2. NLP – Text wrangling and cleaning

Contents

2. NLP – Text wrangling and cleaning

1. Text wrangling and cleansing

- ✓ The ultimate goal of doing text processing is to, machine can understand the text data.
- ✓ Text processing is very important as it helps us understand our data better and gain insights from it.
- ✓ Text processing normally involves the following steps,
  - Tokenization
  - Lemmatization
  - Stemming
  - Stop word removal

Program Name: Downloading punkt
demo1.py

```
import nltk
nltk.download("punkt")
```

Output

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\admin\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

- ✓ We have then downloaded the punkt, which is a Tokenizer for a text into a list of sentences.
- ✓ We can now do text wrangling.

2. Sentence Splitting

- ✓ Generally we used to understand sentence by reading words
- ✓ We will start reading sentence to understand paragraph.
- ✓ So, a paragraph we need to split it into a set of sentences.
- ✓ So, let's understand words, sentences and paragraph.
- ✓ This we can easily do by using nltk package

Program Name: Sentence splitting by using split() method
demo2.py

```
myString = "This is a paragraph. It should split at the end of sentence marker, such as a period. It can tell that the period in Mr.Daniel is not an end. Run it!, Hey How are you doing"

result = myString.split(".")
print(result)
```

Output

```
['This is a paragraph', ' It should split at the end of sentence marker, such as a period', ' It can tell that the period in Mr', 'Daniel is not an end', ' Run it!, Hey How are you doing']
```

Program Name: Sentence splitting
demo3.py

```
from nltk.tokenize import sent_tokenize

myString = "This is a paragraph. It should split at the end of sentence marker, such as a period. It can tell that the period in Mr.Daniel is not an end. Run it!, Hey How are you doing"

tokenized_sentence = sent_tokenize(myString)
print(tokenized_sentence)
```

Output

```
['This is a paragraph.', 'It should split at the end of sentence marker, such as a period.', 'It can tell that the period in Mr.Daniel is not an end.', 'Run it!, Hey How are you doing']
```

Note

- ✓ We can understand from above output, the paragraph was split into the exact sentences.
- ✓ It was also able to tell the difference between a period that has been used to end a sentence from one used on the name Mr.Daniel also

3. Tokenization

- ✓ Tokenization refers to the process by which a large text is broken down into various pieces.
- ✓ In NLP, a token is the minimal piece of text that a machine can be able to understand.
- ✓ A text may be tokenized into words or sentences.
    - In the case of sentence tokenization, every sentence will be identified as a token.
    - In the case of word tokenization, every word will be identified as a token.

4. Word tokenization

- ✓ There are various ways through which tokenization can be done, but the most popular way of doing it is through word tokenization.
- ✓ What happens in word tokenization is that a large text is broken down into words and the words are used as the tokens.
  - word_tokenizer
  - sent_tokenizer
  - punkt_tokenizer
  - Regexp_tokenizer

- ✓ To tokenize a text, we can still apply split() method from string

Program Name: Word tokenization
demo4.py

```
myString = "These are sentences. Let us tokenize it! Run it!"

print(myString.split())
```

Output

```
['These', 'are', 'sentences.', 'Let', 'us', 'tokenize', 'it!', 'Run', 'it!']
```

Program Name: Word tokenization with word_tokenize function
demo5.py

```
from nltk.tokenize import word_tokenize

myString = "These are sentences. Let us tokenize it! Run it!"

print(word_tokenize(myString))
```

Output

```
['These', 'are', 'sentences', '.', 'Let', 'us', 'tokenize', 'it', '!', 'Run', 'it', '!']
```

Program Name: Word tokenization with regexp_tokenize function
demo6.py

```
from nltk.tokenize import regexp_tokenize

myString = "These are sentences. Let us tokenize it! Run it!"

print(regexp_tokenize(myString, pattern="\w+"))
```

Output

```
['These', 'are', 'sentences', 'Let', 'us', 'tokenize', 'it', 'Run', 'it']
```

Program Name: Capture the digit from sentence
demo7.py

```
from nltk.tokenize import regexp_tokenize

myString = "These are 3 sentences. Let us tokenize them! Run the code!"

print(regexp_tokenize(myString, pattern="\d+"))
```

Output

```
['3']
```

5. Stemming

- ✓ This is another step in text wrangling.
- ✓ From the name, you can get the meaning, cutting down a token to its root stem.
- ✓ Stemming algorithm works by cutting the suffix from the word.
- ✓ In simple, it cuts either the beginning or end of the word.
- ✓ Consider the word "cutting".
- ✓ This word can be broken down to its root, which is "cut".

