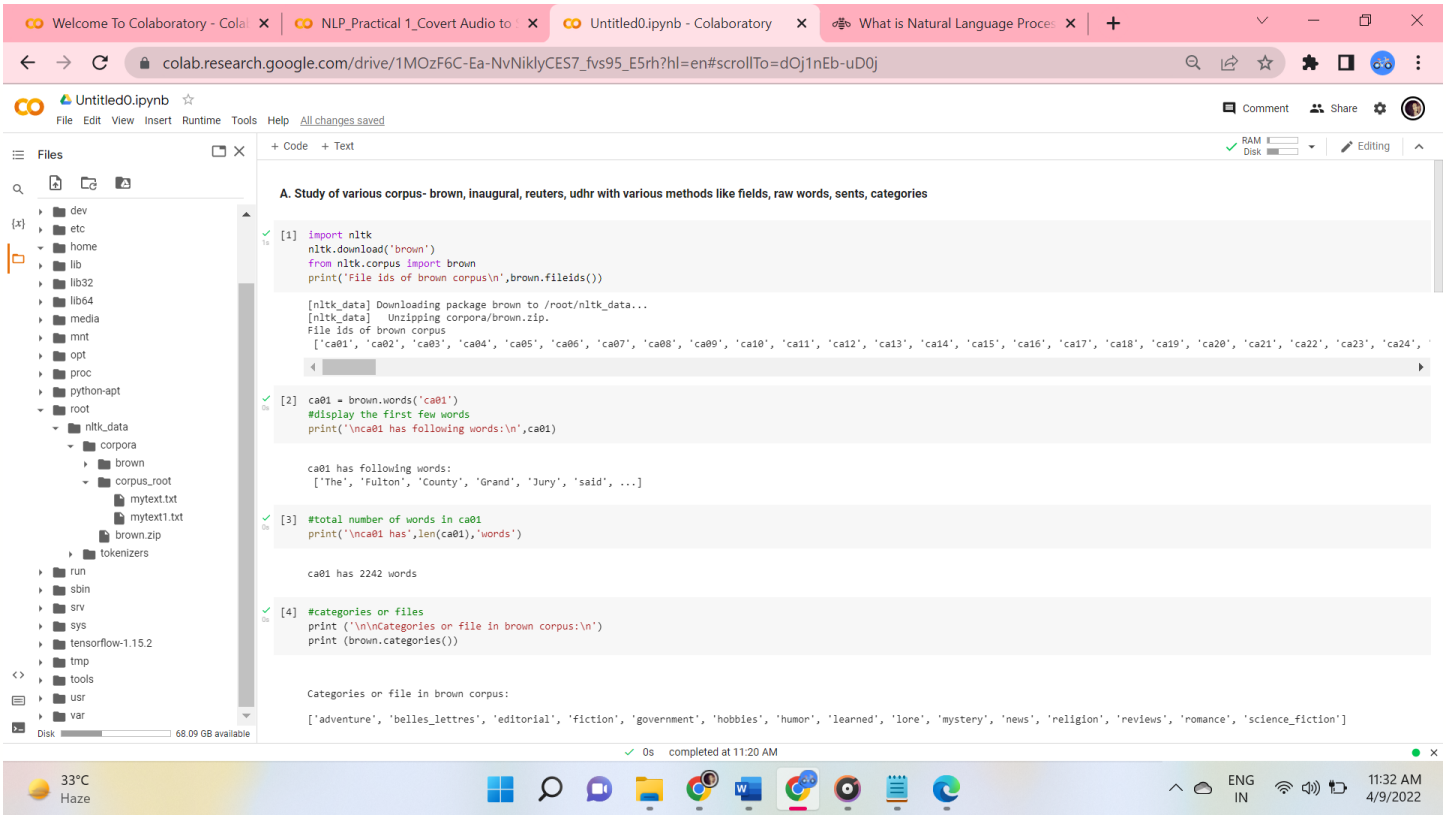


Practical No. 02

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

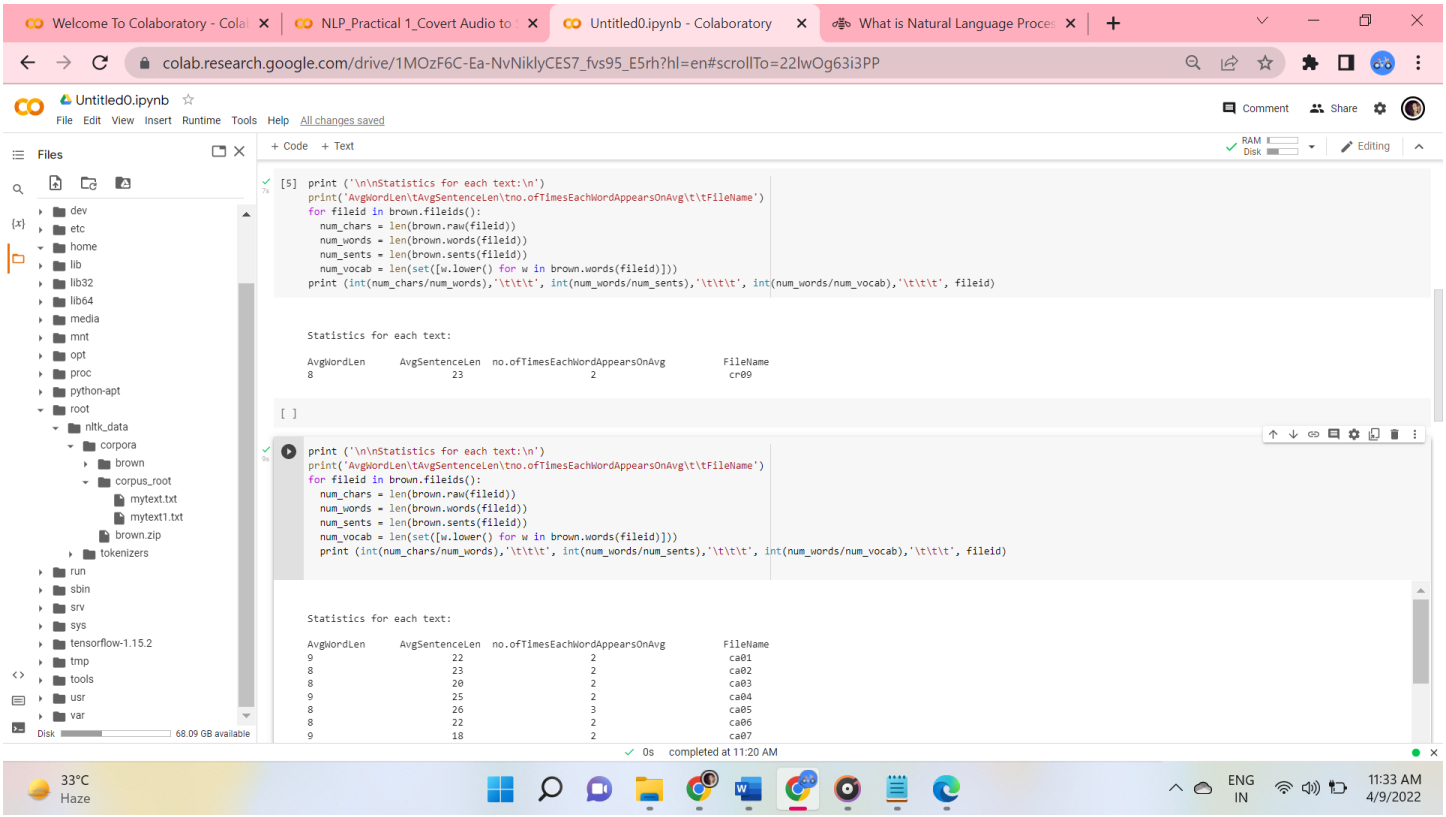
Code & Output:

```
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())
```



```
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)
```

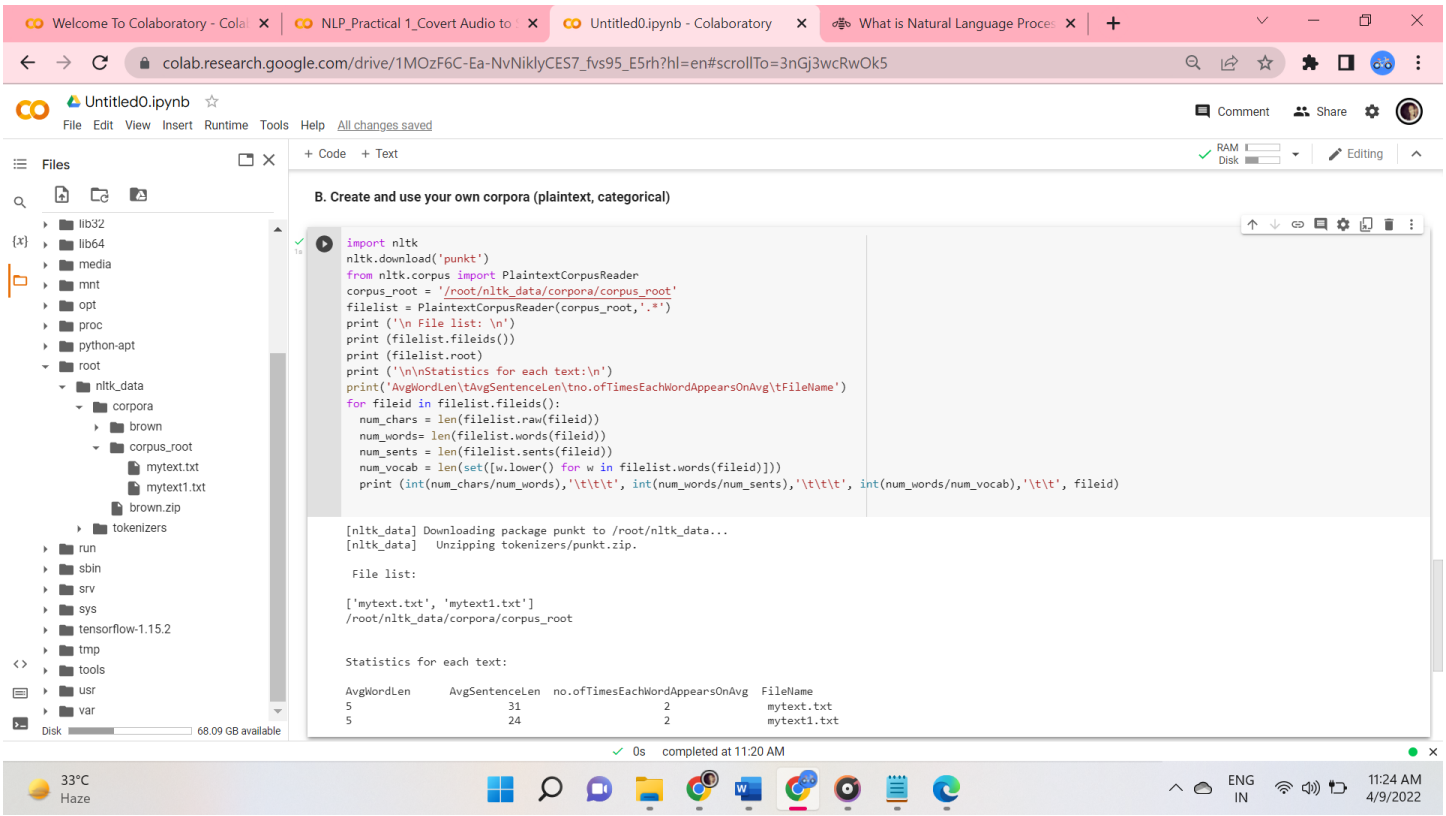
```
print ('\n\nStatistics for each text:\n')
print('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\t\tFileName')
for fileid in brown.fileids():
  num_chars = len(brown.raw(fileid))
  num_words = len(brown.words(fileid))
  num_sents = len(brown.sents(fileid))
  num_vocab = len(set([w.lower() for w in brown.words(fileid)]))
  print (int(num_chars/num_words),'\t\t\t', int(num_words/num_sents),'\t\t\t', int(nu
m_words/num_vocab),'\t\t\t', fileid)
```



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

C. Code & Output:

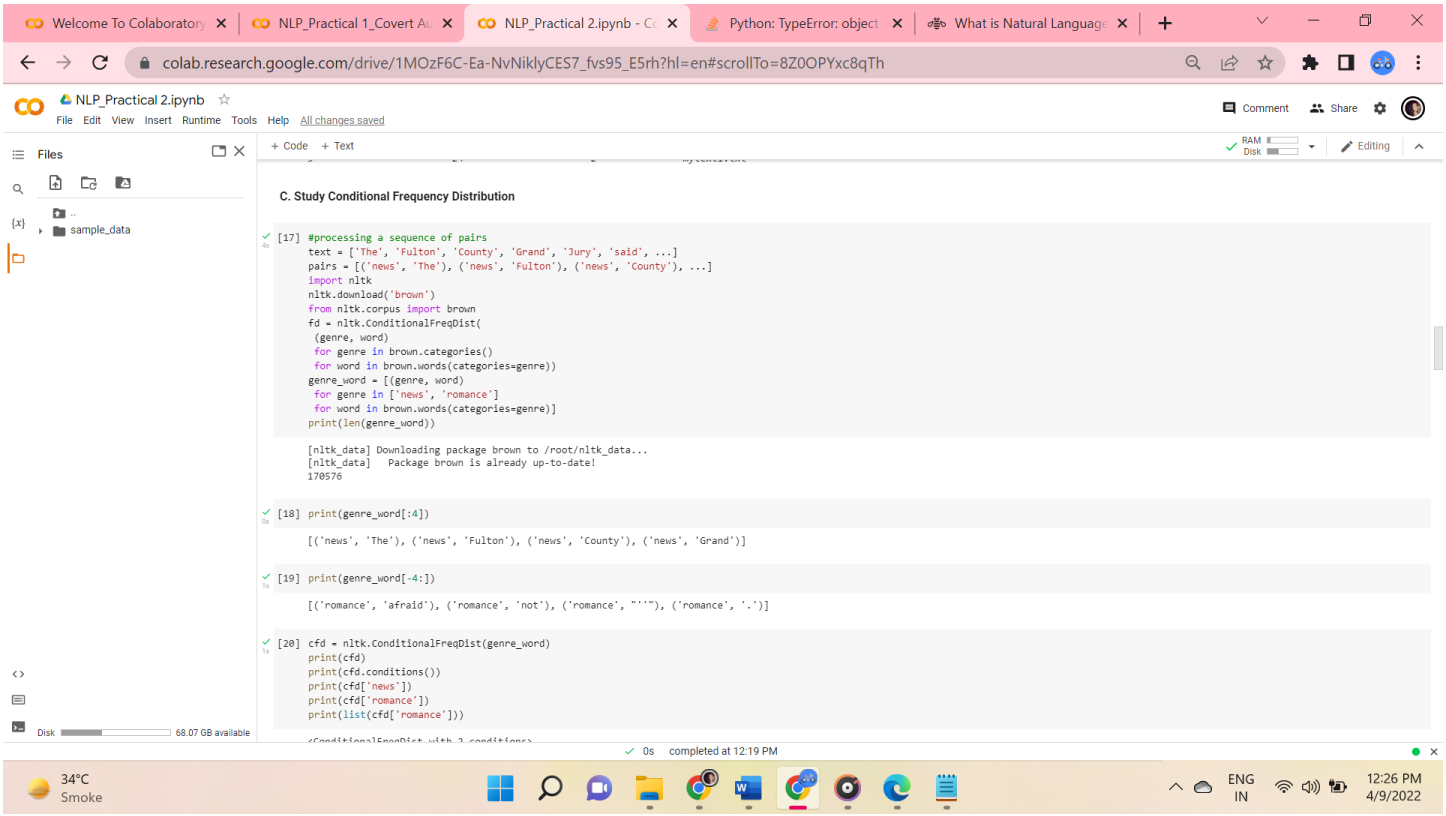
```
D. import nltk
E. nltk.download('punkt')
F. from nltk.corpus import PlaintextCorpusReader
G. corpus_root = '/root/nltk_data/corpora/corpus_root'
H. filelist = PlaintextCorpusReader(corpus_root,'.*')
I. print ('\n File list: \n')
J. print (filelist.fileids())
K. print (filelist.root)
L. print ('\n\nStatistics 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', fileid)
```

