Experiment 2

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

AIM:

1. Generate word forms from root and suffix information.
2. Understanding the morphology of a word by the use of the Add-Delete table

THEORY:

Stemming:

Stemming is basically removing the suffix from a word and reducing it to its root word. It also involves producing morphological variants of a root/base word.

For example: “Flying” is a word and its suffix is “ing”, if we remove “ing” from “Flying” then we will get the base word or root word which is “Fly”. We use these suffixes to create a new word from the original stem word.

Sometimes spelling may also change in order to make a new word.

1. beauty, duty + -ful → beautiful, dutiful (-y changes to i)
2. heavy, ready + -ness → heaviness, readiness (-y changes to i)

The main aim of stemming is to reduce the inflectional forms of each word into a common base word or root word or stem word. Inflection is a process of word formation, in which a word is modified to express different grammatical categories such as tense, case, voice, aspect, person, number, gender, mood, animacy, and definiteness

Errors in Stemming:

There are mainly two errors in stemming – Overstemming and Understemming.

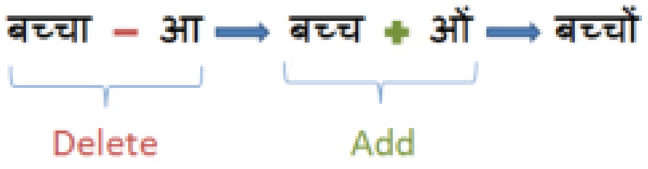
1. Overstemming occurs when two words are stemmed to the same root that are of different stems. Over-stemming can also be regarded as false-positives.
2. Under-stemming occurs when two words are stemmed to the same root that are not of different stems. Under-stemming can be interpreted as false-negatives.

Morphology:

Morphology is the study of the way words are built up from smaller meaning bearing units i.e., morphemes. A morpheme is the smallest meaningful linguistic unit. For eg:

- “Played” has two morphemes play and -ed having information verb "play" and "past tense", so the given word is in the past tense form of the verb "play".
- Words can be analysed morphologically if we know all variants of a given root word. We can use an 'Add-Delete' table for this analysis.

Example:



Analysis of a word :

Played = play(root) + ed(suffix)

A linguistic paradigm is the complete set of variants of a given lexeme. These variants can be classified according to shared inflectional categories (eg: number, case etc) and arranged into tables.

Algorithm to get ‘playing’ from ‘played’:

1. Take Root ‘play’
2. Delete ‘ed’
3. Output ‘play’
4. Add ‘ing’ to output
5. Return ‘playing’
6. Therefore ‘ed’ is deleted and ‘ing’ is added to get ‘playing’

Paradigm Class:

A paradigm class contains the classes of lexemes i.e. the prototypical root and all the roots that fall in its class including the given root. Those words which decline or conjugate in exactly the same way, fall into one paradigm class.

The English verbs ‘PLAY’ and ‘LOOK’ have the following paradigm:

- play plays played played playing
- look looks looked looked looking

So they belong to the same class. But ‘PUSH’ since it differs in its present tense form i.e. it has ‘-es’ and not ‘- s’ falls in another class. Its paradigm is as follows:

- push pushes pushed pushed pushing

Add-Delete Rules for Generation and Analysis of words:

- Most of the morphological analyzers handle only inflectional morphology.
- The rules given here are for such inflectional processes only.

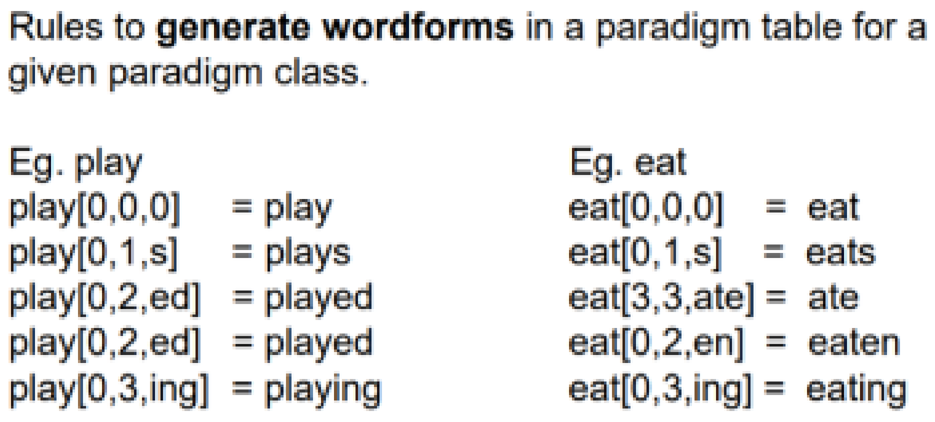
An add-delete rule would look like:

[n1, n2, xyz] where,

n1 = number of characters to delete from the end

n2 = number of characters to add at the end

xyz = the characters to add



Rules to **generate wordforms** in a paradigm table for a given paradigm class.

