Name: Niyati Patil
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 "connect".

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.

Output:

Stemming

```
In [ ]: from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer

In [ ]: porter = PorterStemmer()
        snow = SnowballStemmer(language = 'english')
        lancaster = LancasterStemmer()

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

Porter Stemmer

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

In [ ]: porter_stemmed

Out[ ]: ['play', 'play', 'play', 'play', 'player']
```

Porter Stemmer List Comprehension

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

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

Snowball Stemmer

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

In [ ]: snow_stemmed

Out[ ]: ['play', 'play', 'play', 'play', 'player']
```

Snowball Stemmer List Comprehension

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

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

Lancaster Stemmer

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

In [ ]: lancaster_stemmed

Out[ ]: ['play', 'play', 'play', 'play', 'play']
```

Lancaster Stemmer List Comprehension

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

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

Lemmatization : This has a more expansive vocabulary than Stemming

```
In [ ]: from nltk.stem import WordNetLemmatizer
        nltk.download('wordnet')
        wordnet = WordNetLemmatizer()

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

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

In [ ]: lemmatized

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

Conclusion:

Implementation of stemming for an Indian language: To implement stemming for an Indian language, you can follow these steps:

Choose a stemming algorithm. There are a number of stemming algorithms available, such as the Lovins stemmer, the Porter stemmer, and the Krovetz stemmer. You can choose an algorithm that is specifically designed for Indian languages, or you can use a general-purpose stemming algorithm.

Build a language-specific stemmer. If you are using a general-purpose stemming algorithm, you will need to build a language-specific stemmer. This involves creating a list of affixes that are specific to the Indian language that you are interested in. You can use a dictionary or a corpus of text to identify the affixes.

Implement the stemming algorithm. Once you have chosen a stemming algorithm and built a language-specific stemmer, you can implement the stemming algorithm in your code.

Implementation of stemming for English:

To implement stemming for English, you can use the Porter stemmer or the Krovetz stemmer. These stemmers are both widely available and easy to use.