Program Name: Stemming
demo8.py

```
from nltk.stem import PorterStemmer

porter = PorterStemmer()
print(porter.stem("cutting"))
```

Output

```
cut
```

Program Name: Stemming
demo9.py

```
from nltk.stem import PorterStemmer

e_words= ["wait", "waiting", "waited", "waits"]

ps = PorterStemmer()

for w in e_words:
        rootWord = ps.stem(w)
        print(rootWord)
```

Output

```
wait
wait
wait
wait
```

Program Name: Stemming
demo10.py

```
import nltk
from nltk.stem.porter import PorterStemmer

porter_stemmer  = PorterStemmer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)

for w in tokenization:
        print("Stemming for {} is {}".format(w, porter_stemmer.stem(w)))
```

Output

```
Stemming for studies is studi
Stemming for studying is studi
Stemming for cries is cri
Stemming for cry is cri
```

- ✓ Stemming is good for its simplicity when it comes to dealing with NLP problems.
- ✓ However, when more complex stemming is needed, stemming is not the best option, but we have lemmatization.

6. Lemmatization

- ✓ Lemmatization is a more advanced compared to stemming.
- ✓ It follow specific rules to get results.
- ✓ It understand the context, parts of the speech to understand the root of the word

Program Name: Lemmatization
demo11.py

```
from nltk.stem import WordNetLemmatizer

wl = WordNetLemmatizer()

print("rocks :", wl.lemmatize("rocks"))
print("corpora :", wl.lemmatize("corpora"))
print("better :", wl.lemmatize("better", pos = "a"))
```

Output

```
rocks : rock
corpora : corpus
better : good
```

Program Name: Lemmatization
demo12.py

```
import nltk
from nltk.stem import WordNetLemmatizer

wl = WordNetLemmatizer()

text = "studies studying cries cry"

tokens = nltk.word_tokenize(text)

for word in tokens:
        print("Lemmatization for {} is {}".format(word, wl.lemmatize(word)))
```

Output

```
Lemmatization for studies is study
Lemmatization for studying is studying
Lemmatization for cries is cry
Lemmatization for cry is cry
```

7. Stop word removal

- ✓ Some common words that are present in text but do not contribute in the meaning of a sentence.
- ✓ Such words are not at all important for the purpose of information retrieval or natural language processing.
- ✓ The most common stopwords are ‘a’, ‘in’ ‘the’ etc
- ✓ The good news is that nltk comes with a list of stop words and in different languages.

Program Name: list of the stopwords
demo13.py

```
import nltk
from nltk.corpus import stopwords

print(stopwords.words('english'))
```

Output

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
"you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they',
'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been',
'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the',
'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with',
'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above',
'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again',
'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any',
'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not',
'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don',
"don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren',
"aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',
"hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
"shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]
```

Program Name: list of the languages
demo13.py

```
from nltk.corpus import stopwords

langs = stopwords.fileids()
print(len(langs))
```

Output

```
29
```

Program Name: Lemmatization
demo14.py

```
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')

mylist = stopwords.words('english')

line = "This is really good, how are you doing"

postPa = [word for word in line.split()]

print(postPa)
```

Output

```
['This', 'is', 'really', 'good,', 'how', 'are', 'you', 'doing']
```

Program Name: Lemmatization
demo15.py

```
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')

mylist = stopwords.words('english')

line = "This is really good, how are you doing"

postPa = [word        for word in line.split()        if word not in mylist]

print(postPa)
```

Output

```
['This', 'really', 'good,']
```