Atharva Prashant Pawar [9427] - [Batch-D]

NLP: Exp ~~3~~ 2 : PostLab

Q1. What all python libraries are available to work with Indian language like Hindi, Punjabi, Marathi, etc?

→1. Indic NLP Library: provides tools for tokenization, stemming & other NLP tasks for various Indian languag.

2. Snow NLP: Supports sentiment analysis, keyword extracti & languag identification for multiple lang. including Indian ones.

3. Polyglot: Offers lang. detection, entity recognition & sentiment analysis for a wide range of languages, including Indian ones.

4. NLTK (Natural lang. Toolkit):
Provides basic NLP functionalities for Indian lang. through various lang. specific corpora & resources.

5. Spacy:
An NLP library with support for tokenization & named entity recognition for Indian lang.

6. Open NLP:
Can be used for tokenization, part of speech tagging, & more for lang. like Hindi.

7. Transliterate:
Allows transliteration b/w scripts, such as from English to Indian lang. scripts.

8. Gensim:
Offers word embeddings & topic modeling for Indian lang.

9. Bharati:
Specifically designed for Indian lang., it covers tasks like stemming, tokenization & more.

10. Indian NLP Tool kit:
A collection of NLP tools for Indian language, including POS tagging & morphological analysis.

Example Code:

```
from indicnlp.tokenizer import indic_tokenize

hindi_sentence = "मैं एक उदाहरण हूँ !"

tokens = indic_tokenize.trival_tokenize (hindisentence)

print(tokens)
```

NLP - EXP - 2

Atharva Prashant Pawar (9427) - [Batch - D]

Q1. Write a python code to remove punctuations, URLs and stop words.

```
import string
import re
import nltk
from nltk.corpus import stopwords
# nltk.download('stopwords')

def remove_punctuations(text): # Remove punctuations
    return text.translate(str.maketrans("", "", string.punctuation))

def remove_urls(text): # Remove URLs
    return re.sub(r"http\S+|www\S+|https\S+", "", text, flags=re.MULTILINE)

def remove_stop_words(text): # Remove stop words
    stop_words = set(stopwords.words("english"))
    words = text.split()
    filtered_words = []
    for word in words:
        if word.lower() not in stop_words:
            filtered_words.append(word)
    return (" ".join(filtered_words), stop_words)

def preprocess_text(text):
    print("## Remove_punctuations from text : ", remove_punctuations(text))
    print("## Remove_urls from text : ", remove_urls(text))
    text,stop_words = remove_stop_words(text)
    # print("All stop_words : ", stop_words)
    print("## Remove_stop_words from text : ", text)

# == Main Run ========================================================
input_text = "Hello, world! This is a sample text with a URL: https://www.myname.com"
print("## Raw Text : ", input_text)
preprocess_text(input_text)
```

```
## Raw Text :  Hello, world! This is a sample text with a URL: https://www.myname.com
## Remove_punctuations from text :  Hello world This is a sample text with a URL httpswwwmynamecom
## Remove_urls from text :  Hello, world! This is a sample text with a URL:
## Remove_stop_words from text :  Hello, world! sample text URL: https://www.myname.com
```