Eg. play		Eg. eat	
play[0,0,0]	= play	eat[0,0,0]	= eat
play[0,1,s]	= plays	eat[0,1,s]	= eats
play[0,2,ed]	= played	eat[3,3,ate]	= ate
play[0,2,ed]	= played	eat[0,2,en]	= eaten
play[0,3,ing]	= playing	eat[0,3,ing]	= eating

IDE USED: Jupyter Notebook

LIBRARIES USED:

Nltk:

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

pip install nltk

Pattern:

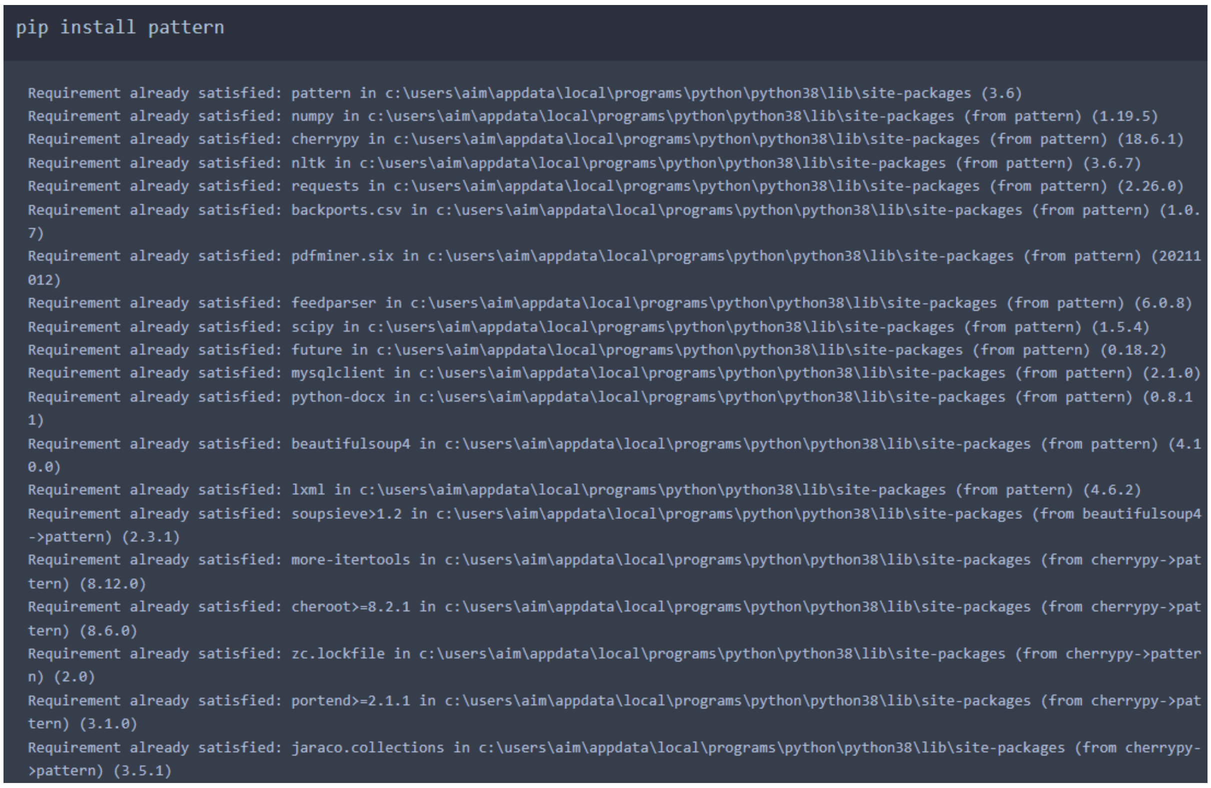
Pattern is a web mining module for Python. It has tools for –

- Data Mining: web services (Google, Twitter, Wikipedia), web crawler, HTML DOM parser

- Natural language Processing: part-of-speech taggers, n-gram search, sentiment analysis, WordNet.
- Machine Learning: vector space model, clustering, classification (KNN, SVM, Perceptron)
- Network Analysis: graph centrality and visualization.

