NLP PROGRAM-6

A program for stemming Non-English words

Soundarya G_ 2048057

- Use a minimum of 5 Stemmers(Specific to the language) and maximum N
- Compare each output with any one of the stemmers used before(English:Porter stemmer)
- Interpret the inflection for each output.

Downloading and Importing

```
# Import nltk
import nltk
nltk.download('all')
```

```
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Package abc is already up-to-date!
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Package alpino is already up-to-date!
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package biocreative_ppi is already up-to-date!
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Package brown is already up-to-date!
[nltk_data]    | Downloading package brown_tei to /root/nltk_data...
[nltk_data]    |   Package brown_tei is already up-to-date!
[nltk_data]    | Downloading package cess_cat to /root/nltk_data...
[nltk_data]    |   Package cess_cat is already up-to-date!
[nltk_data]    | Downloading package cess_esp to /root/nltk_data...
[nltk_data]    |   Package cess_esp is already up-to-date!
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Package chat80 is already up-to-date!
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package city_database is already up-to-date!
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package comparative_sentences is already up-to-
[nltk_data]    |       date!
[nltk_data]    | Downloading package comtrans to /root/nltk_data...
[nltk_data]    |   Package comtrans is already up-to-date!
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Package conll2000 is already up-to-date!
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Package conll2002 is already up-to-date!
[nltk_data]    | Downloading package conll2007 to /root/nltk_data...
[nltk_data]    |   Package conll2007 is already up-to-date!
[nltk_data]    | Downloading package crubadan to /root/nltk_data...
[nltk_data]    |   Package crubadan is already up-to-date!
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package dependency_treebank is already up-to-date!
[nltk_data]    | Downloading package dolch to /root/nltk_data...
[nltk_data]    |   Package dolch is already up-to-date!
[nltk_data]    | Downloading package europarl_raw to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package europarl_raw is already up-to-date!
```

```
[nltk_data]    | Downloading package floresta to /root/nltk_data...
[nltk_data]    |   Package floresta is already up-to-date!
[nltk_data]    | Downloading package framenet_v15 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package framenet_v15 is already up-to-date!
[nltk_data]    | Downloading package framenet_v17 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package framenet_v17 is already up-to-date!
[nltk_data]    | Downloading package gazetteers to /root/nltk_data...
[nltk_data]    |   Package gazetteers is already up-to-date!
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
```

```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3.2.5)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from nltk)
```

```
!pip install Tashaphyne
```

```
Requirement already satisfied: Tashaphyne in /usr/local/lib/python3.7/dist-packages (0.3
Requirement already satisfied: pyarabic in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python3.7/dist-packages (fr
```

Stemming

- Stemming is also a type of text normalization that enables you to standardize some words into specific expressions also called stems.
- Stemming is basically removing the suffix from a word and reduce it to its root word.

> For example: “Flying” is a word and its suffix is “ing”, if we remove “ing” from “Flying” then we will get base word or root word which is “Fly”

- Stemming VS Lemmatization

> Stemming: Produces a words "stem" which is produced by stemmers.

```
Ex: am -am
    the goin -the go
    having -hav
```

> Lemmatization: Produces words called "lemma" which is produced by lemmatizers.

```
Ex: am -be
    the going -the going
    having -have
```

- Application of Stemming and Lemmatization

```
- Sentimental Analysis
- Document Clustering
```

```
    - Information Retrieval: search engines.
    - To determine domain vocabularies in domain analysis.
```

Stemmers for Non-English words

There are different types of stemmers that supports english and non-english words

```
- Porter Stemmer
- Lancaster Stemmer
- Lovins Stemmer
- Dawson Stemmer
- Krovetz Stemmer
- Snowball Stemmer
- ISRI Stemmer
- RSLPS Stemmer
- Regexp Stemmer
- Germen Stemmer
- Cistem Stemmer
- Lancaster Stemmer
```

Python nltk provides not only two English stemmers: PorterStemmer and LancasterStemmer but also a lot of non-English stemmers as part of SnowballStemmers, ISRIStemmer, RSLPSStemmer. Python NLTK included SnowballStemmers as a language to create to create non-English stemmers. One can program one's own language stemmer using snowball. Currently, it supports the following languages:

```
# See which languages are supported
print(" ".join(SnowballStemmer.languages))
```

```
arabic danish dutch english finnish french german hungarian italian norwegian porter por
```

German

```
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.snowball import GermanStemmer
# from nltk.stem.cistem import Cistem
from nltk.stem import LancasterStemmer
from nltk.stem import PorterStemmer
```

```
# Defining the function that will take the String/word as the input and gives us the german s
def German_Stemmer(word):
    print ("The string is:",word)
    # Snow Ball Stemmer
    stemmer_1 = SnowballStemmer(language="german")
    print('Snowball stemmer:',stemmer_1.stem(word))

    # German Stemmer
    stemmer_2 = GermanStemmer()
    print('German stemmer:',stemmer_2.stem(word))

    # Cistem Stemmer
    #stemmer_3 = Cistem()
    #print('Cistem stemmer\t:',stemmer_3.segment(word))
    # Lancaster Stemmer
    stemmer_4 =LancasterStemmer()
    print("Lancaster stemmer:",stemmer_4.stem(word))

    # Porter Stemmer
    stemmer_5 = PorterStemmer()
    print("Porter stemmer:",stemmer_5.stem(word))
```

```
# English : Pretend
German_Stemmer("Vorgeben")
```

```
The string is: Vorgeben
Snowball stemmer: vorgeb
German stemmer: vorgeb
Lancaster stemmer: vorgeb
Porter stemmer: vorgeben
```

```
# English : Amusement
German_Stemmer("Amüsement")
```

```
The string is: Amüsement
Snowball stemmer: amusement
German stemmer: amusement
Lancaster stemmer: amüs
Porter stemmer: amüsement
```

```
# English : Translate
German_Stemmer("Übersetzen")
```

```
The string is: Übersetzen
Snowball stemmer: ubersetz
German stemmer: ubersetz
Lancaster stemmer: übersetz
Porter stemmer: übersetzen
```

```
# English : Sunrise
German_Stemmer("Singen")
```

```
The string is: Singen
Snowball stemmer: sing
German stemmer: sing
Lancaster stemmer: sing
Porter stemmer: singen
```

