Name : Bodhisatya Ghosh

Class : CSE DS

UID : 2021700026

Subject : NLP

Experiment number : 2

Aim:

1. Generate word forms from root and suffix information using Add-Delete table.
2. Comparative study of Porter/Snowball/Lancaster Stemmer and Stemmer vs Lemmatizer

Importing and cleaning test data

In [ ]:
```
import pandas as pd
import nltk
```

In [ ]:
```
data = pd.read_csv('./reviews.csv')
data.head()
```

Out[ ]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. <br /><br />The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

In [ ]:
```
data['review'] = data['review'].str.replace(r'[^A-Za-z0-9]',' ',regex=True)
data['review'] = data['review'].str.replace(r'\s+',' ',regex=True)
data.head()
```

Out[ ]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production br br The filmin...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there s a family where a little boy ...	negative
4	Petter Mattei s Love in the Time of Money is a...	positive

In [ ]:
```
reviews = data.loc[1,'review']
```

Creating add-delete table of all words in sample

In [ ]:
```
from word_forms.word_forms import get_word_forms
```

In [ ]:
```
data = reviews.split(" ")
addDel_data = []

for word in data:
    forms = []

    mapping = {}

    wordsData = get_word_forms(word)
    for category in wordsData:
        for form in wordsData[category]:
            forms.append(form)
    mapping["Original"] = word
    mapping['Forms'] = forms

    addDel_data.append(mapping)
    # print(mapping)
wordFormTable = pd.DataFrame(addDel_data)
wordFormTable
```

Out[ ]:

	Original	Forms
0	A	[a, as]
1	wonderful	[wonderfulnesses, wonderfulness, wonderful, wo...
2	little	[littlenesses, littleness, littles, little, li...
3	production	[producers, products, produce, producer, produ...
4	br	[]
...	...	...
162	are	[ares, beings, are, being, wasn't, am, been, a...
163	terribly	[terriblenesses, terribleness, terrible, terri...
164	well	[wellness, wells, wellnesses, well, well, well...
165	done	[doers, do, does, doer, done, do, does, didn't...
166		[]

167 rows × 2 columns

Comparing stemmers

Stemming function

In [ ]:
```
def stemFunction(data,stemmer):
    original = []
    root = []

    wordList = data.split(" ")

    for word in wordList:
        original.append(word)
        root.append(stemmer.stem(word))
    return root, original
```

Stemming using Snowball stemmer

In [ ]:
```
original = []
```

In [ ]:
```
#Using snowball stemmer
from nltk.stem.snowball import SnowballStemmer
snowballStemmer = SnowballStemmer('english')
```

In [ ]:
```
snowball, original = stemFunction(reviews,snowballStemmer)
```

Stemming using PorterStemmer

In [ ]:
```
from nltk.stem import PorterStemmer
portStemmer = PorterStemmer()
```

Stemming using LancasterStemmer

In [ ]:
```
porter, _ = stemFunction(reviews,portStemmer)
```

In [ ]:
```
from nltk.stem import LancasterStemmer
lancasterStemmer = LancasterStemmer()
```

In [ ]:
```
lancaster, _ = stemFunction(reviews, lancasterStemmer)
```

In [ ]:
```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

```
In [ ]: lemmatized = []
        data = reviews.split(" ")

        for word in data:
            lemmatized.append(lemmatizer.lemmatize(word))
```

Create comparison table

```
In [ ]: compare = pd.DataFrame({"Original":original,
                                "Snowball Stemmer":snowball,
                                "Porter Stemmer":porter,
                                "Lancaster Stemmer":lancaster,
                                "Lemmatized":lemmatized})
```

```
In [ ]: compare
```

Out[ ]:

	Original	Snowball Stemmer	Porter Stemmer	Lancaster Stemmer	Lemmatized
0	A	a	a	a	A
1	wonderful	wonder	wonder	wond	wonderful
2	little	littl	littl	littl	little
3	production	product	product	produc	production
4	br	br	br	br	br
...	...	...	...	...	...
162	are	are	are	ar	are
163	terribly	terribl	terribl	terr	terribly
164	well	well	well	wel	well
165	done	done	done	don	done
166					

167 rows × 5 columns

What is paradigm class? Give example

In linguistics, a paradigm refers to a set of related words that share a common grammatical feature, such as tense, person, or number.

For example, in English, the verb "to be" has a paradigm for the present tense with variations like "am," "is," and "are" depending on the person (I am, he/she/it is, we/you/they are). This demonstrates a paradigm class for the verb "to be" in the present tense.

What are the different types of morphemes. Give example of each.

Morphemes are the smallest units of meaning in a language. There are two main types of morphemes: free morphemes and bound morphemes. Additionally, bound morphemes can be further classified into two types: roots and affixes.

1. **Free Morphemes:**
    - **Example:** In the word "book," the morpheme "book" is a free morpheme because it can stand alone as a meaningful word.
2. **Bound Morphemes:**
    - **Roots:** These are the core, meaningful units to which affixes can be added.
        - **Example:** In the word "happiness," "happy" is the root morpheme.
    - **Affixes:** These are morphemes added to the root to create a new meaning.
        - **Prefix:** Added at the beginning of a root.
            - **Example:** In the word "unhappy," "un-" is a prefix.
        - **Suffix:** Added at the end of a root.
            - **Example:** In the word "happily," "-ly" is a suffix.
        - **Infix:** Added within a root.
            - **Example:** In some languages, an infix might be inserted for emphasis or grammatical purposes, but it's less common in English.