It can be installed as :

pip install pattern3



```
pip install pattern

Requirement already satisfied: pattern in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (3.6)
Requirement already satisfied: numpy in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (1.19.5)
Requirement already satisfied: cherrypy in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (18.6.1)
Requirement already satisfied: nltk in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (3.6.7)
Requirement already satisfied: requests in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (2.26.0)
Requirement already satisfied: backports.csv in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (1.0.
7)
Requirement already satisfied: pdfminer.six in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (20211
012)
Requirement already satisfied: feedparser in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (6.0.8)
Requirement already satisfied: scipy in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (1.5.4)
Requirement already satisfied: future in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (0.18.2)
Requirement already satisfied: mysqlclient in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (2.1.0)
Requirement already satisfied: python-docx in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (0.8.1
1)
Requirement already satisfied: beautifulsoup4 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (4.1
0.0)
Requirement already satisfied: lxml in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from pattern) (4.6.2)
Requirement already satisfied: soupsieve>1.2 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from beautifulsoup4
->pattern) (2.3.1)
Requirement already satisfied: more-itertools in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from cherrypy->pat
tern) (8.12.0)
Requirement already satisfied: cheroot>=8.2.1 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from cherrypy->pat
tern) (8.6.0)
Requirement already satisfied: zc.lockfile in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from cherrypy->patter
n) (2.0)
Requirement already satisfied: portend>=2.1.1 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from cherrypy->pat
tern) (3.1.0)
Requirement already satisfied: jaraco.collections in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from cherrypy-
>pattern) (3.5.1)
```

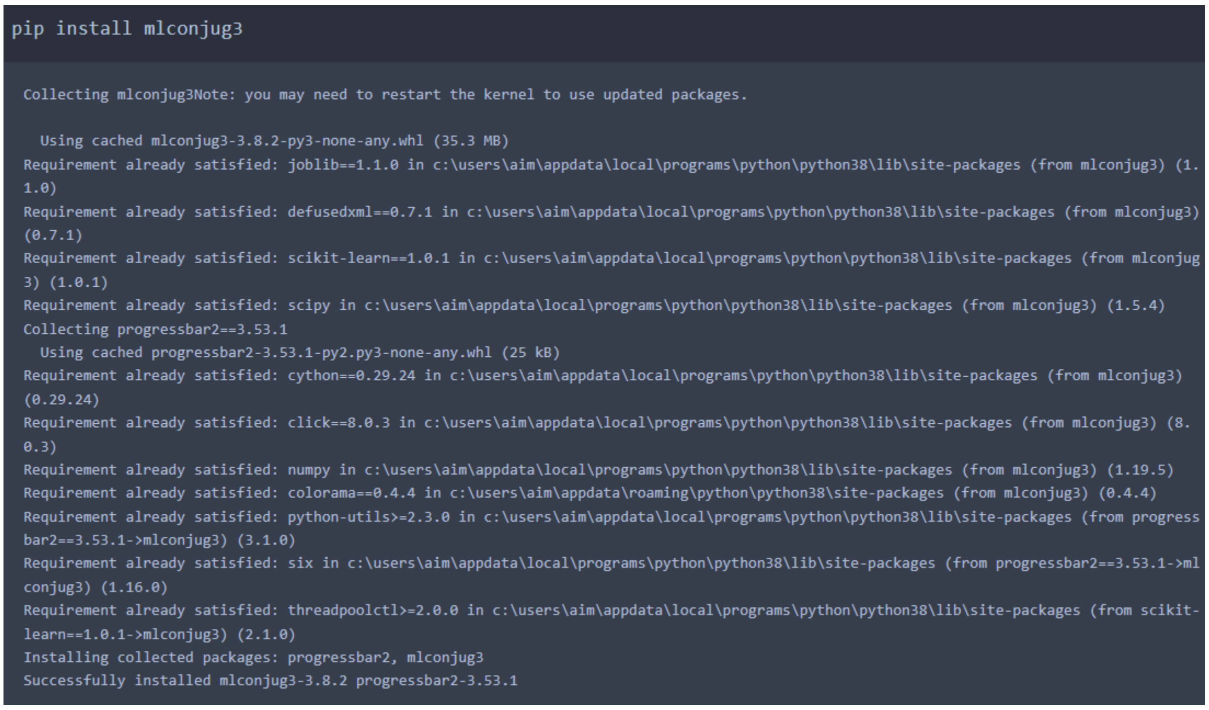
Mlconjug3:

A Python library to conjugate verbs in French, English, Spanish, Italian, Portuguese and Romanian (more soon) using Machine Learning techniques. Any verb in one of the supported language can be conjugated, as the module contains a Machine Learning model of how the verbs behave. Even completely new or made-up verbs can be successfully conjugated in this manner. The supplied pre-trained models are composed of:

- a binary feature extractor,
- a feature selector using Linear Support Vector Classification,
- a classifier using Stochastic Gradient Descent.

