Experiment No.4
Apply Stemming on the given Text input
Date of Performance:
Date of Submission:

Aim: Apply Stemming on the given Text input.

Objective: Understand the working of stemming algorithms and apply stemming on the given input text.

Theory:

Stemming is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "conect".

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words "chocolates", "chocolatey", "choco" to the root word, "chocolate" and "retrieval", "retrieved", "retrieves" and reduces to the stem "retrieve". Stemming is an important part of the pipelining process in Natural language processing. The input to the stemmer is tokenized words.

Applications of stemming :

1. Stemming is used in information retrieval systems like search engines.

2. It is used to determine domain vocabularies in domain analysis.

Porter's Stemmer Algorithm:

It is one of the most popular stemming methods proposed in 1980. It is based on the idea that the suffixes in the English language are made up of a combination of smaller and simpler suffixes. This stemmer is known for its speed and simplicity. The main applications of Porter Stemmer include data mining and Information retrieval. However, its applications are only limited to English words. Also, the group of stems is mapped on to the same stem and the output stem is not necessarily a meaningful word. The algorithms are fairly lengthy in nature and are known to be the oldest stemmer.

Example: EED -> EE means "if the word has at least one vowel and consonant plus EED ending, change the ending to EE" as 'agreed' becomes 'agree'.

Advantage: It produces the best output as compared to other stemmers and it has less error rate.

Limitation: Morphological variants produced are not always real words.

```
!pip install nltk
```

```
     Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
     Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
     Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
     Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
     Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

```
text = 'TON 618 is a hyperluminous, broad-absorption-line, radio-loud quasar and Lyman-alpha blob located near the border of the constellatio
```

```
text
```

```
     'TON 618 is a hyperluminous, broad-absorption-line, radio-loud quasar and Lyman-alpha b
     lob located near the border of the constellations Canes Venatici and Coma Berenices, wi
     th the projected comoving distance of approximately 18.2 billion light-years from Eart
```

```
from nltk.corpus import stopwords
```

```
import nltk
```

```
nltk.download('stopwords')
```

```
     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Unzipping corpora/stopwords.zip.
     True
```

```
stop_words = stopwords.words('english')
```

```
nltk.download('punkt')
```

```
     [nltk_data] Downloading package punkt to /root/nltk_data...
     [nltk_data]   Unzipping tokenizers/punkt.zip.
     True
```

```
from nltk.tokenize import word_tokenize
words = word_tokenize(text)
```

```
holder = list()
for w in words:
    if w not in set(stop_words):
        holder.append(w)
```

```
holder
```

```
     ['TON',
      '618',
      'hyperluminous',
      ',',
      'broad-absorption-line',
      ',',
      'radio-loud',
      'quasar',
      'Lyman-alpha',
      'blob',
      'located',
      'near',
      'border',
      'constellations',
      'Canes',
      'Venatici',
      'Coma',
      'Berenices',
      ',',
      'projected',
      'comoving',
      'distance',
      'approximately',
      '18.2',
      'billion',
      'light-years',
      'Earth',
      '.']
```

```
holder = [w for w in words if w not in set(stop_words)]
print(holder)
```

```
['TON', '618', 'hyperluminous', ',', 'broad-absorption-line', ',', 'radio-loud', 'quasar', 'Lyman-alpha', 'blob', 'located', 'near', 'bc
```

```
from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer
```

```
porter = PorterStemmer()
snow = SnowballStemmer(language = 'english')
lancaster = LancasterStemmer()
```

```
words = ['play', 'plays', 'played', 'playing', 'player']
```

```
porter_stemmed = list()
for w in words:
    stemmed_words = porter.stem(w)
    porter_stemmed.append(stemmed_words)
porter_stemmed
```

```
['play', 'play', 'play', 'play', 'player']
```

```
porter_stemmed = [porter.stem(x) for x in words]
print (porter_stemmed)
```

```
['play', 'play', 'play', 'play', 'player']
```

```
snow_stemmed = list()
for w in words:
    stemmed_words = snow.stem(w)
    snow_stemmed.append(stemmed_words)
```

```
snow_stemmed
```

```
['play', 'play', 'play', 'play', 'player']
```

```
snow_stemmed = [snow.stem(x) for x in words]
print (snow_stemmed)
```

```
['play', 'play', 'play', 'play', 'player']
```

```
lancaster_stemmed = list()
for w in words:
    stemmed_words = lancaster.stem(w)
    lancaster_stemmed.append(stemmed_words)
```

```
lancaster_stemmed
```

```
['play', 'play', 'play', 'play', 'play']
```

```
lancaster_stemmed = [lancaster.stem(x) for x in words]
print (lancaster_stemmed)
```

```
['play', 'play', 'play', 'play', 'play']
```

```
from nltk.stem import WordNetLemmatizer
wordnet = WordNetLemmatizer()
```

```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
lemmatized = [wordnet.lemmatize(x) for x in words]
```

```
lemmatized

    ['play', 'play', 'played', 'playing', 'player']
```

Double-click (or enter) to edit

Conclusion:

Comment on the implementation of stemming for an Indian language. Comment on the implementation of stemming for English (Explain which rules have been applied for identifying the stem words in your output).

Implementing stemming for an Indian language involves using language-specific rules or algorithms to reduce inflected or derived words to their root or base form. Stemming in Indian languages faces unique challenges due to the rich morphological complexity of these languages, with various tenses, gender, and conjugations. This requires the development of custom stemming algorithms or leveraging existing libraries that are tailored to specific languages. Additionally, the quality and accuracy of stemming can vary significantly between different Indian languages, emphasizing the need for careful language-specific implementation and evaluation.