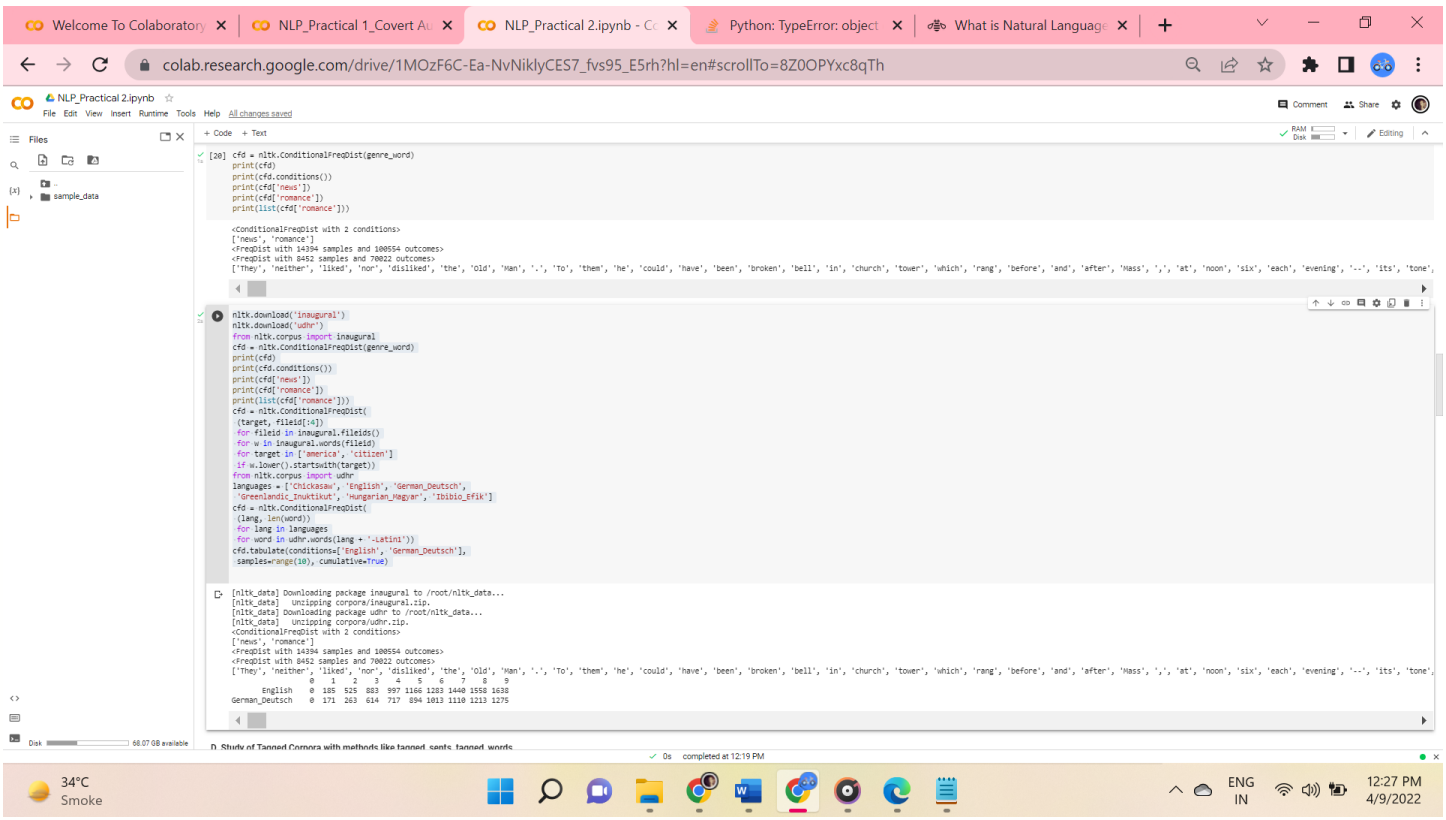


C. Study Conditional Frequency Distribution

```
#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:])


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

nltk.download('inaugural')
nltk.download('udhr')
from nltk.corpus import inaugural
cfd = nltk.ConditionalFreqDist(genre_word)
print(cfd)
print(cfd.conditions())
print(cfd['news'])
print(cfd['romance'])
print(list(cfd['romance']))
cfd = 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']
cfd = nltk.ConditionalFreqDist(
 (lang, len(word))
 for lang in languages
 for word in udhr.words(lang + '-Latin1'))
cfd.tabulate(conditions=['English', 'German_Deutsch'],
 samples=range(10), cumulative=True)
```