MLConjug3 uses scikit-learn to implement the Machine Learning algorithms. It can be installed as :

pip install mlconjug3



```
pip install mlconjug3
```

```
Collecting mlconjug3Note: you may need to restart the kernel to use updated packages.

  Using cached mlconjug3-3.8.2-py3-none-any.whl (35.3 MB)
Requirement already satisfied: joblib==1.1.0 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from mlconjug3) (1.
1.0)
Requirement already satisfied: defusedxml==0.7.1 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from mlconjug3)
(0.7.1)
Requirement already satisfied: scikit-learn==1.0.1 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from mlconjug
3) (1.0.1)
Requirement already satisfied: scipy in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from mlconjug3) (1.5.4)
Collecting progressbar2==3.53.1
  Using cached progressbar2-3.53.1-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: cython==0.29.24 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from mlconjug3)
(0.29.24)
Requirement already satisfied: click==8.0.3 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from mlconjug3) (8.
0.3)
Requirement already satisfied: numpy in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from mlconjug3) (1.19.5)
Requirement already satisfied: colorama==0.4.4 in c:\users\aim\appdata\roaming\python\python38\site-packages (from mlconjug3) (0.4.4)
Requirement already satisfied: python-utils>=2.3.0 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from progress
bar2==3.53.1->mlconjug3) (3.1.0)
Requirement already satisfied: six in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from progressbar2==3.53.1->ml
conjug3) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\aim\appdata\local\programs\python\python38\lib\site-packages (from scikit-
learn==1.0.1->mlconjug3) (2.1.0)
Installing collected packages: progressbar2, mlconjug3
Successfully installed mlconjug3-3.8.2 progressbar2-3.53.1
```

PROCEDURE:

1. Generate word forms from root and suffix information

   - Import all the packages and modules required and declared a list ‘valid_list’ that contains all the valid English words.
   - Take input of the root word from user
   - Take input of the suffix from user
   - In the if statement if by simply concatenating the word and suffix a valid word present in the ‘valid_list’ is produced then the obtained word is valid and hence printed
   - If the above if condition is false then the elif statements are executed.
   - In case the word and suffix combination is not valid then the output is printed as ‘Valid Word not Found’ or ‘Not able to find the word’

2. Understanding the morphology of a word by the use of the Add-Delete table

   - Imported all the packages and modules required

- The table.txt file contains the paradigm class and also created the list of all valid English words – ‘valid_list’.
- Took the sentence as an input from the user.
- Used tokenization to get the words
- Used pos_tag to only categorize the verbs
- Used the lemmatizer to store root of the word.
- Applied the paradigm class present in the table.txt to find the required conjugations for the list of verbs and checked that the validity of the final wordand then added it to the final list.
- mlconjug package is used to get the conjugation of the verb and the generated conjugations are added to the final list
- Printed the entire generated conjugation table of given verb
- The same procedure is repeated with Corpus genesis