Q 2 Write a python code perform stemmer operation using Porterstemmer ,Snowballstemmer, Lancasterstemmer, RegExpStemmer

```
import nltk
from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer, RegexpStemmer
from nltk.corpus import stopwords
# nltk.download('punkt')
# nltk.download('stopwords')

def stem_with_porter(text): # Simple, Widely used, efficient.
    ps = PorterStemmer()
    words = nltk.word_tokenize(text)
    stemmed_words = [ps.stem(word) for word in words]
    return stemmed_words

def stem_with_snowball(text): # Multilingual, aggressive.
    ss = SnowballStemmer("english")
    words = nltk.word_tokenize(text)
    stemmed_words = [ss.stem(word) for word in words]
    return stemmed_words

def stem_with_lancaster(text): # Fast, aggressive.
    ls = LancasterStemmer()
    words = nltk.word_tokenize(text)
    stemmed_words = [ls.stem(word) for word in words]
    return stemmed_words

def stem_with_regexp(text, regexp):
    rs = RegexpStemmer(regexp)
    words = nltk.word_tokenize(text)
    stemmed_words = [rs.stem(word) for word in words]
    return stemmed_words
```

```
def preprocess_text(input_text):
    print("Original Text:", input_text)
    words = nltk.word_tokenize(input_text)
    porter_stemmed    , snowball_stemmed  = stem_with_porter(input_text)    , stem_with_snowball(input_text)
    lancaster_stemmed , regexp_stemmed    = stem_with_lancaster(input_text) , stem_with_regexp(input_text, r'ing$|ed$')
    return pd.DataFrame({'Original Word': words, 'Porter': porter_stemmed, 'Snowball': snowball_stemmed, 'Lancaster': lancaster_stemmed,

# == Main Run =========================================================
input_text = "Coders coding coded code"
print(preprocess_text(input_text))
```

```
Original Text: Coders coding coded code
  Original Word Porter Snowball Lancaster  RegExp
0        Coders  coder    coder       cod  Coders
1        coding   code     code       cod     cod
2         coded   code     code       cod     cod
3          code   code     code       cod    code
```

Q 3 Write a python code to demonstrate the comparative study of all 4 stemmers for a given text corpus.

```
# Extra Testing : only for Snowball with multi language example

import nltk
from nltk.stem import SnowballStemmer
# nltk.download('punkt')

def stem_with_snowball(input_text, _language):
    ss = SnowballStemmer(_language)
    words = nltk.word_tokenize(input_text)
    stemmed_words = [ss.stem(word) for word in words]
    return " ".join(stemmed_words)

def preprocess_text(input_text):

    # Stemming using SnowballStemmer
    print("\n\n## Snowball Stemmed Text (English): \n", stem_with_snowball(input_text, "english"))
    supported_languages = nltk.stem.snowball.SnowballStemmer.languages
    print("\nSupported Languages for SnowballStemmers:")
    print(supported_languages)

    # "The children play in the garden while running."
    french_input_text = "Les enfants jouent dans le jardin en courant."
    print("\n\n\t\t## Snowball Stemmed Text (french): \n\t\t(inp)",french_input_text,"\n\t\t(out)", stem_with_snowball(french_input_text,
    dutch_input_text = "De kinderen spelen in de tuin en rennen rond."
    print("\n\n\t\t## Snowball Stemmed Text (dutch): \n\t\t(inp)",dutch_input_text,"\n\t\t(out)", stem_with_snowball(dutch_input_text, "c

# == Main Run =========================================================
input_text = "Coders coding coded code jumping jumped jumps runs running run apples oranges playing played plays"
words = nltk.word_tokenize(input_text)
print("## Original Text: \n", input_text)
preprocess_text(input_text)
```

```
## Original Text:
 Coders coding coded code jumping jumped jumps runs running run apples oranges playing played plays


## Porter Stemmed Text:
 coder code code code jump jump jump run run run appl orang play play play


## Snowball Stemmed Text (English):
 coder code code code jump jump jump run run run appl orang play play play

Supported Languages for SnowballStemmers:
('arabic', 'danish', 'dutch', 'english', 'finnish', 'french', 'german', 'hungarian', 'italian', 'norwegian', 'porter', 'portuguese'


                ## Snowball Stemmed Text (french):
                (inp) Les enfants jouent dans le jardin en courant.
                (out) le enfant jouent dan le jardin en cour .


                ## Snowball Stemmed Text (dutch):
                (inp) De kinderen spelen in de tuin en rennen rond.
                (out) de kinder spel in de tuin en renn rond .


## Lancaster Stemmed Text:
 cod cod cod cod jump jump jump run run run appl orang play play play


## RegExp Stemmed Text:
 Coder cod cod code jump jump jump run runn run apple orange play play play
```

```
import nltk
from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer, RegexpStemmer
import pandas as pd
# nltk.download('punkt')

def stem_with_porter(words):
    ps = PorterStemmer()
    return [ps.stem(word) for word in words]

def stem_with_snowball(input_text, _language):
    ss = SnowballStemmer(_language)
    words = nltk.word_tokenize(input_text)
    return [ss.stem(word) for word in words]

def stem_with_lancaster(words):
    ls = LancasterStemmer()
    return [ls.stem(word) for word in words]

def stem_with_regexp(words, regexp):
    rs = RegexpStemmer(regexp)
    return [rs.stem(word) for word in words]


def preprocess_text(input_text):
    words = nltk.word_tokenize(input_text)

    porter_stemmed = stem_with_porter(words)
    snowball_stemmed = stem_with_snowball(input_text, "english")
    lancaster_stemmed = stem_with_lancaster(words)
    regexp_stemmed = stem_with_regexp(words, r'(ing|ed|s)$')

    df = pd.DataFrame({'Original Word': words, 'Porter': porter_stemmed, 'Snowball': snowball_stemmed, 'Lancaster': lancaster_stemmed, 'R
    print(df)


# == Main Run ==========================================================
input_text = "Coders jumping apples oranges playing "

print("## Original Text: ", input_text, "\n")

preprocess_text(input_text)
```

```
## Original Text:  Coders jumping apples oranges playing

  Original Word Porter Snowball Lancaster  RegExp
0        Coders  coder    coder       cod   Coder
1       jumping   jump     jump      jump    jump
2        apples   appl     appl      appl   apple
3       oranges  orang    orang     orang  orange
4       playing   play     play      play    play
```