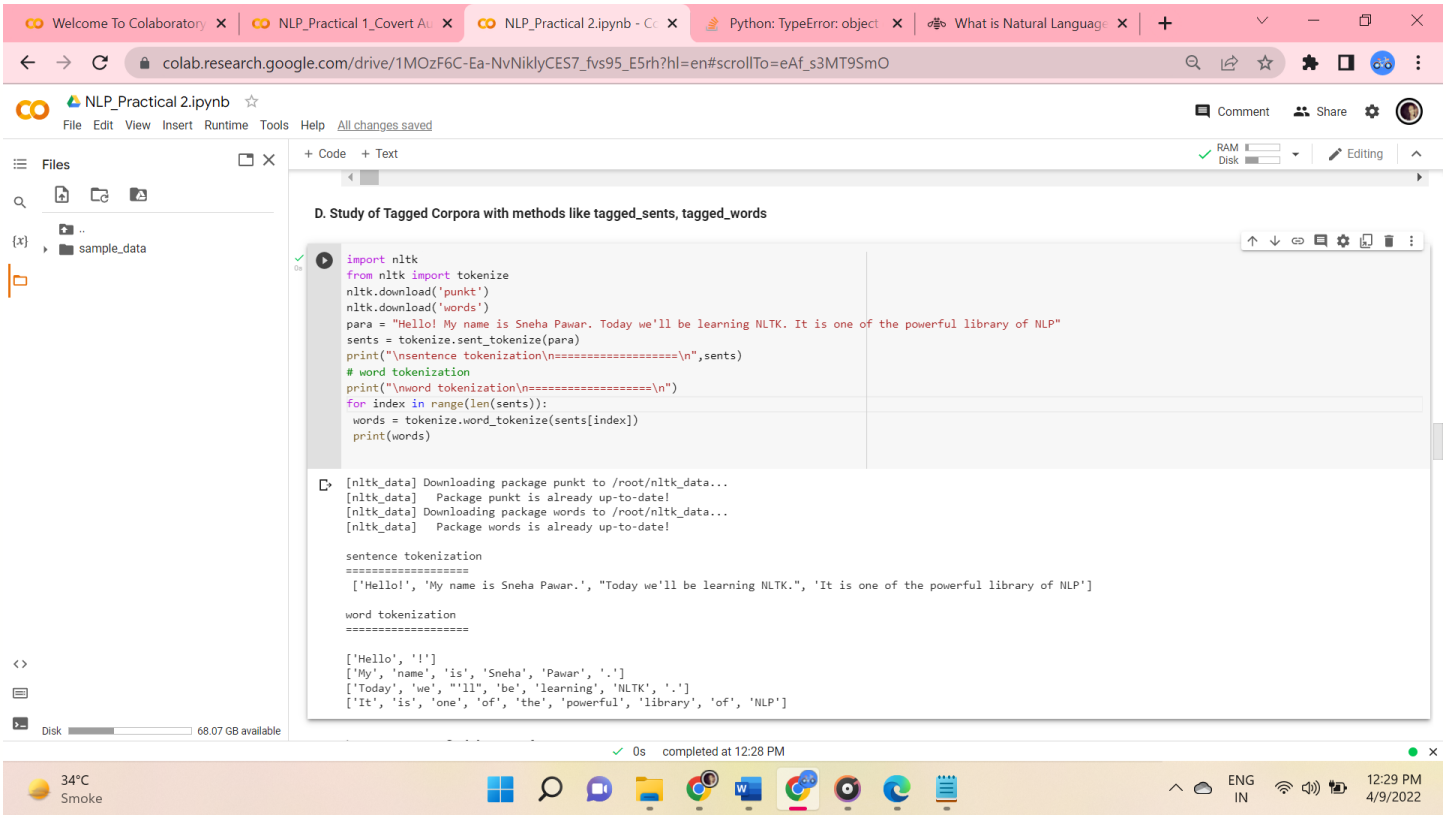




D. Study of Tagged Corpora with methods like tagged_sents, tagged_words

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

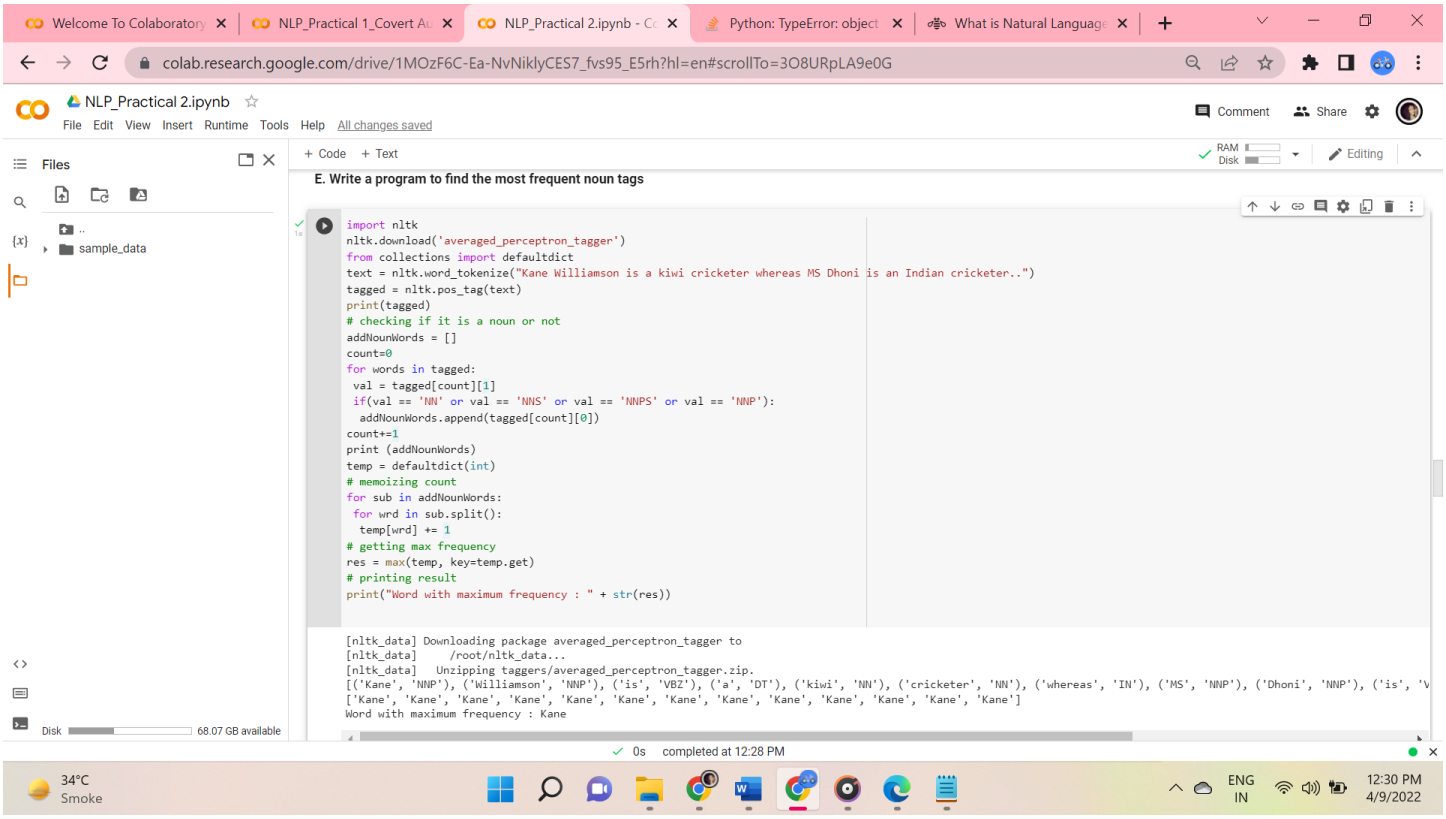
```
nltk.download('words')
para = "Hello! My name is Sneha Pawar. Today we'll be learning NLTK. It is one of the
 powerful library of NLP"
sents = tokenize.sent_tokenize(para)
print("\nsentence tokenization\n==================\n",sents)
# word tokenization
print("\nword tokenization\n==================\n")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



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

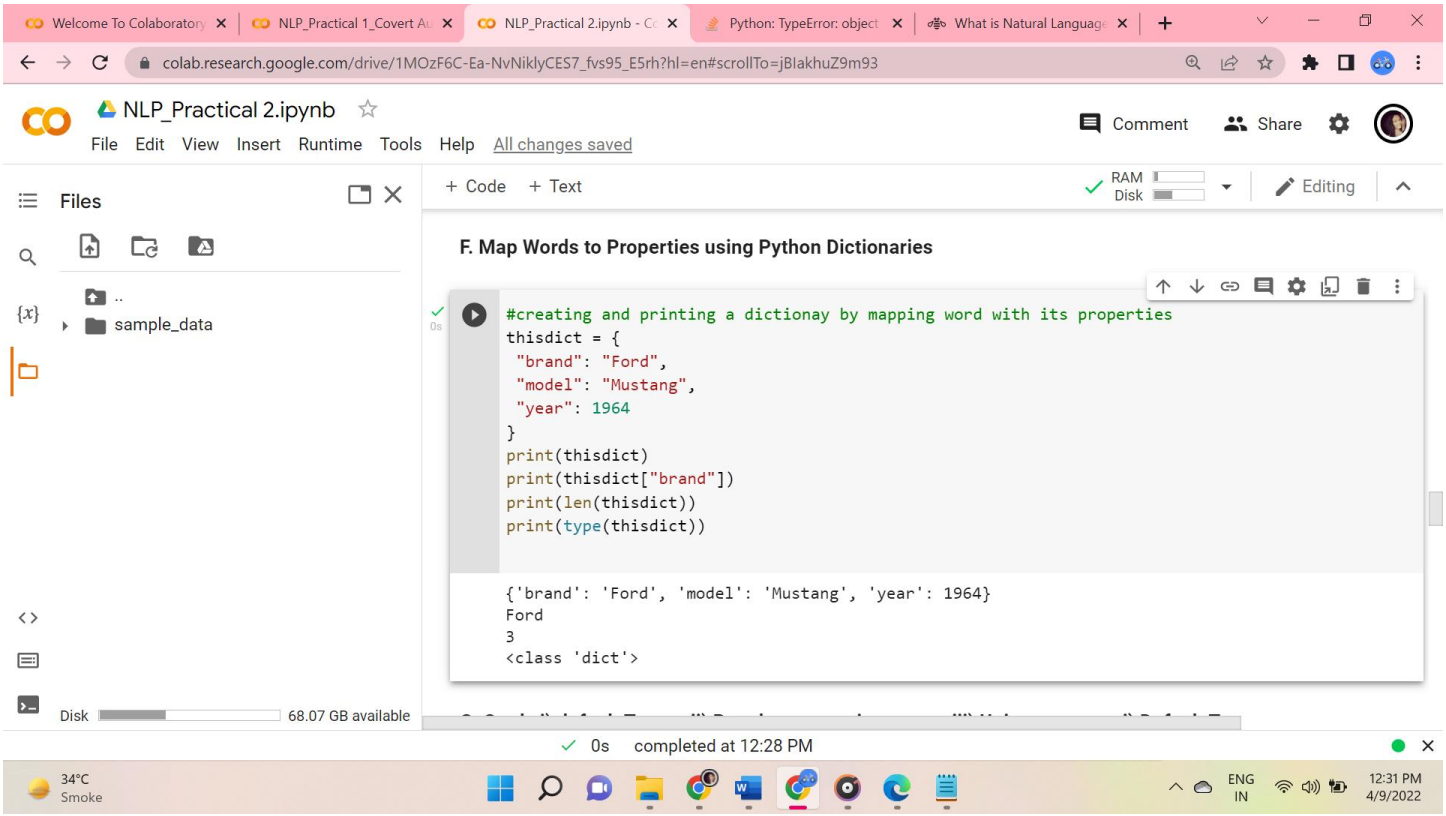
```
import nltk
nltk.download('averaged_perceptron_tagger')
from collections import defaultdict
text = nltk.word_tokenize("Kane Williamson is a kiwi cricketer whereas MS Dhoni is an
 Indian cricketer..")
tagged = nltk.pos_tag(text)
print(tagged)
# checking if it is a noun or not
addNounWords = []
count=0
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))
```



F. Map Words to Properties using Python Dictionaries

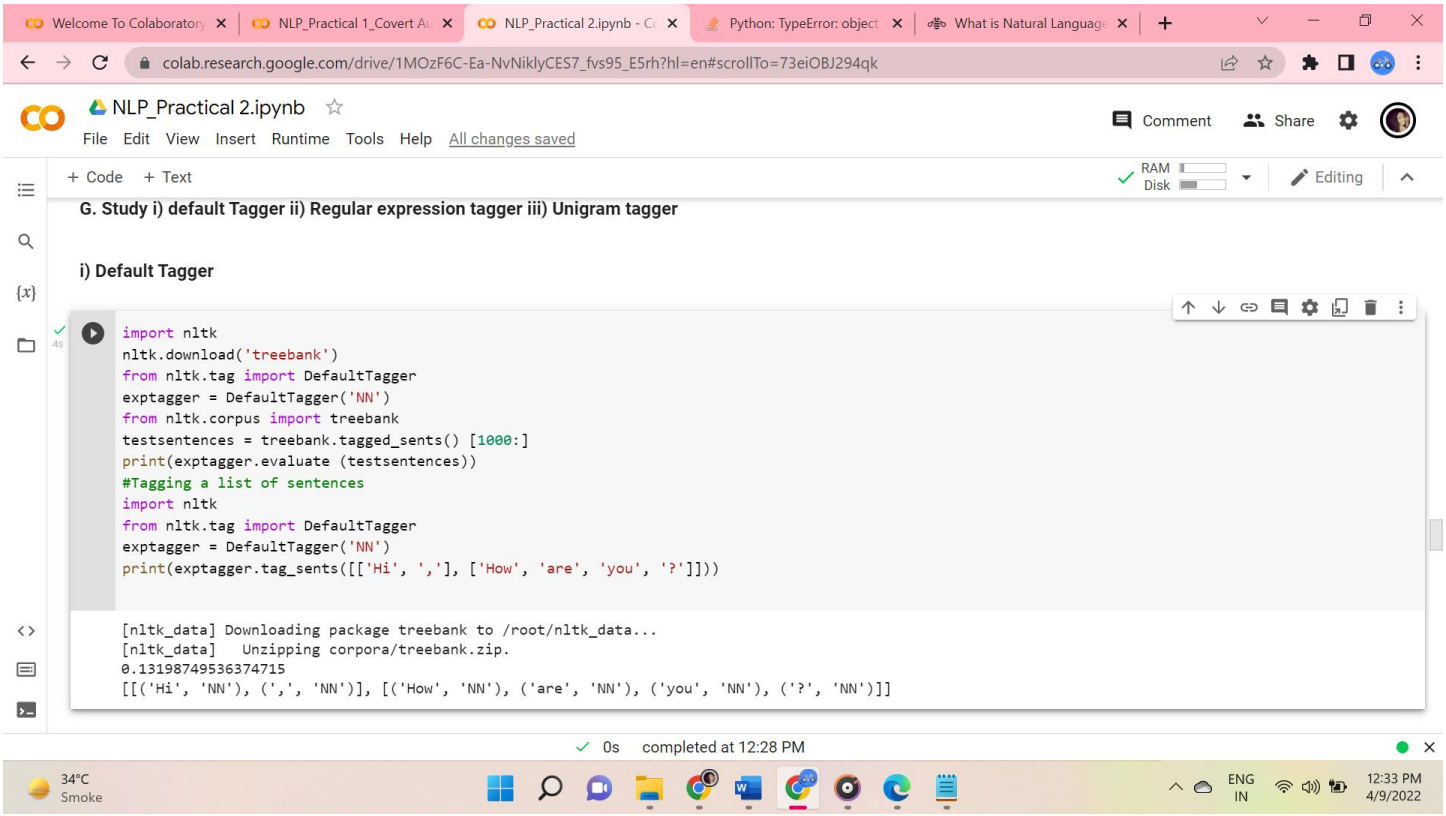
```
#creating and printing a dictionay by mapping word with its properties
thisdict = {
 "brand": "Ford",
 "model": "Mustang",
 "year": 1964
}
print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))
```



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

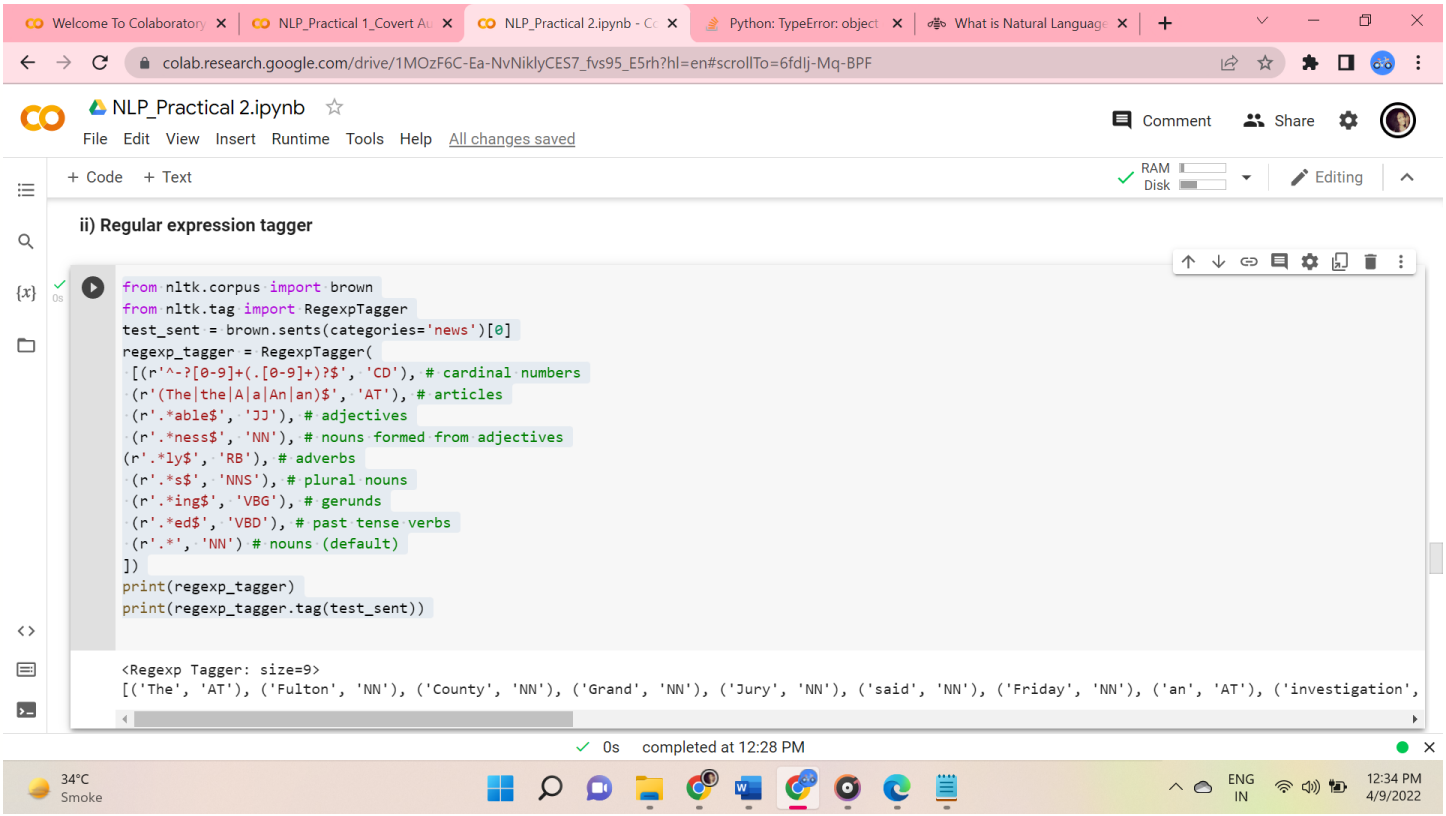
i) Default Tagger