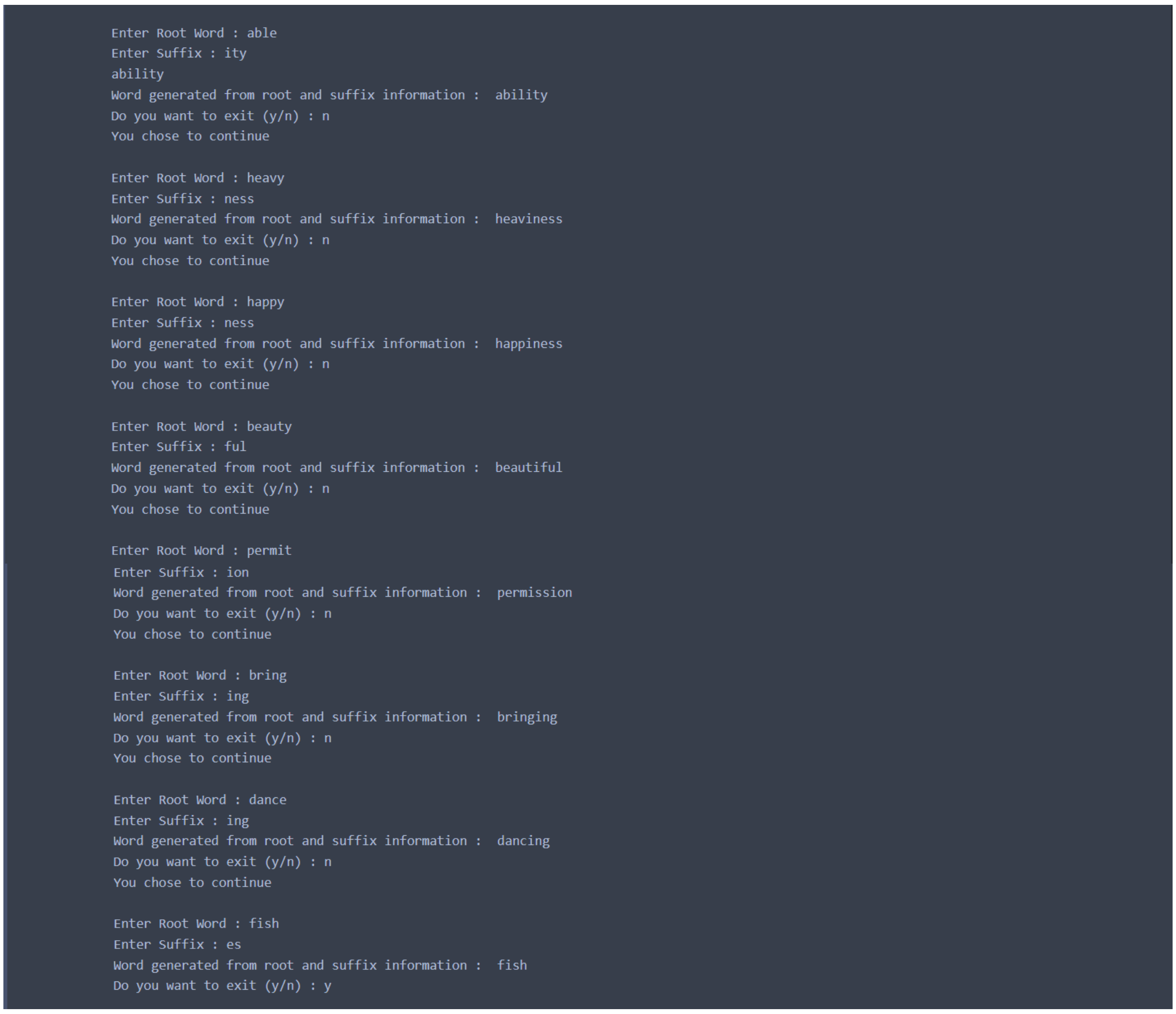
PART 1 CODE:

```
import nltk
# Dowloading Packages
# nltk.download('wordnet')
# nltk.download('words')
from nltk.corpus import words
from nltk.corpus import wordnet as wn
# Dowloading Pattern Module
# pip install pattern
from pattern.en import pluralize, singularize

valid_list = list(wn.words()) + words.words()
# Defining Vowels
vowel=['a','e','i','o','u']
while(1):
    word= input('Enter Root Word : ')
    suffix = input('Enter Suffix : ')
    #  When  word  form  by  joing  word  and  suffix  is  present  in
valid_list
    if word+suffix in valid_list:
        word+=suffix
        print('Word  generated  from  root  and  suffix  information  :
',word)
```

```
    else:
        if suffix=="s" or suffix=="es":
            if pluralize(word) in valid_list:
                print(pluralize(word))
            else:
                print('Valid Word not found')
        elif word[-1]=="y" and word[-2] not in vowel:
            word=word[0:-1]+"i"+suffix
            if word in valid_list:
                print('Word   generated   from   root   and   suffix
information : ',word)
            else:
                print('Valid Word not found')
        elif word[-2:]=="le" and suffix=="ity":
            word=word[0:-2]+"il"+suffix
            print(word)
    #         -le to -is
            if word in valid_list:
                print('Word   generated   from   root   and   suffix
information : ',word)
            else:
                print('Valid Word not found')
        elif word[-1]=='e' and suffix[0] in vowel:
            word=word[0:-1]+suffix
            if word in valid_list:
                print('Word   generated   from   root   and   suffix
information : ',word)
            else:
                print('Valid Word not found')
        elif word[-2:]=="ie" and suffix=="ing":
            word=word[0:-2]+"y"+suffix
            if word in valid_list:
                print('Word   generated   from   root   and   suffix
information : ',word)
            else:
                print('Valid Word not found')

        elif word[-1]=="t" and suffix=="ion":
            word=word[0:-1]+"ss"+suffix
    #         -t to -ss
            if word in valid_list:
                print('Word   generated   from   root   and   suffix
information : ',word)
```

```
            else:
                print('Valid Word not found')
        elif word[-2] in vowel and word[-1] not in vowel and word[-
1] not in ["x","y","z"]:
            word=word+word[-1]+suffix
            if word in valid_list:
                print('Word   generated   from   root   and   suffix
information : ',word)
            else:
                print('Valid Word not found')
        else:
            print('Not able to find the word')
    ans=input('Do you want to exit (y/n) : ')
    if ans=='y':
        break
    else:
        print('You chose to continue\n')
```