Q 4 Write a python code perform lemmatization using NLTK library.

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
import pandas as pd

# nltk.download('punkt')
# nltk.download('wordnet')

def lemmatize_text(text):
    tokens = word_tokenize(text)
    lemmatizer = WordNetLemmatizer()
    lemmas = []
    for token in tokens:
      lemmas.append(lemmatizer.lemmatize(token))

    df = pd.DataFrame({'Raw Word': tokens, '(spaCy)': lemmas})

    return df

# == Main Run ==========================================================
text = "Coders jumping apples oranges playing "
print(lemmatize_text(text))
```

```
  Raw Word  (spaCy)
0   Coders   Coders
1  jumping  jumping
2   apples    apple
3  oranges   orange
4  playing  playing
```

Q 5 Write a python code perform lemmatization using Spacy library.

```
import spacy
import pandas as pd

def lemmatize_text(text):
    nlp = spacy.load('en_core_web_sm')
    doc = nlp(text)
    lemmas = []
    for token in doc:
      lemmas.append(token.lemma_)
    df = pd.DataFrame({'Raw Word': doc, '(spaCy)': lemmas})
    return df

# == Main Run ==========================================================
text = "Coders jumping apples oranges playing "
print(lemmatize_text(text))
```

```
  Raw Word (spaCy)
0   Coders   coder
1  jumping    jump
2   apples   apple
3  oranges  orange
4  playing    play
```

Q 6 Compare the results lemmatization with Spacy and NLTK for the corpus given below- walking, is , main, animals , foxes, are, jumping , sleeping.

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
import spacy
import pandas as pd

# nltk.download('punkt')
# nltk.download('wordnet')

def lemmatize_text_nltk(text):
    tokens = word_tokenize(text)
    lemmatizer = WordNetLemmatizer()
    lemmas = []
    for token in tokens:
      lemmas.append(lemmatizer.lemmatize(token))
    return lemmas

def lemmatize_text_spacy(text):
    nlp = spacy.load('en_core_web_sm')
    doc = nlp(text)
    lemmas = []
    for token in doc:
      lemmas.append(token.lemma_)
    return lemmas

# == Main Run ==========================================================
text = "Coders jumping apples oranges playing "
tokens = word_tokenize(text)
Lemmatized_text_NLTK = lemmatize_text_nltk(text)
Lemmatized_text_spaCy = lemmatize_text_spacy(text)

df = pd.DataFrame({'Raw Word': tokens, '(NLTK)': Lemmatized_text_NLTK, '(spaCy)': Lemmatized_text_spaCy})
print(df)
```

```
  Raw Word   (NLTK) (spaCy)
0   Coders   Coders   coder
1  jumping  jumping    jump
2   apples    apple   apple
3  oranges   orange  orange
4  playing  playing    play
```