```
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', '?']]))
```



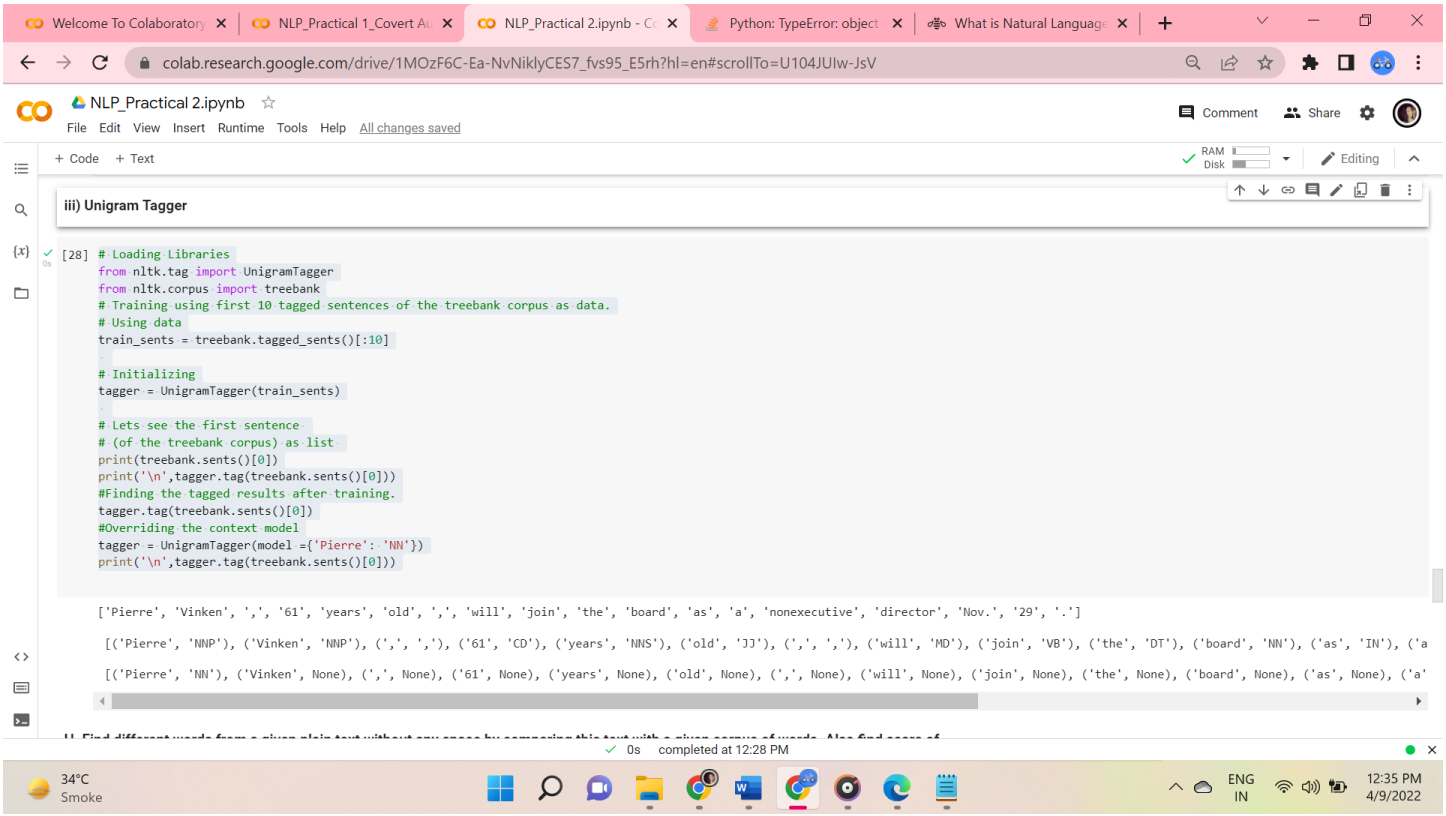
ii) Regular expression tagger

```
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))
```



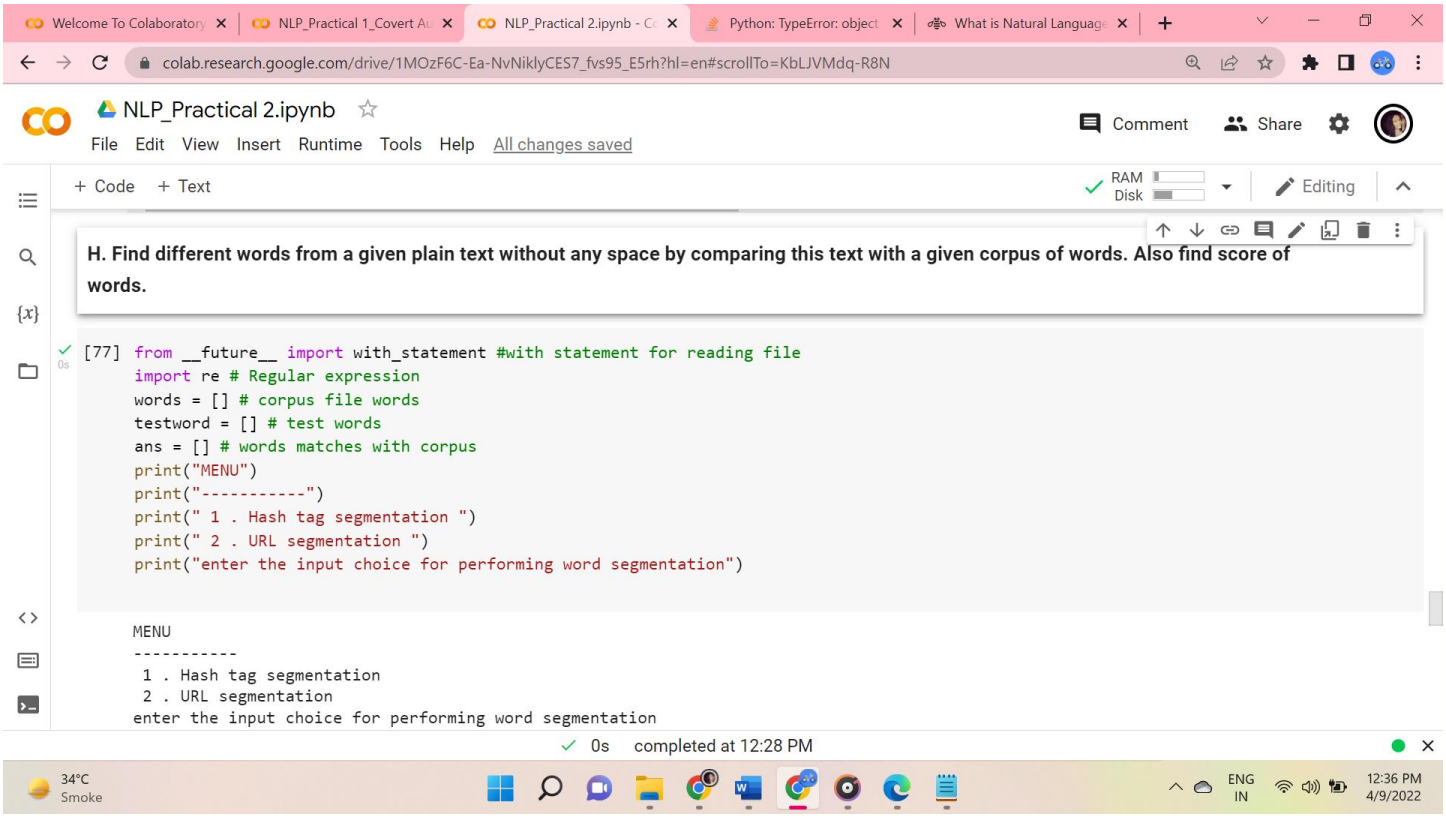
iii) Unigram Tagger

```
# 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]))
```

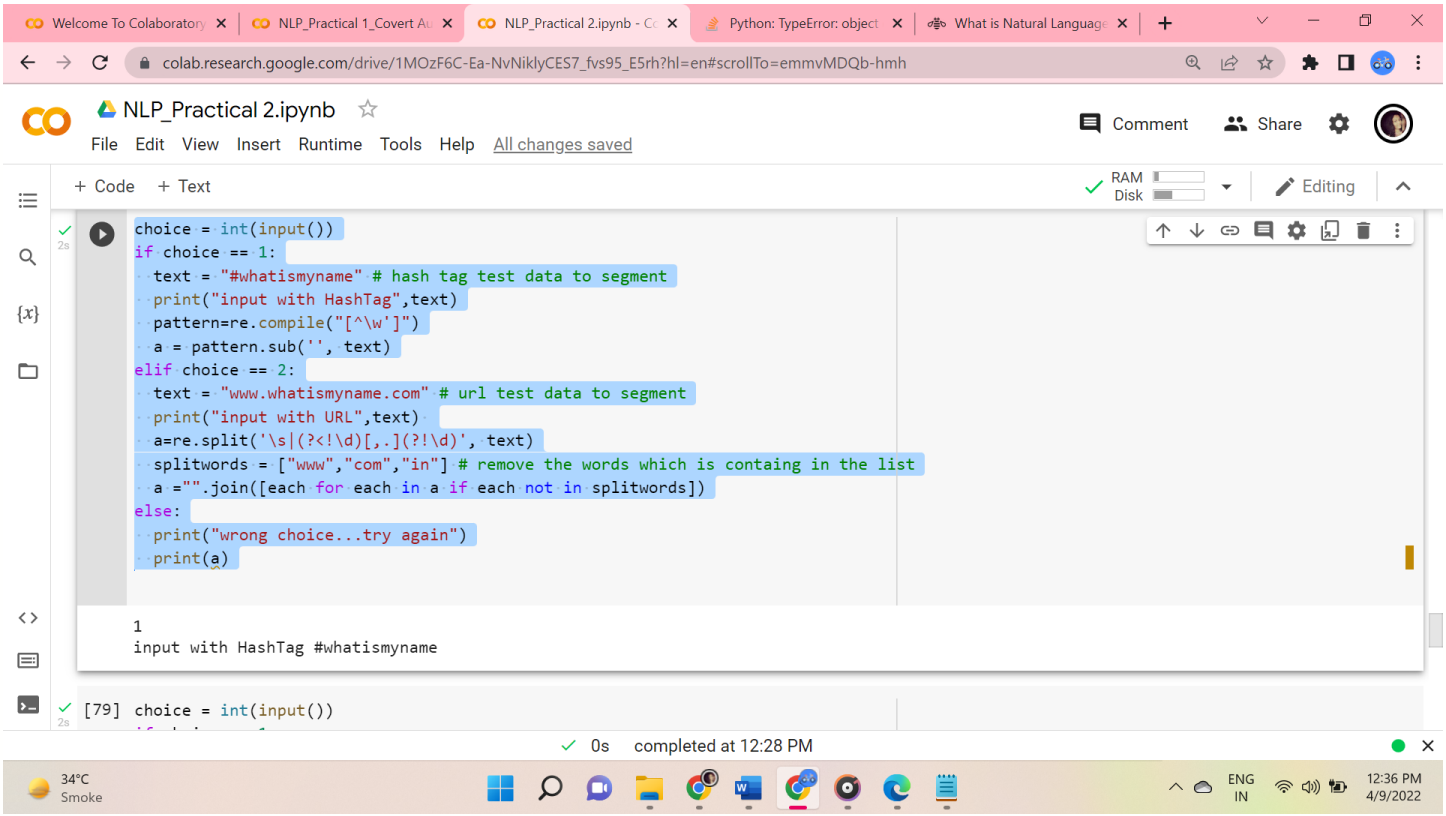


H. Find different words from a given plain text without any space by comparing this text with a given corpus of words. Also find score of words.

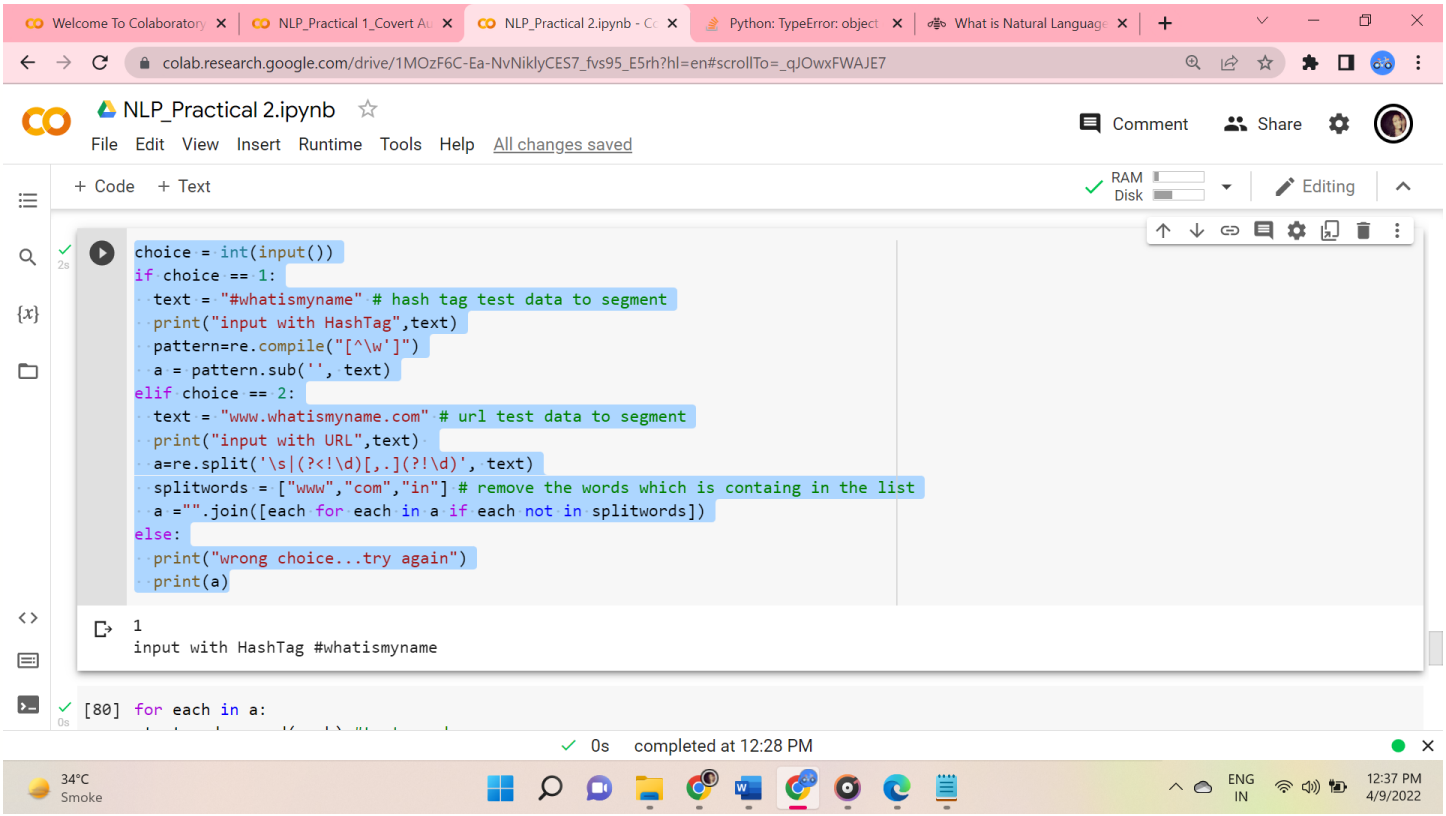
```
from __future__ import with_statement #with statement for reading file
import re # Regular expression
words = [] # corpus file words
testword = [] # test words
ans = [] # words matches with corpus
print("MENU")
print("-----------")
print(" 1 . Hash tag segmentation ")
print(" 2 . URL segmentation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




```
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)
```