OUTPUT:



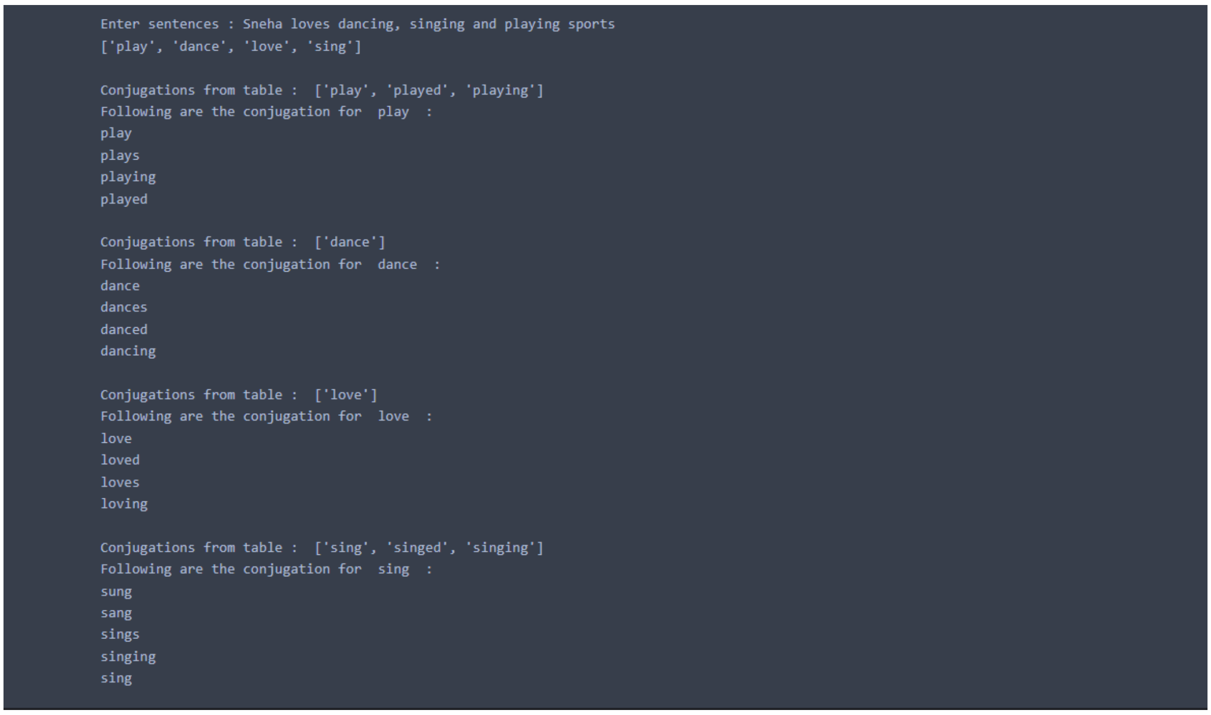
```
Enter Root Word : able
Enter Suffix : ity
ability
Word generated from root and suffix information :  ability
Do you want to exit (y/n) : n
You chose to continue

Enter Root Word : heavy
Enter Suffix : ness
Word generated from root and suffix information :  heaviness
Do you want to exit (y/n) : n
You chose to continue

Enter Root Word : happy
Enter Suffix : ness
Word generated from root and suffix information :  happiness
Do you want to exit (y/n) : n
You chose to continue

Enter Root Word : beauty
Enter Suffix : ful
Word generated from root and suffix information :  beautiful
Do you want to exit (y/n) : n
You chose to continue

Enter Root Word : permit
Enter Suffix : ion
Word generated from root and suffix information :  permission
Do you want to exit (y/n) : n
You chose to continue

Enter Root Word : bring
Enter Suffix : ing
Word generated from root and suffix information :  bringing
Do you want to exit (y/n) : n
You chose to continue

Enter Root Word : dance
Enter Suffix : ing
Word generated from root and suffix information :  dancing
Do you want to exit (y/n) : n
You chose to continue

Enter Root Word : fish
Enter Suffix : es
Word generated from root and suffix information :  fish
Do you want to exit (y/n) : y
```

