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):

```
['playing', 'played', 'player', 'plays', 'playful', 'playable']
['enjoying', 'enjoyed', 'enjoyer', 'enjoys', 'enjoyment', 'enjoyable']
['truths', 'truthful']
['breaking', 'breaked', 'breaker', 'breaks', 'breakable']
['starting', 'started', 'starter', 'starts', 'startless']
['engineer', 'engines']
['quieting', 'quieted', 'quieter', 'quiets', 'quietly', 'quietness', 'quieten']
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Version 1:

```
Generated words: ['played', 'playing', 'player']
Generated words: ['enjoying', 'enjoyer', 'enjoyed']
Generated words: []
Generated words: ['breaker', 'breaked', 'breaking']
Generated words: ['started', 'starting', 'starter']
Generated words: ['engineer']
Generated words: ['quieted', 'quieting', 'quieter']
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Version 2:

```
Generated words: ['player', 'playess', 'playing', 'plays', 'playable', 'played', 'playfully', 'playful', 'playes']
Generated words: ['enjoying', 'enjoyes', 'enjoyer', 'enjoyed', 'enjoyess', 'enjoyable', 'enjoyment', 'enjoys']
Generated words: ['truths', 'truthful', 'truthfully']
Generated words: ['breakes', 'breakage', 'breaked', 'breaking', 'breaks', 'breakess', 'breakable', 'breaker']
Generated words: ['startess', 'startes', 'starts', 'starting', 'startless', 'started', 'starter']
Generated words: ['engineer', 'engines']
Generated words: ['quietly', 'quieted', 'quieting', 'quieter', 'quietes', 'quietist', 'quiets', 'quietness', 'quietess', 'quieten']
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

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.