> Inference:

- For Vorgeben: according to Snowball Stemmer, German stemmer,Porter Stemmer and Lancaster Stemmer,the stem we got is 'vorgeb' which means 'specified'.
- For Übersetzen: Snowball Stemmer,German stemmer,and Porter Stemmer,the stem we got is 'ubersetz' which means 'translated'. Incase of Lancaster stemmer we get 'übersetz' which also means 'translated'.
- For Singen: Snowball Stemmer,German stemmer,and Lancaster Stemmer,the stem we got is 'sing' which means 'sing'. Incase of Porter stemmer we get 'singen' which also means 'singing'.

Spanish

```
# Defining the function that will take the String/word as the input and gives us the spanish
def Spanish_Stemmer(word):
    print ("The string is:",word)
    # Snow Ball Stemmer
    stemmer_1 = SnowballStemmer(language='spanish')
    print('Snowball stemmer:',stemmer_1.stem(word))

    # Porter Stemmer
    stemmer_2 = PorterStemmer()
    print("Porter stemmer:",stemmer_2.stem(word))

    # Lancaster Stemmer
    stemmer_3 =LancasterStemmer()
    print("Lancaster stemmer:",stemmer_3.stem(word))
```

```
# English: dancing
Spanish_Stemmer('baile')
```

```
The string is: baile
Snowball stemmer: bail
Porter stemmer: bail
Lancaster stemmer: bail
```

```
# English: translator
Spanish_Stemmer('traductora')
```

```
The string is: traductora
Snowball stemmer: traductor
Porter stemmer: traductora
Lancaster stemmer: traductor
```

> Inference:

- For baile: according to Snowball Stemmer, Porter Stemmer and Lancaster Stemmer,the stem we got is 'bail' which means 'dance'.
- For traductora: Snowball Stemmer and Lancaster Stemmer,the stem we got is 'traductor' which means 'translator'. Incase of Porter stemmer we get 'traductora' which also means 'translator'.

Portuguese

```
from nltk.stem import RSLPStemmer
```

```
# Defining the function that will take the String/word as the input and gives us the Portugue
def Portuguese_Stemmer(word):
    print ("The string is:",word)
    # Snow Ball Stemmer
    stemmer_1 = SnowballStemmer(language="portuguese")
    print('Snowball stemmer:',stemmer_1.stem(word))

    # RSLP Stemmer
    stemmer_2 = RSLPStemmer()
    print("RSLP stemmer:",stemmer_2.stem(word))
```

```
# English: playing
Portuguese_Stemmer('jogando')
```

```
The string is: jogando
Snowball stemmer: jog
RSLP stemmer: jog
```

> Inference:

- According to Snowball Stemmer the stem we got is 'jog' which means 'play'.
- According to RSLP Stemmer the stem we got is 'jog' which means 'play'.

```
# English: translator
Portuguese_Stemmer('tradutora')
```

```
The string is: tradutora
Snowball stemmer: tradutor
RSLP stemmer: tradu
```

> Inference:

- According to Snowball Stemmer and ArabicLight Stemmer the stem we got is 'tradutor', which means 'a translator'.
- According to ISRIS Stemmer the stem we got is 'tradu' which means 'translate'.

Arabic

```
from tashaphyne.stemming import ArabicLightStemmer
from nltk.stem.isri import ISRIStemmer
```

```
# Defining the function that will take the String/word as the input and gives us the Arabic S
def Arabic_Stemmer(word):
    print ("The string is:",word)
    # Snow Ball Stemmer
    stemmer_1 = SnowballStemmer(language="arabic")
    print('Snowball stemmer:',stemmer_1.stem(word))

    # ArabicLight Stemmer
    stemmer_2 = ArabicLightStemmer()
    print("ArabicLight stemmer:",stemmer_2.light_stem(word))

    # ISRIStemmer
    stemmer_3 = ISRIStemmer()
    print("ISRIStemmer:",stemmer_3.stem(word))
```

```
# English: translator
Arabic_Stemmer('مترجم')
```

```
The string is: مترجم
Snowball stemmer: مترجم
ArabicLight stemmer: مترجم
ISRIStemmer: ترجم
```

> Inference:

- According to Snowball Stemmer and ArabicLight Stemmer the stem we got is 'مترجم', it dint get changed.
- According to ISRIS Stemmer the stem we got is 'ترجم' which means 'translate'.

```
# English: fairly
Arabic_Stemmer('تماما')
```

```
The string is: تماما
Snowball stemmer: تمام
```

```
ArabicLight stemmer: مام
ISRIStemmer: تما
```

Inference:

- According to Snowball Stemmer the stem we got is 'تمام' which means 'OK'.
- According to ArabicLight Stemmer the stem we got is 'مام' which means 'mum' while translating it to english.
- According to ISRIStemmer the stem we obtained is 'تما' which means 'Perfect'.