PART 2 CODE:

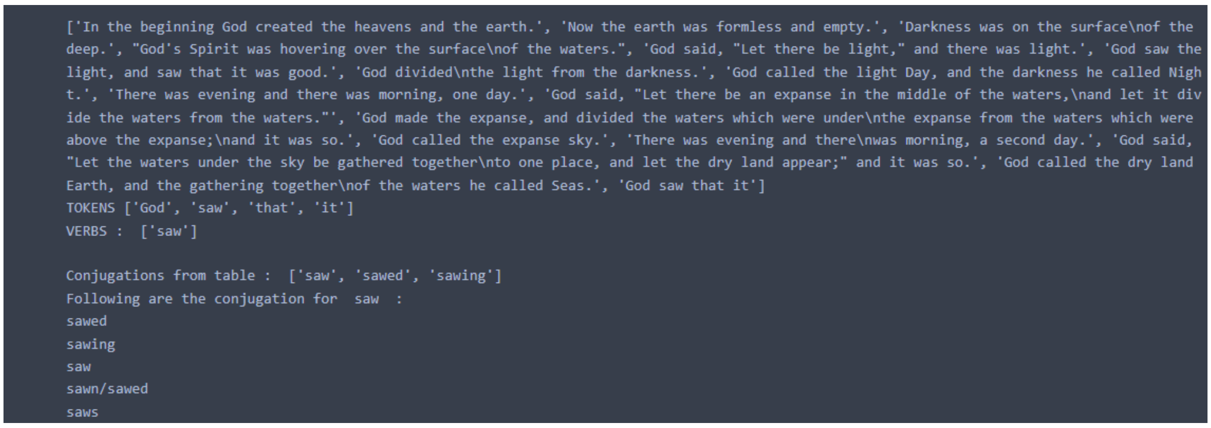
```
import nltk
from nltk.corpus import wordnet as wn
from nltk.corpus import words
from nltk.stem import WordNetLemmatizer

# Installing mlconjug3 module
# pip install mlconjug3
import mlconjug3
default_conjugator = mlconjug3.Conjugator(language='en')
wordnet_lemmatizer = WordNetLemmatizer()
valid_list = list(wn.words()) + words.words()
# Input sentence from the user
sente=input('Enter sentences : ')
# Used tokenization to get the words
```

```
tokens=nltk.word_tokenize(sente)
verbs=[]
# Used pos_tag to only categorize the verbs
tagged = nltk.pos_tag(tokens)
# Used the lemmatizer to store root of the word
for i in tagged:
    if "V"==i[1][0]:
        verbs.append(wordnet_lemmatizer.lemmatize(i[0],'v'))
verbs=list(set(verbs))
print(verbs)
finwords=[]
for word in verbs:
    allwords=[]
    f=open("table.txt","r")
    for x in f:
        a=x.find(']')
        if a!=-1:
            if x[a-1]=='0':
                allwords.append(word)
            else:
                wo=word[0:(len(word)-int(x[1]))]+x[5:a]
                if        wo        in        valid_list        and
wordnet_lemmatizer.lemmatize(wo,'v')==word:
                    allwords.append(wo)
        else:
            if not len(allwords):continue
            else:break
    print("\nConjugations from table : ",allwords)
# mlconjug package is used to get the conjugation of the verb
# the generated conjugations are added to the final list
    test_verb = default_conjugator.conjugate(word)
    conjugated_forms = test_verb.iterate()
    finverbs=set()
    for i in conjugated_forms:
        if len(i)==3:
            continue
        else:
            finverbs.add(i[3])
#     Printing the entire generated conjugation table of given verb
    print("Following are the conjugation for ",word," : ")
    #To check accuracy of conjugation by the table
    if set(allwords).union(finverbs) == finverbs:
        for i in finverbs:print(i)
    else:
        for i in finverbs:print(i)

1from nltk.corpus import genesis
from nltk.tokenize import sent_tokenize, word_tokenize
#Sentence Tokenizer
sentences=sent_tokenize(genesis.raw('english-web.txt')[:1000])
# print(sentences)
print(sentences)
for i in range(len(sentences)):
    tokens=word_tokenize(sentences[i])
print('TOKENS',tokens)
verbs=[]
# Used pos_tag to only categorize the verbs
tagged = nltk.pos_tag(tokens)
# Used the lemmatizer to store root of the word
for i in tagged:
    if "V"==i[1][0]:
        verbs.append(wordnet_lemmatizer.lemmatize(i[0],'v'))
verbs=list(set(verbs))
print('VERBS : ',verbs)

finwords=[]
for word in verbs:
    allwords=[]
    f=open("table.txt","r")
    for x in f:
        a=x.find(']')
        if a!=-1:
            if x[a-1]=='0':
                allwords.append(word)
            else:
                wo=word[0:(len(word)-int(x[1]))]+x[5:a]
                if        wo        in        valid_list        and
wordnet_lemmatizer.lemmatize(wo,'v')==word:
                    allwords.append(wo)
        else:
            if not len(allwords):continue
            else:break
    print("\nConjugations from table : ",allwords)
# mlconjug package is used to get the conjugation of the verb
# the generated conjugations are added to the final list
    test_verb = default_conjugator.conjugate(word)
    all_conjugated_forms = test_verb.iterate()
    finverbs=set()
    for i in all_conjugated_forms:
        if len(i)==3:
            continue
        else:
            finverbs.add(i[3])
#     Printing the entire generated conjugation table of given verb
    print("Following are the conjugation for ",word," : ")
    #To check accuracy of conjugation by the table
    if set(allwords).union(finverbs) == finverbs:
        for i in finverbs:
            print(i)
    else:
        for i in finverbs:
            print(i)
```

