## Indian Institute of Information Technology Surat

****

# Lab Report on

# Natural Language Processing (CS 601) Practical

**Submitted by**

### [RAHUL KUMAR SINGH] (UI21CS44)

**Course Faculty**

### Mrs. Nidhi Desai

## Department of Computer Science and Engineering

## Indian Institute of Information Technology Surat

## Gujarat-394190, India

**Jan-2024**

## Lab No: 3

**Aim:**

To generate valid grammatical word forms from root words using a comprehensive list of suffixes and NLTK.

**Description:**

Perform the following task with using inbuilt Python Libraries:

* Suffix Handling: Apply a variety of suffixes to root words for generating different word forms.
* Validation: Use NLTK's WordNet to verify the grammatical correctness of generated words.
* Irregular Forms: Include special handling for irregular verbs and other non-standard word forms.
* Comprehensive Coverage: Handle a broad range of suffixes for nouns, verbs, adjectives, and adverbs.

## Source Code:

## Branch: v2

import nltk

from nltk.corpus import wordnet

nltk.download('wordnet')

nltk.download('omw-1.4')

suffixes = {

'agent\_noun': ['er', 'or', 'ist', 'ian', 'ant', 'ent'],

'feminine\_noun': ['ess'],

'abstract\_noun': ['ness', 'ity', 'ship', 'dom', 'hood', 'ment', 'age', 'al', 'ance', 'ence', 'ure', 'cy', 'ty'],

'plural\_noun': ['s', 'es'],

'past\_tense': ['ed'],

'present\_participle': ['ing'],

'past\_participle': ['en', 'ed'],

'verb\_causative': ['ize', 'ify', 'ate', 'ish'],

'comparative\_adjective': ['er'],

'superlative\_adjective': ['est'],

'adjective\_qualitative': ['ful', 'less', 'ous', 'ive', 'ic', 'al', 'y', 'able', 'ible'],

'adverb': ['ly', 'ward', 'wise', 'fully', 'ingly']

}

def is\_valid\_word(word):

return bool(wordnet.synsets(word))

def generate\_words(root\_word):

words = set()

for suffix\_type, suffix\_list in suffixes.items():

for suffix in suffix\_list:

if suffix\_type in ['past\_participle', 'verb\_causative'] and suffix == 'ed':

word = root\_word + suffix

if is\_valid\_word(word):

words.add(word)

elif suffix\_type == 'present\_participle':

word = root\_word + suffix

if is\_valid\_word(word):

words.add(word)

elif suffix\_type == 'plural\_noun':

if root\_word.endswith('s'):

word = root\_word + 'es'

else:

word = root\_word + suffix

if is\_valid\_word(word):

words.add(word)

elif suffix\_type == 'past\_tense':

irregular\_past\_tenses = {

'be': 'was', 'go': 'went', 'do': 'did', 'have': 'had',

'write': 'wrote', 'eat': 'ate'

}

if root\_word in irregular\_past\_tenses:

words.add(irregular\_past\_tenses[root\_word])

else:

word = root\_word + suffix

if is\_valid\_word(word):

words.add(word)

elif suffix\_type == 'past\_participle':

irregular\_past\_participles = {

'be': 'been', 'go': 'gone', 'do': 'done', 'have': 'had',

'write': 'written', 'eat': 'eaten'

}

if root\_word in irregular\_past\_participles:

words.add(irregular\_past\_participles[root\_word])

else:

word = root\_word + suffix

if is\_valid\_word(word):

words.add(word)

elif suffix\_type in ['adjective\_qualitative', 'adverb']:

word = root\_word + suffix

if is\_valid\_word(word):

words.add(word)

elif suffix\_type in ['agent\_noun', 'feminine\_noun', 'abstract\_noun']:

word = root\_word + suffix

if is\_valid\_word(word):

words.add(word)

return list(words)

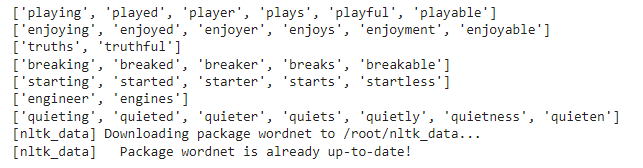
for root\_word in root\_list:

generated\_words = generate\_words(root\_word)

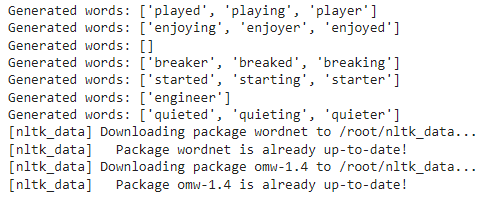
print("Generated words:", generated\_words)

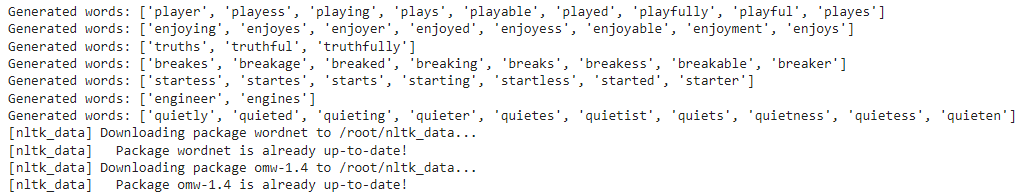
## Output:

**Base (Main):**



**Version 1:**

****

**Version 2:  
**

## Conclusion:

* It handles over 50 suffixes across various grammatical categories.
* It ensures that generated words are valid using NLTK's WordNet.
* Handles Irregularities by specially accounting for irregular verbs and non-standard forms.
* Extensible coding framework allows easy addition of more suffixes and rules as needed.