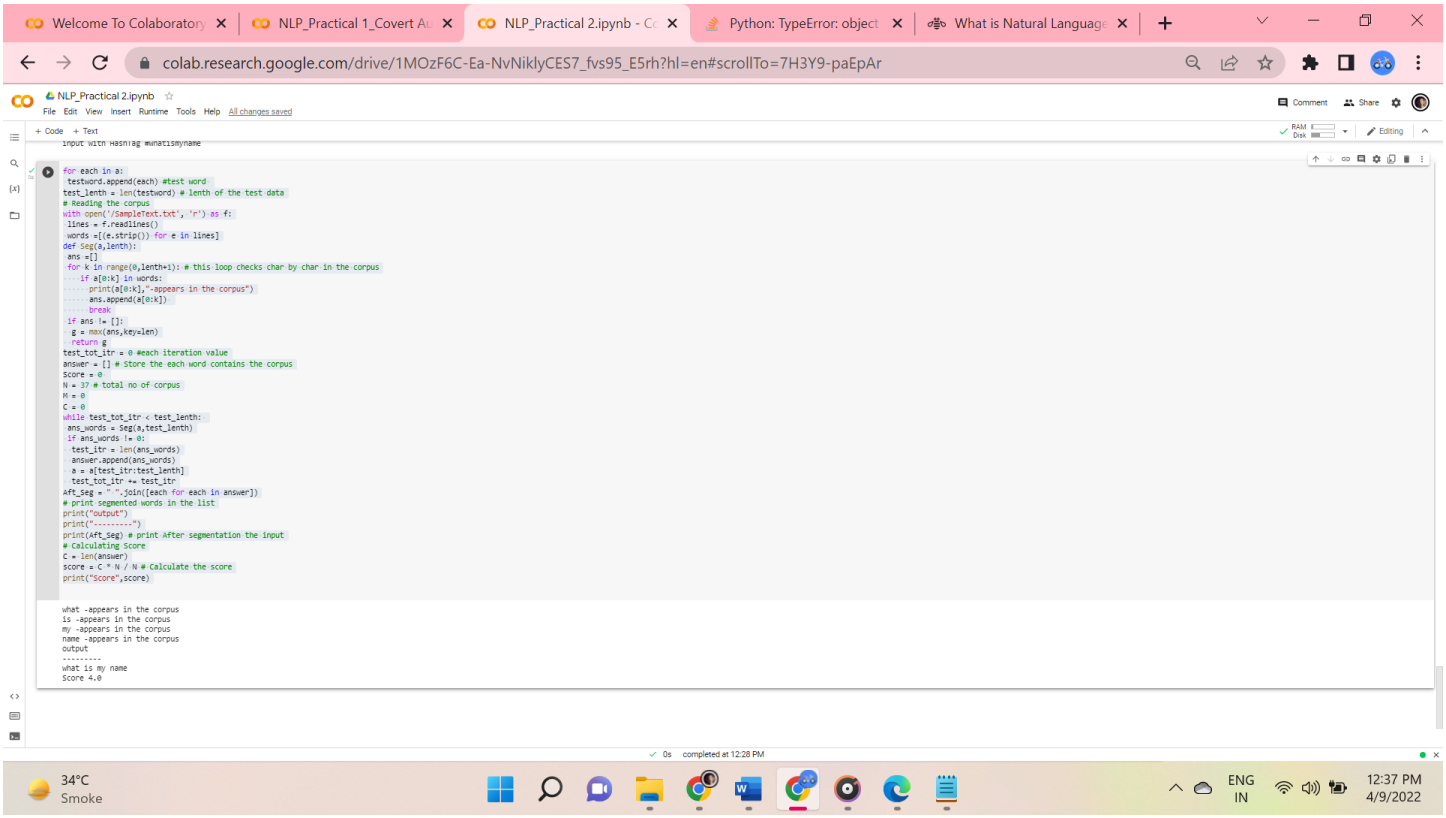



```
choice = int(input())
if choice == 1:
  text = "#whatismyname" # hash tag test data to segment
  print("input with HashTag",text)
  pattern=re.compile("[^\w']")
  a = pattern.sub('', text)
elif choice == 2:
  text = "www.whatismyname.com" # url test data to segment
  print("input with URL",text)
  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)
```




```
for each in a:
 testword.append(each) #test word
test_lenth = len(testword) # lenth 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)
```

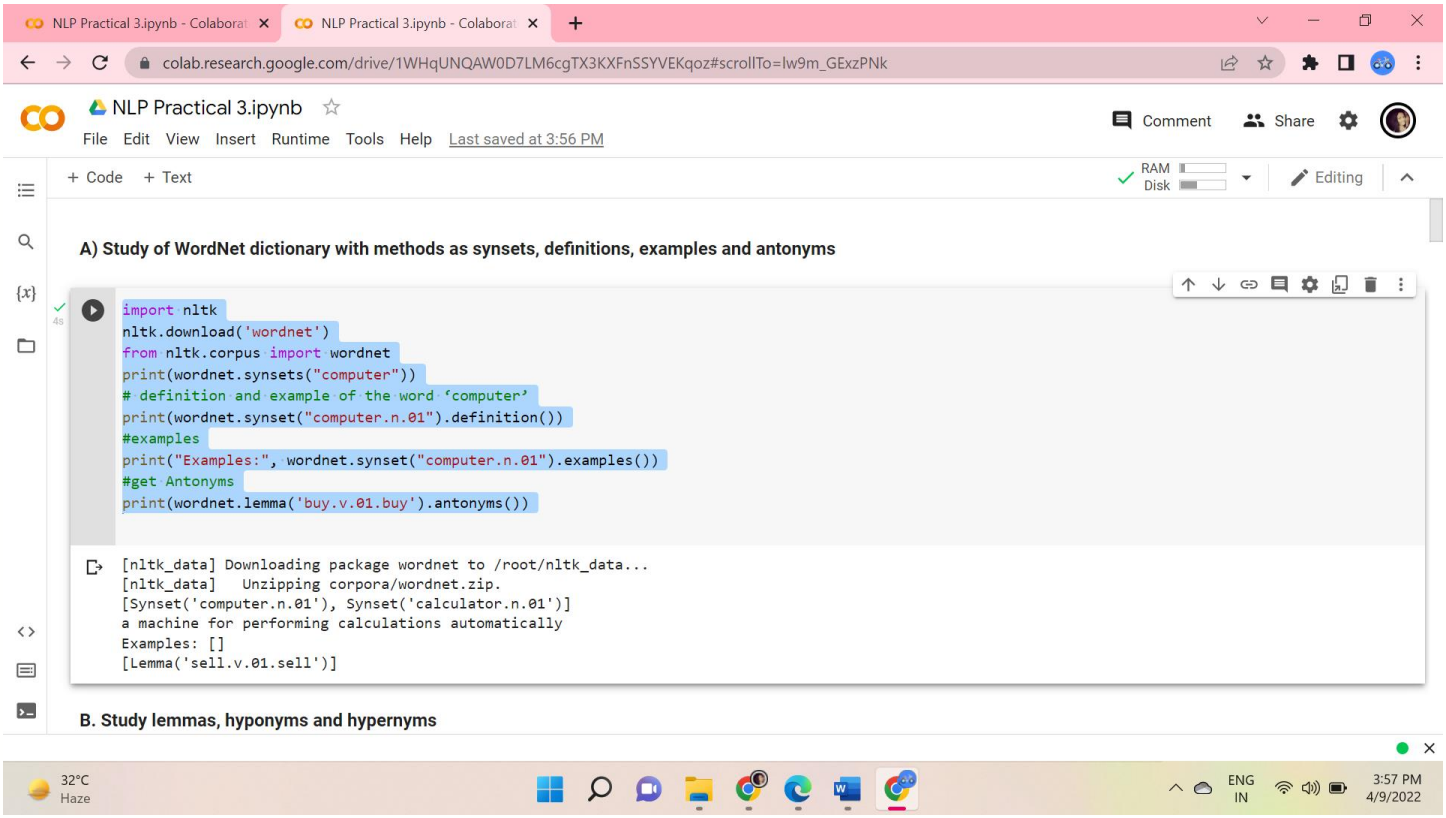



```
what -appears in the corpus
is -appears in the corpus
my -appears in the corpus
name -appears in the corpus
output
---------
what is my name
Score 4.0
```

Practical No. 03

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

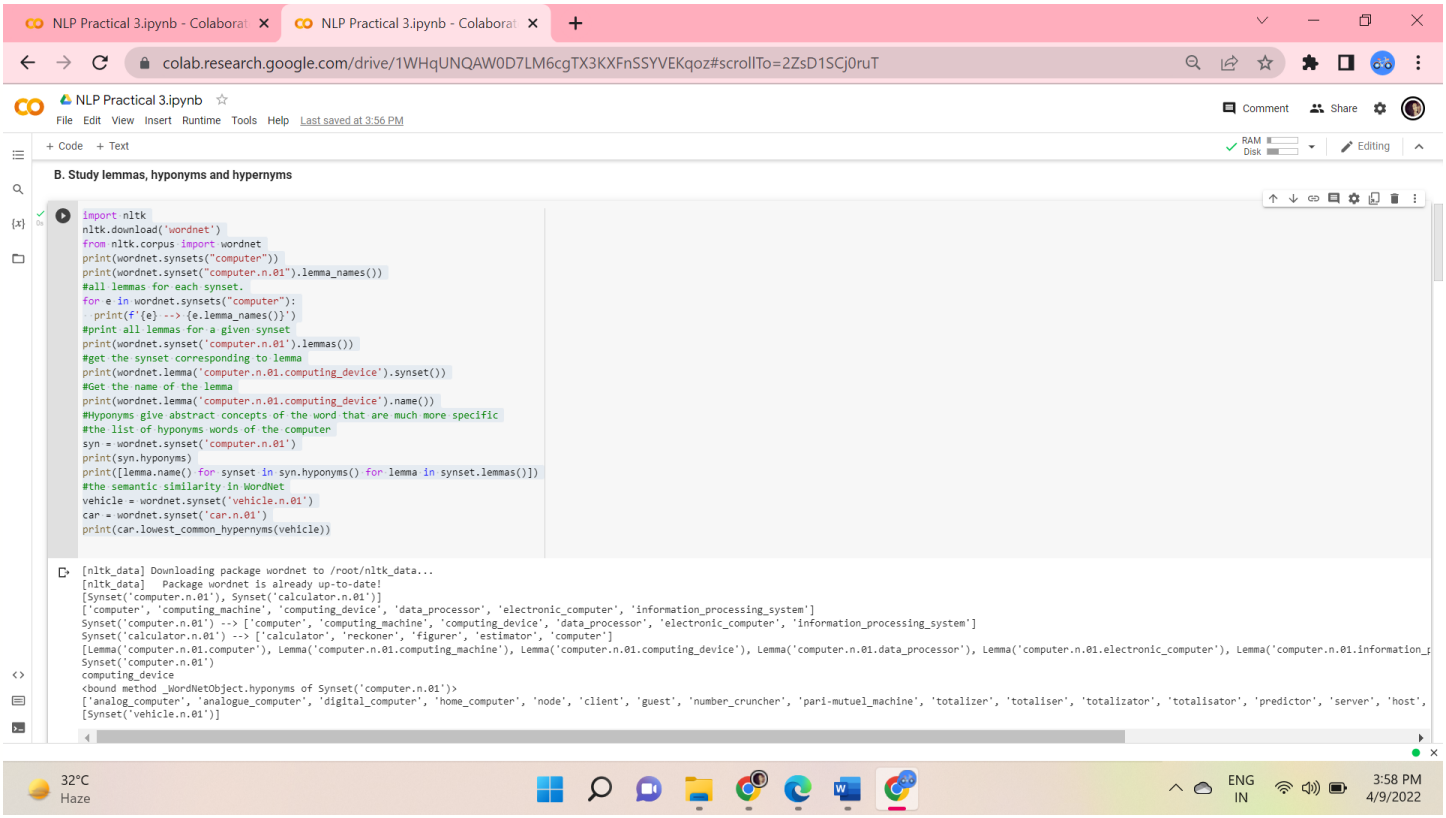
```
import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet
print(wordnet.synsets("computer"))
# definition and example of the word 'computer'
print(wordnet.synset("computer.n.01").definition())
#examples
print("Examples:", wordnet.synset("computer.n.01").examples())
#get Antonyms
print(wordnet.lemma('buy.v.01.buy').antonyms())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



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))
```



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
[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)
```

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