OUTPUT:



```
Enter sentences : Sneha loves dancing, singing and playing sports
['play', 'dance', 'love', 'sing']

Conjugations from table :  ['play', 'played', 'playing']
Following are the conjugation for  play  :
play
plays
playing
played

Conjugations from table :  ['dance']
Following are the conjugation for  dance  :
dance
dances
danced
dancing

Conjugations from table :  ['love']
Following are the conjugation for  love  :
love
loved
loves
loving

Conjugations from table :  ['sing', 'singed', 'singing']
Following are the conjugation for  sing  :
sung
sang
sings
singing
sing
```



```
['In the beginning God created the heavens and the earth.', 'Now the earth was formless and empty.', 'Darkness was on the surface\nof the
deep.', "God's Spirit was hovering over the surface\nof the waters.", 'God said, "Let there be light," and there was light.', 'God saw the
light, and saw that it was good.', 'God divided\nthe light from the darkness.', 'God called the light Day, and the darkness he called Nigh
t.', 'There was evening and there was morning, one day.', 'God said, "Let there be an expanse in the middle of the waters,\nand let it div
ide the waters from the waters."', 'God made the expanse, and divided the waters which were under\nthe expanse from the waters which were
above the expanse;\nand it was so.', 'God called the expanse sky.', 'There was evening and there\nwas morning, a second day.', 'God said,
"Let the waters under the sky be gathered together\nto one place, and let the dry land appear;" and it was so.', 'God called the dry land
Earth, and the gathering together\nof the waters he called Seas.', 'God saw that it']
TOKENS ['God', 'saw', 'that', 'it']
VERBS :  ['saw']

Conjugations from table :  ['saw', 'sawed', 'sawing']
Following are the conjugation for  saw  :
sawed
sawing
saw
sawn/sawed
saws
```

REFERENCES:

1. https://www.geeksforgeeks.org/python-stemming-words-with-nltk/
2. https://www.geeksforgeeks.org/introduction-to-stemming/
3. https://medium.com/@tusharsri/nlp-a-quick-guide-to-stemming-60f1ca5db49e
4. http://www.sau.ac.in/~mlta/studymaterial/17-12-2013/Winter_school_Morph-tutorial-RadhikaMamidi.pdf
5. https://youtu.be/yGKTphqxR9Q
6. https://www.youtube.com/watch?v=TqElkDSs3FE
7. https://pypi.org/project/pattern3/
8. https://pypi.org/project/mlconjug3/