Experiment No. 4
Perform word generation for any given text.
Date of Performance:
Date of Submission:

Exp. No.: 4

Title: Perform word generation for any given text.

Theory:

Word generation in Natural Language Processing (NLP) refers to the process of creating or transforming words based on a given text. This typically involves generating new forms of a word, such as its inflected or derived forms, or even predicting the next word in a sequence. Word generation is used in tasks like **text generation, machine translation, and morphological analysis**.

There are two main types of word generation:

1. **Inflectional Word Generation**: Modifying a base word to fit different grammatical contexts (e.g., tense, number, person).
2. **Derivational Word Generation**: Forming new words by adding prefixes, suffixes, or altering the word root.

1. Inflectional Word Generation

Inflection is a morphological process that modifies a word to express grammatical properties such as:

- **Tense** (past, present, future)
- **Number** (singular, plural)
- **Person** (first, second, third)
- **Gender** (in gendered languages)
- **Case** (in languages with noun cases)

Example:

- Base word: **run**
- Inflected forms: **runs, running, ran**

2. Derivational Word Generation

Derivation is a process that changes the word category or its meaning by adding affixes (prefixes, suffixes).

- **Happy → Happiness** (adjective to noun)
- **Act → Actor**, **Action** (verb to noun)
- **Run → Runner** (verb to noun)

Word Generation Techniques

1. **Morphological Analysis and Generation**: By understanding the structure of words and breaking them into morphemes (smallest meaning units), we can generate new words from a base form.
2. **Contextual Word Prediction**: In tasks like language modeling, **next-word prediction** is a form of word generation where models like LSTMs, GRUs, or transformers generate words based on the context of the input text.

Methods for Word Generation in NLP

1. Finite State Transducers (FSTs)

Finite State Transducers (FSTs) are widely used for word generation, especially for languages with rich morphology. FSTs can convert a given word from one form (e.g., base form) to another (e.g., plural or past tense).

2. Neural Networks:

- **Recurrent Neural Networks (RNNs)**, **LSTMs**, and **GRUs** are commonly used for sequential word generation tasks, such as text generation or machine translation.
- **Transformers** (e.g., GPT, BERT) are now the state-of-the-art for word and text generation, allowing context-based word prediction.

3. Rule-Based Word Generation:

Linguistic rules can be used to generate inflections and derivations, especially for languages with fixed morphological patterns.

Code:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

# Download required NLTK data files
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

```
# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()
```

```
# Function to get wordnet POS tag
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None
```

```
# Example text
text = "The cats are running quickly."
```

```
# Tokenize the text
tokens = word_tokenize(text)
```

```
# POS tagging
pos_tags = nltk.pos_tag(tokens)
```

```
# Morphological analysis and word generation
analysis = []
generated_words = {}
```

```
for token, pos in pos_tags:
    wordnet_pos = get_wordnet_pos(pos) or wordnet.NOUN
```

```
    lemma = lemmatizer.lemmatize(token, pos=wordnet_pos)
    analysis.append((token, lemma, pos))
```

```
    # Generate words
    if wordnet_pos == wordnet.NOUN:
        generated_words[lemma] = [lemma, lemma + 's', lemma + "'s", lemma +
"s'"]
    elif wordnet_pos == wordnet.VERB:
        generated_words[lemma] = [lemma, lemma + 's', lemma + 'ed', lemma +
'ing']
    elif wordnet_pos == wordnet.ADJ:
        generated_words[lemma] = [lemma, lemma + 'er', lemma + 'est']
    elif wordnet_pos == wordnet.ADV:
        generated_words[lemma] = [lemma, lemma + 'ly']
    else:
        generated_words[lemma] = [lemma]
```

```
# Print results
print("Morphological Analysis:")
for token, lemma, pos in analysis:
    print(f"Token: {token}, Lemma: {lemma}, POS: {pos}")
```

```
print("\nGenerated Words:")
for lemma, forms in generated_words.items():
    print(f"Lemma: {lemma}, Forms: {forms}")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
Morphological Analysis:
Token: The, Lemma: The, POS: DT
Token: cats, Lemma: cat, POS: NNS
Token: are, Lemma: be, POS: VBP
Token: running, Lemma: run, POS: VBG
Token: quickly, Lemma: quickly, POS: RB
Token: ., Lemma: ., POS: .

Generated Words:
Lemma: The, Forms: ['The', 'Thes', "The's", "Thes'"]
Lemma: cat, Forms: ['cat', 'cats', "cat's", "cats'"]
Lemma: be, Forms: ['be', 'bes', 'beed', 'being']
Lemma: run, Forms: ['run', 'runs', 'runed', 'runing']
Lemma: quickly, Forms: ['quickly', 'quicklyly']
Lemma: ., Forms: ['.', '.s', ".'s", ".s'"]
```

Conclusion:

In conclusion, word generation in NLP is a crucial process for creating or transforming words based on grammatical or contextual needs. It can be achieved through inflectional generation, which modifies a word to fit different grammatical contexts like tense or number, or derivative generation, which forms new words by adding affixes to alter meaning or word category. Techniques like morphological analysis, rule-based methods, finite state transducers (FSTs), and advanced neural networks like RNNs, LSTMs, and transformers enable effective word generation for various NLP tasks, including text generation, machine translation, and language modeling.