```
#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.
frozenset({'meanwhile', 'these', 'fill', 'made', 'few', 'your', 'someone', 'eleven', 'whole', 'anyhow', 'beforehand', 'perhaps', 'per', 'whenever', 'only', 'call', 'six', 'fifteen', 'but', 'herself', 'anywhere', 'hereupon', 'fify', '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', '.']
```

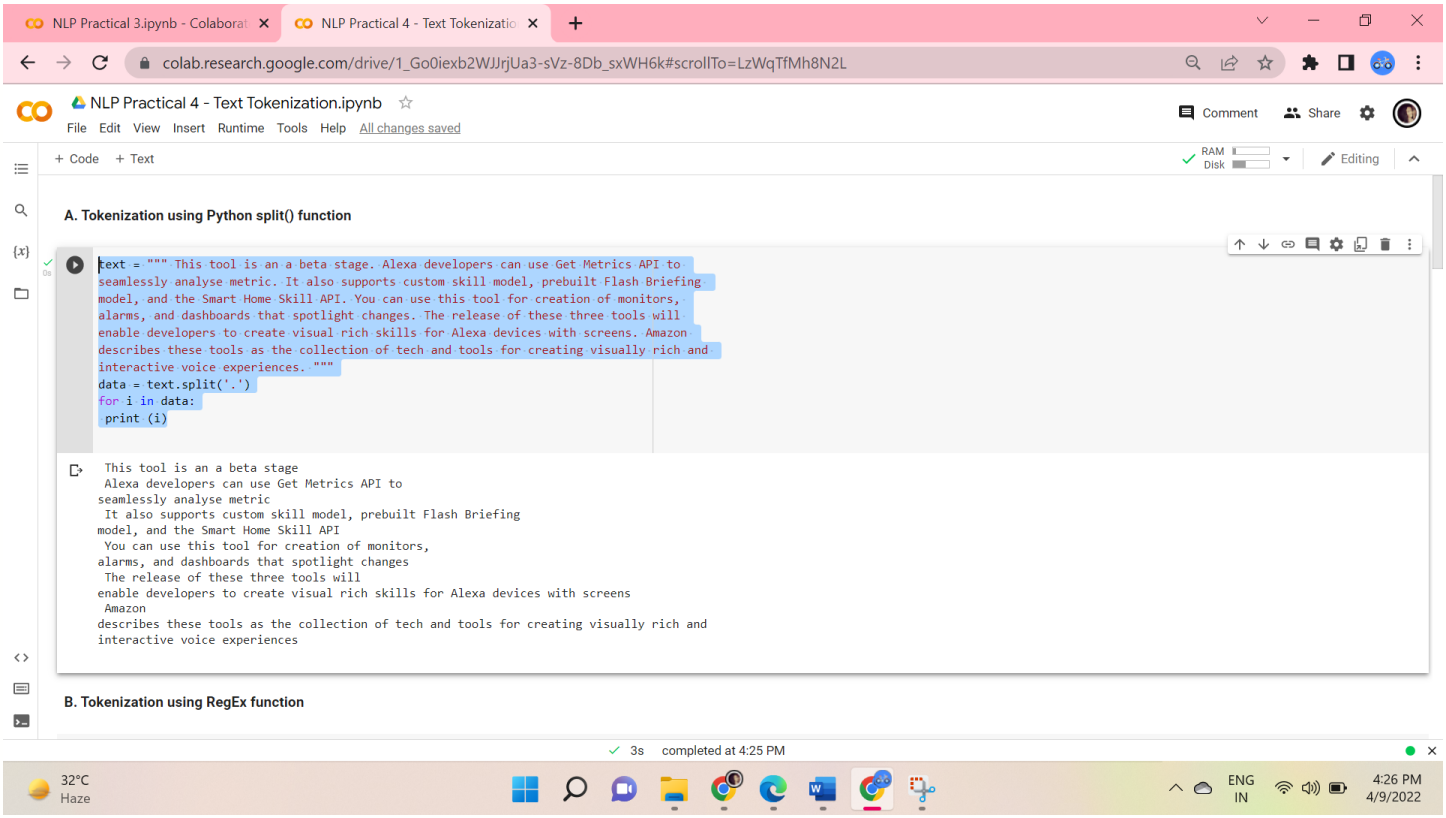
```
#remove 'not' from stop word collection
all_stopwords.remove('not')
tokens_without_sw = [word for word in text_tokens if not word in all_stopwords]
print(tokens_without_sw)
```

```
['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 Briefi
ng
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)
```



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))
```

```
!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))
```

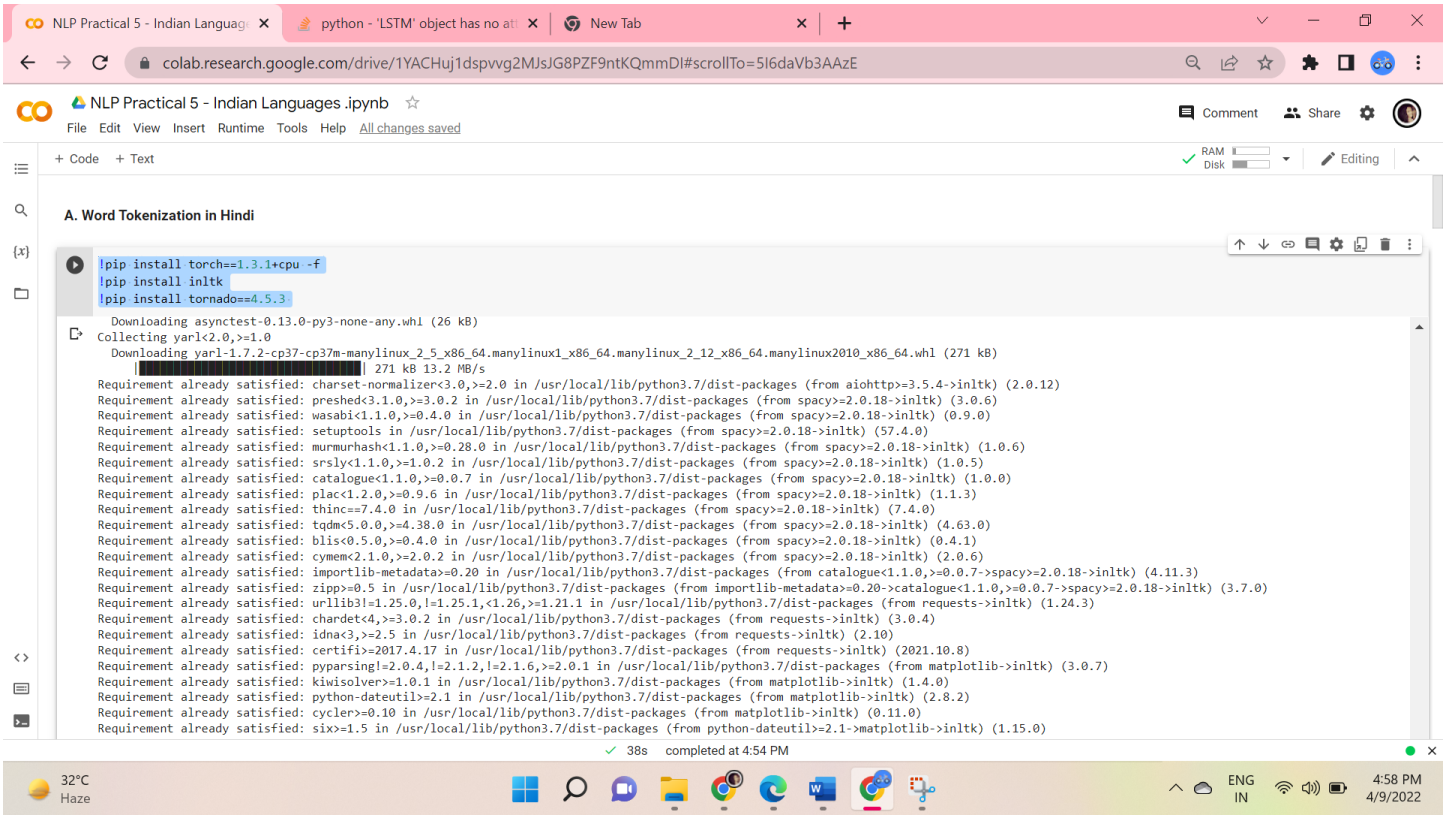
```
Requirement already satisfied: gensim in /usr/local/lib/python3.7/dist-packages (3.6.0)
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/dist-packages (from gensim) (5.2.1)
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.7/dist-packages (from gensim) (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

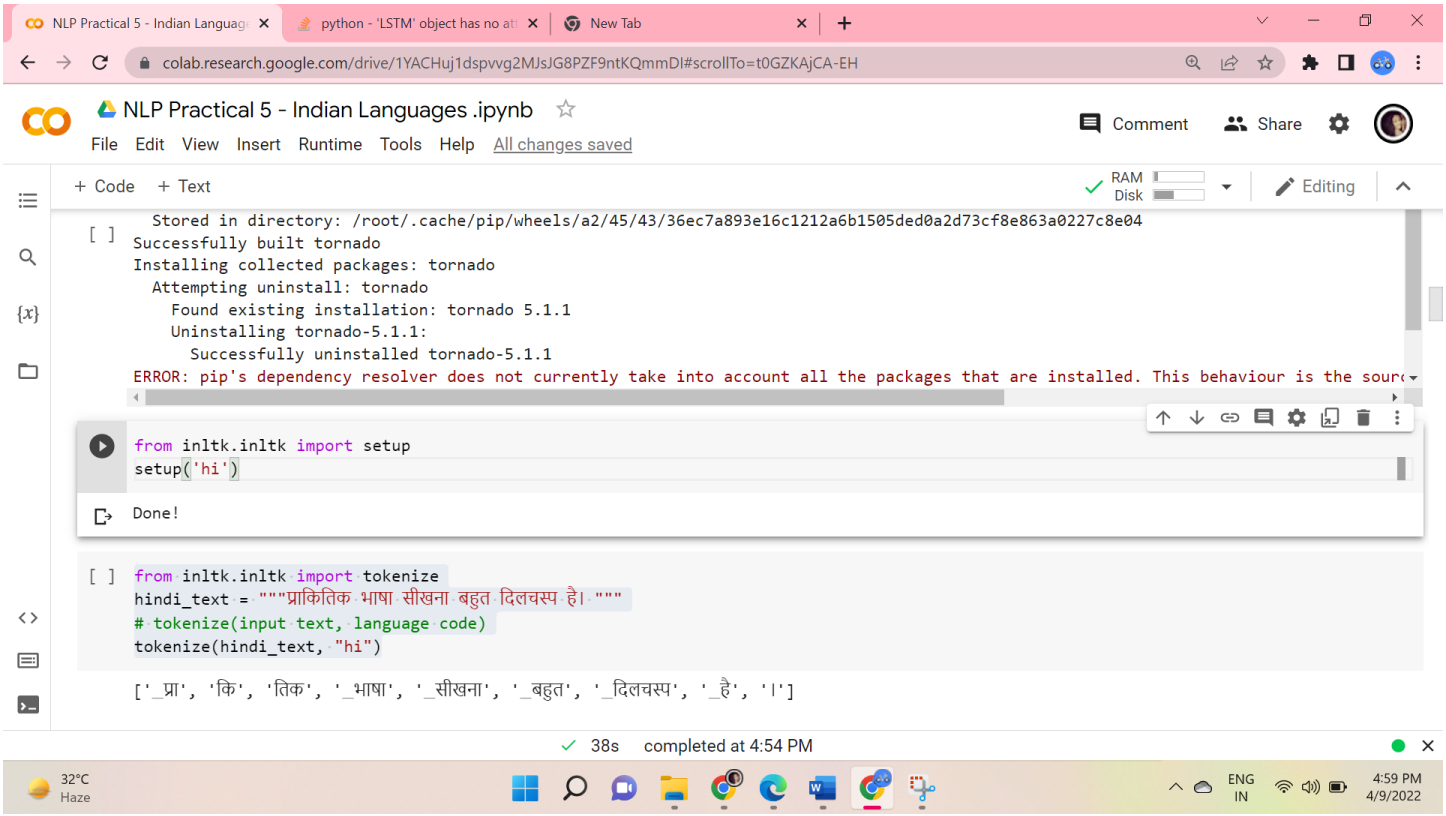
A. Word Tokenization in Hindi

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




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

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



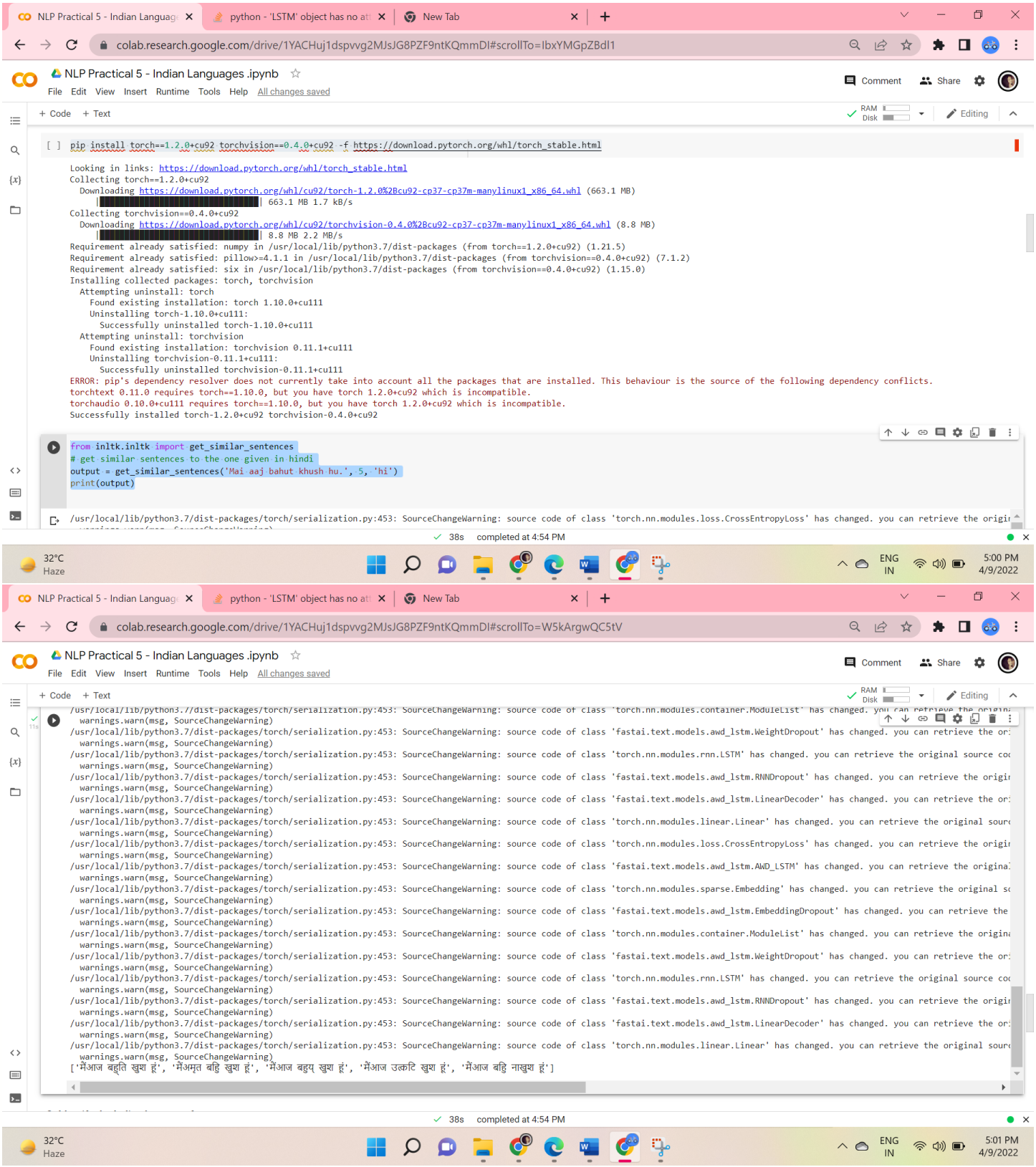
B) Generate similar sentences from a given Hindi text input

```
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 बहुि खुश हूं', 5, 'hi')
print(output)
```

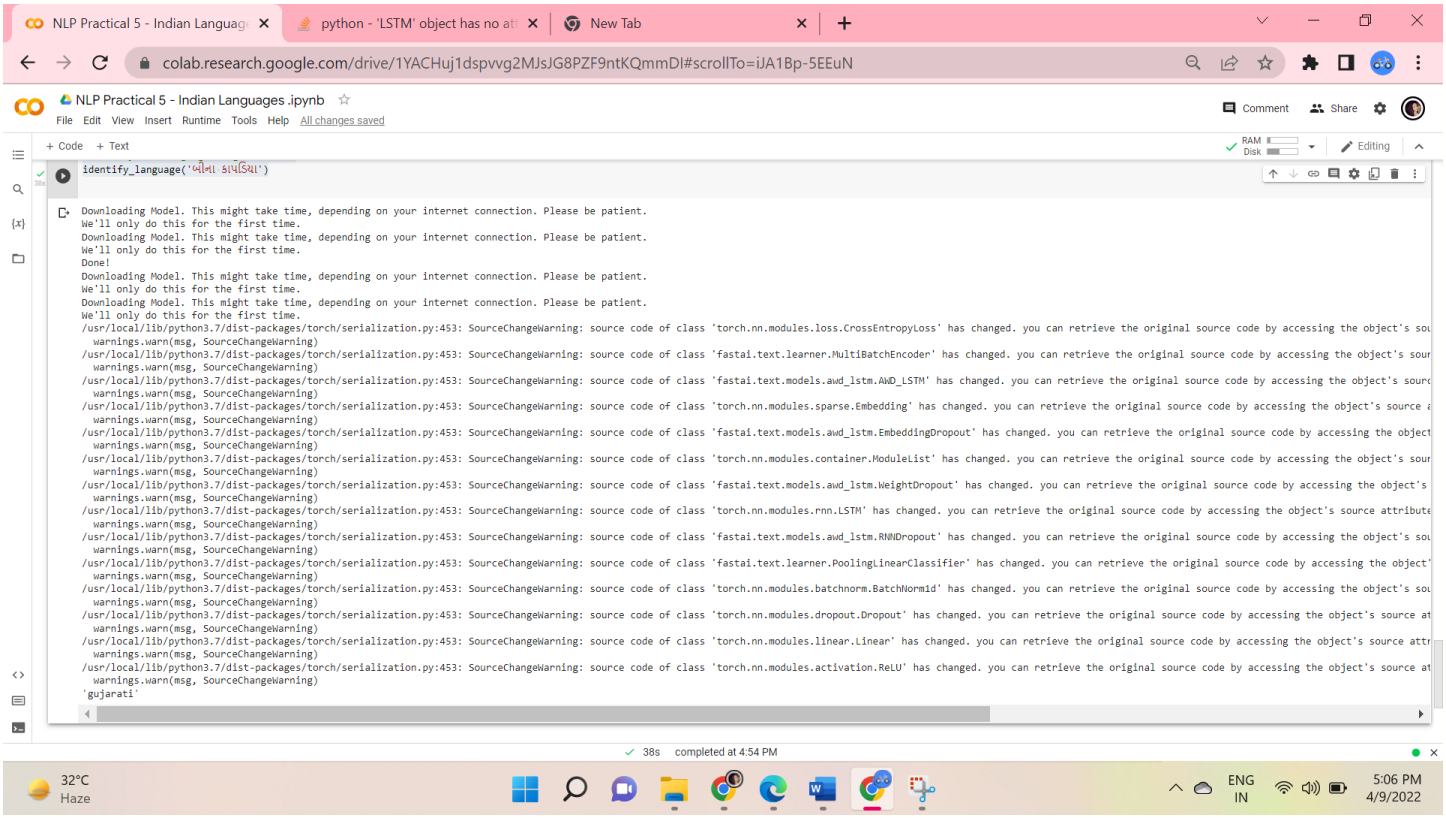


C) Identify the Indian language from a text

```
!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 Lnaguage of given text
identify_language('બીના કાપડિયા')
```

```
gujarati
```



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

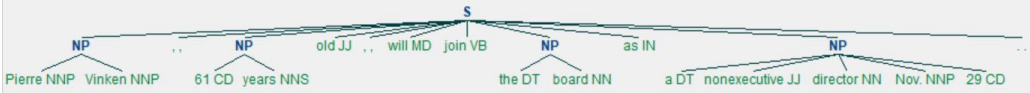
31°C Haze

ENG IN 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.
```

NLTK

File Zoom



31°C Haze

ENG IN 5:42 PM 4/9/2022

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

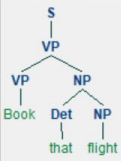
```
import nltk
nltk.download('punkt')
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
S -> VP
 VP -> VP NP
 NP -> Det NP
 Det -> 'that'
 NP -> singular Noun
 NP -> 'flight'
 VP -> 'Book'
""")
sentence = "Book that flight"
for index in range(len(sentence)):
 all_tokens = tokenize.word_tokenize(sentence)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

Ln: 14 Col: 34


```
>>>
================================ 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))))
```

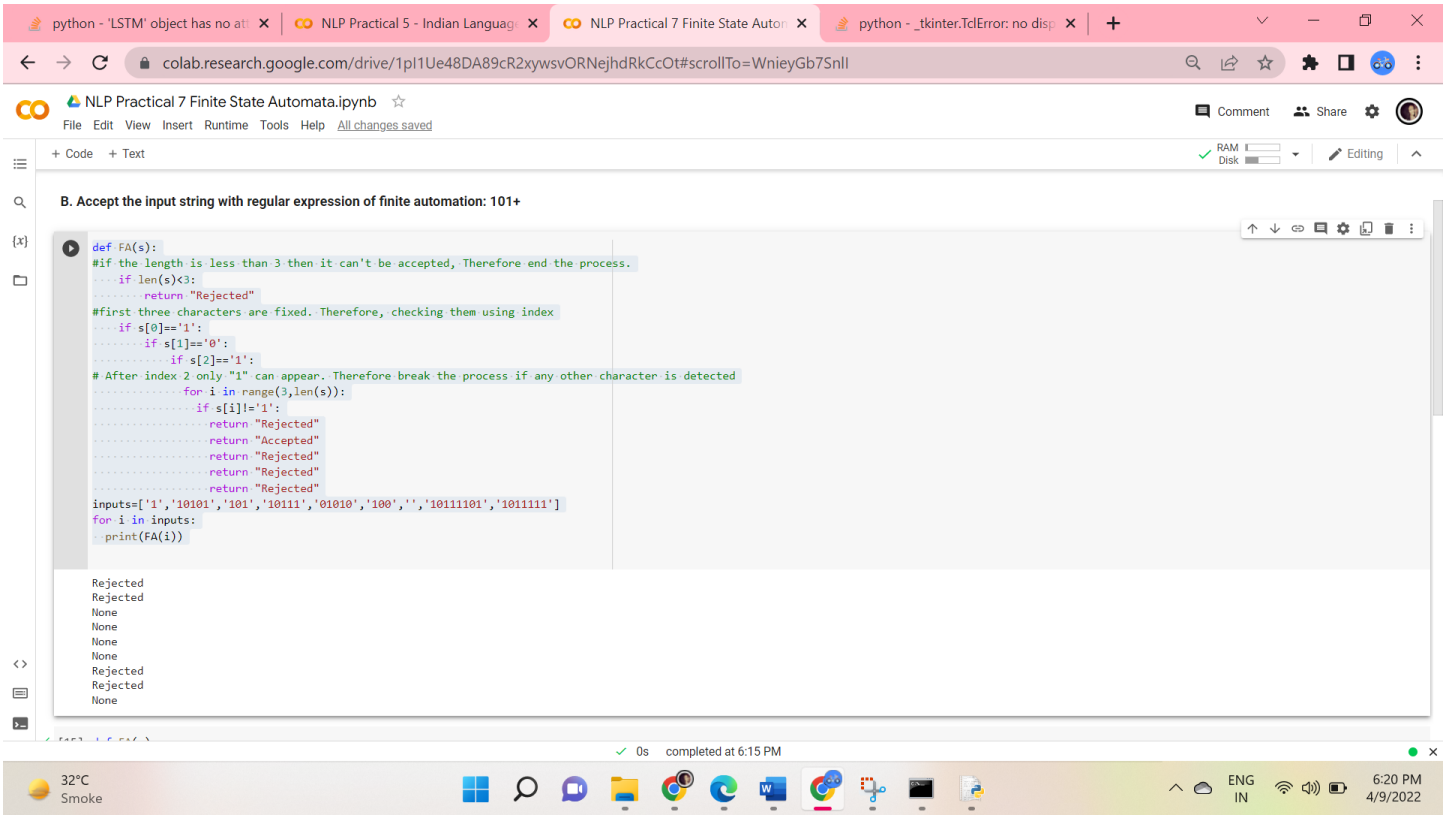
NLTK

File Zoom



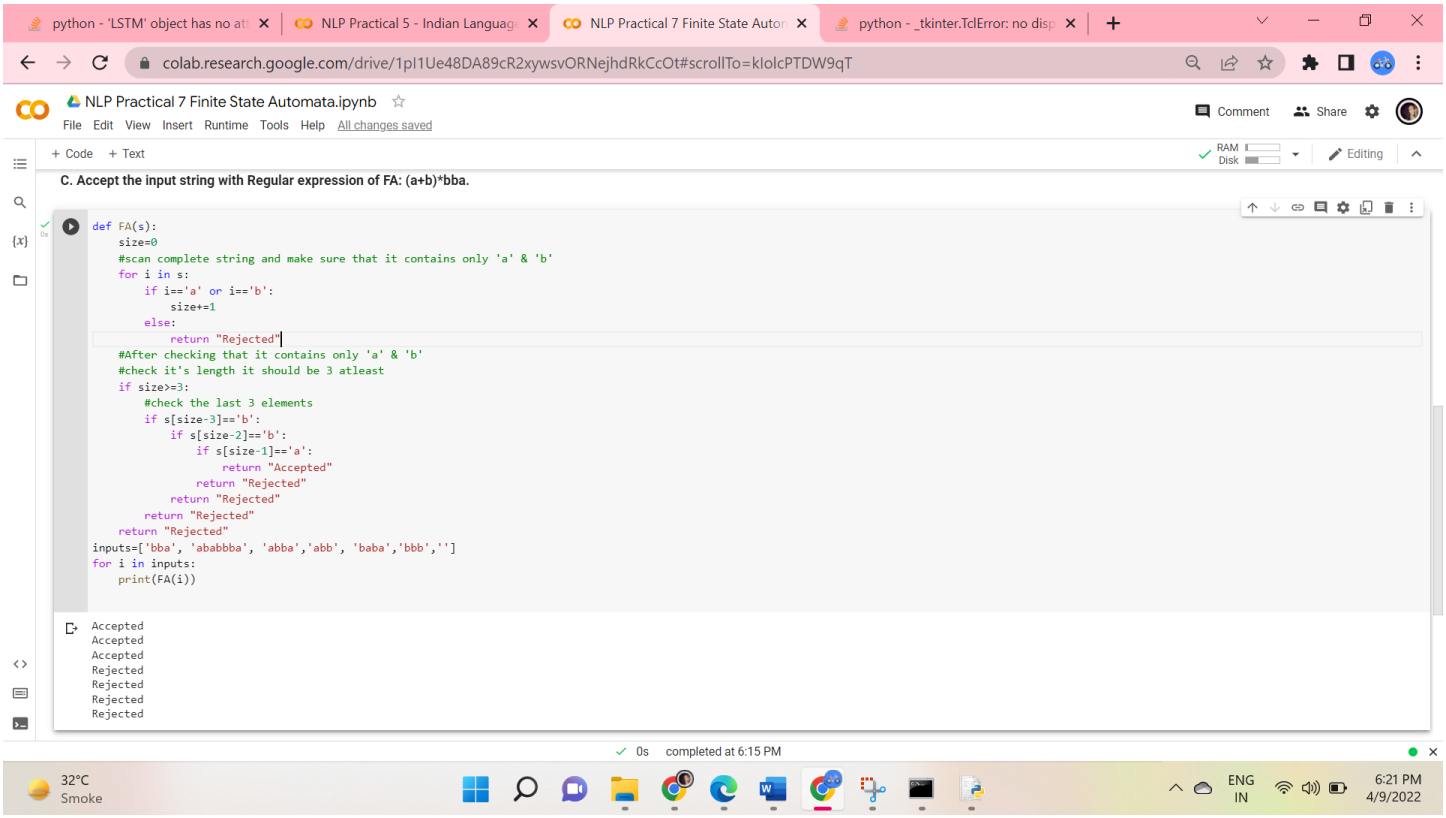
B) Accept the input string with regular expression of finite automation: 101+

```
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 charact
er 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))
```



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))
```



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

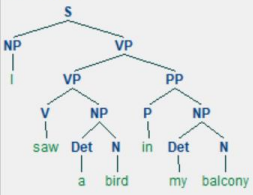
```
import nltk
from nltk import tokenize
grammar1 = nltk.CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'a' | 'my'
N -> 'bird' | 'balcony'
V -> 'saw'
P -> 'in'
""")
sentence = "I saw a bird in my balcony"
for index in range(len(sentence)):
    all_tokens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

Ln: 17 Col: 0


```
>>>
================================ 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)))))
```

NLTK

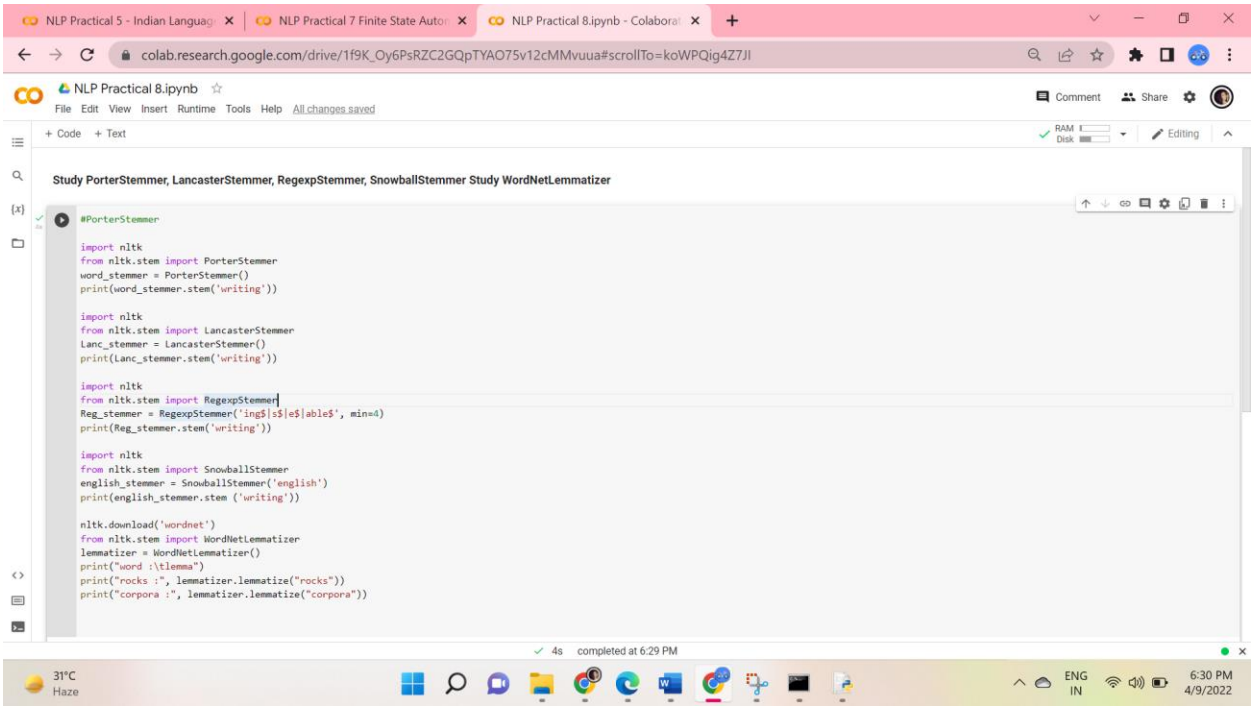
File Zoom



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"))
```




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

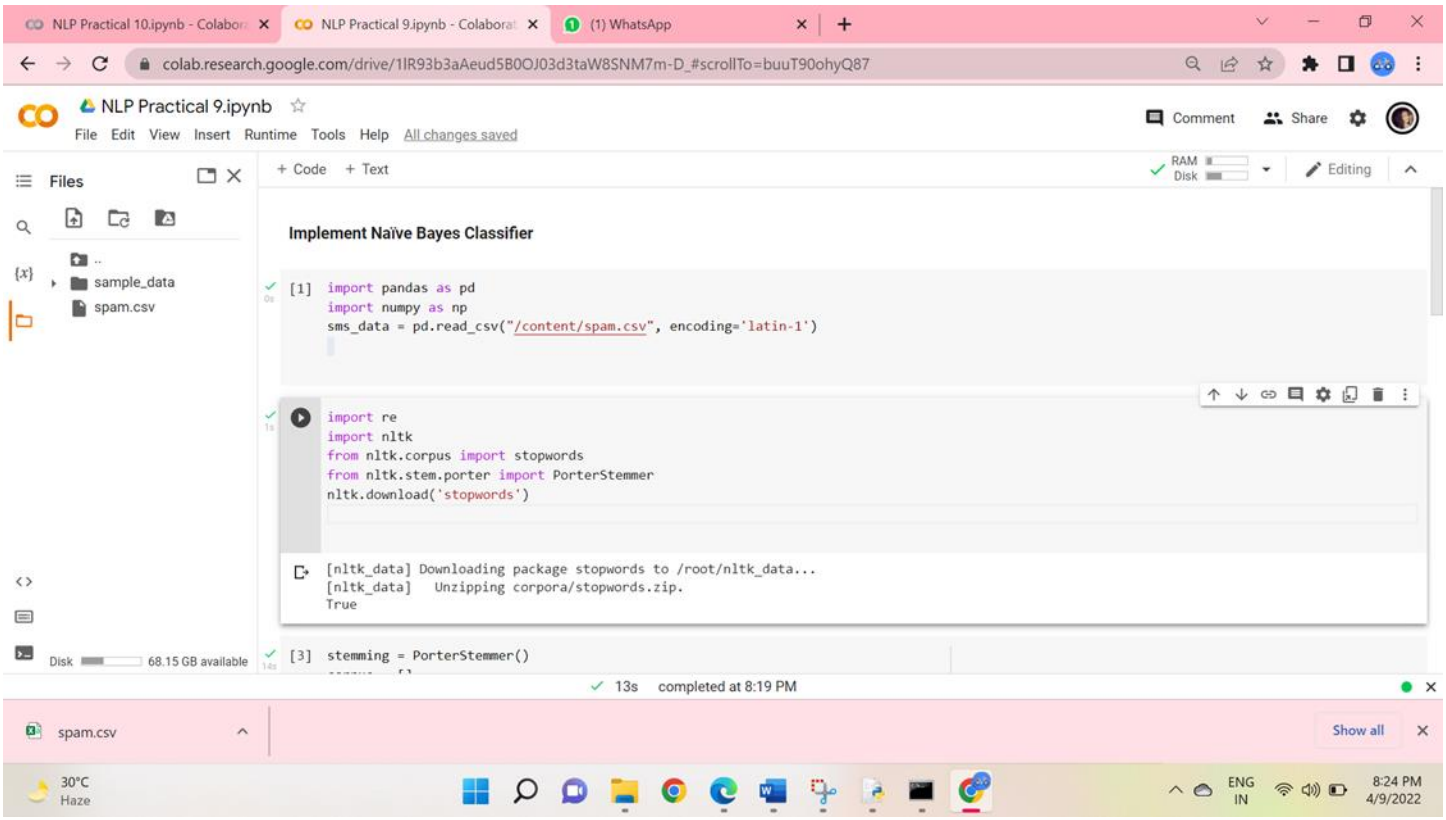
Practical No. 09

Implement Naïve Bayes Classifier

Code:

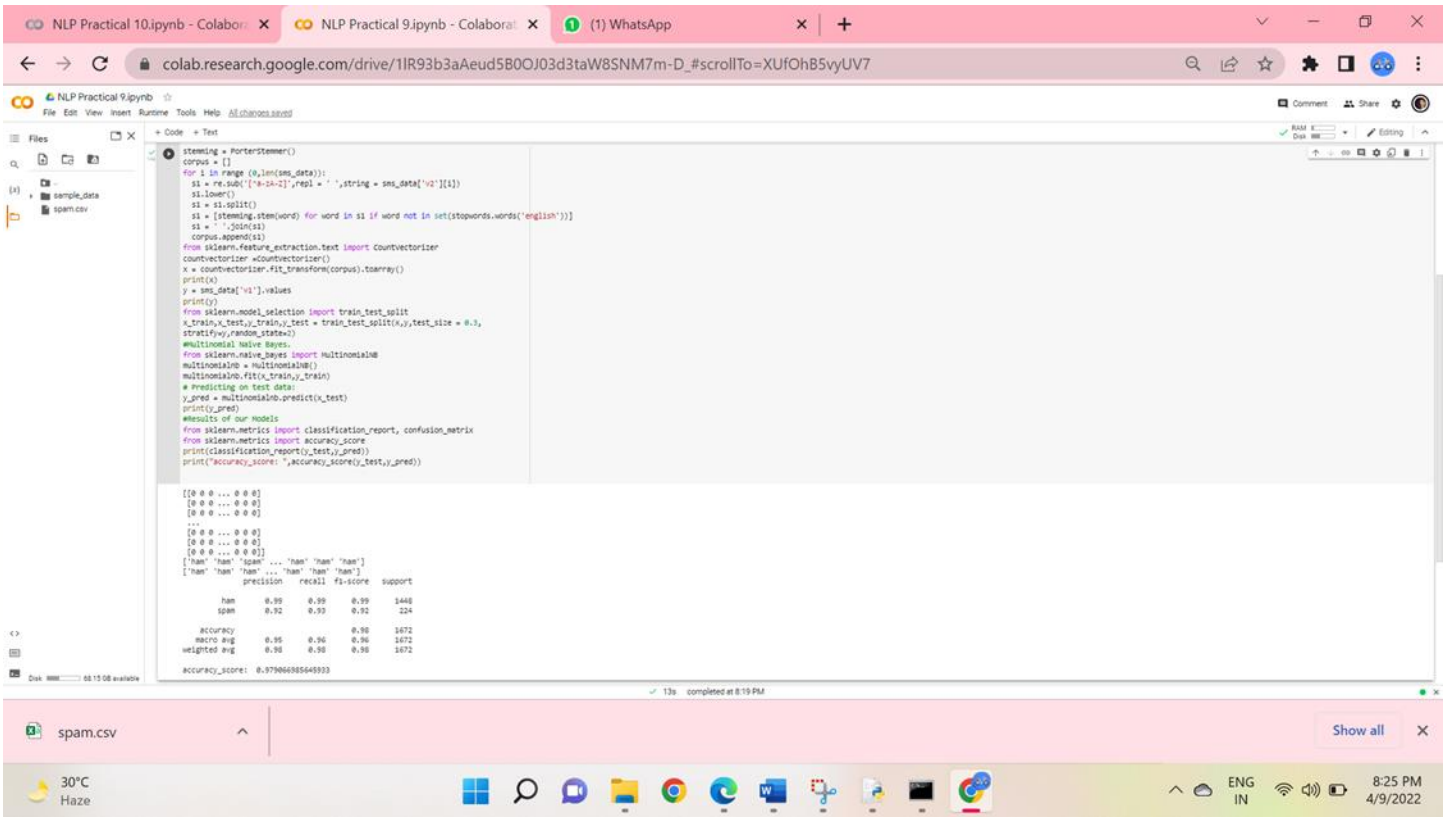
```
import pandas as pd

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')
```




```
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 10.ipynb - Colabor
NLP Practical 9.ipynb - Colaborat
(1) WhatsApp
colab.research.google.com/drive/1lR93b3aAeud5B0OJ03d3taW8SNM7m-D_#scrollTo=XUfOhB5vyUV7
NLP Practical 9.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
sample_data
spam.csv
+ Code + Text
Editing
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 Naive 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))
[[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]
[0 0 0 ... 0 0 0]]
['ham' 'ham' 'spam' ... 'ham' 'ham' 'ham']
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
precision recall f1-score support
ham 0.99 0.99 0.99 1448
spam 0.92 0.93 0.92 224
accuracy 0.98 1672
macro avg 0.95 0.96 0.96 1672
weighted avg 0.98 0.98 0.98 1672
accuracy_score: 0.9790669856459331
13s completed at 8:19 PM
spam.csv
Show all
30°C
Haze
ENG
IN
8:25 PM
4/9/2022

Practical no 10

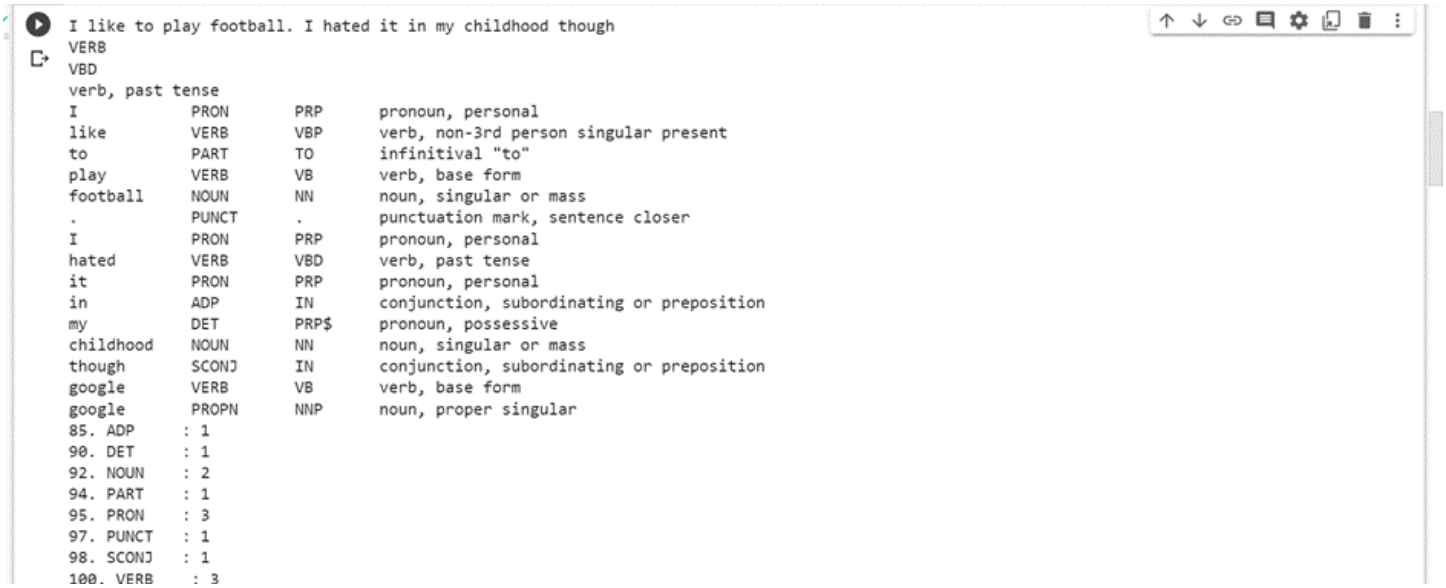
A. Speech Tagging

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:{8}}: {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})
```



```
I like to play football. I hated it in my childhood though
VERB
VBD
verb, past tense
I           PRON      PRP     pronoun, personal
like        VERB      VBP     verb, non-3rd person singular present
to          PART      TO      infinitival "to"
play        VERB      VB      verb, base form
football    NOUN      NN      noun, singular or mass
.           PUNCT     .       punctuation mark, sentence closer
I           PRON      PRP     pronoun, personal
hated       VERB      VBD     verb, past tense
it          PRON      PRP     pronoun, personal
in          ADP       IN      conjunction, subordinating or preposition
my          DET       PRP$    pronoun, possessive
childhood   NOUN      NN      noun, singular or mass
though      SCONJ     IN      conjunction, subordinating or preposition
google      VERB      VB      verb, base form
google      PROPN     NNP     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', 'Relianc
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')>
```