Text Preprocessing in Python

Text Processing pertains to the analysis of text data using a programming language such as Python. Text Processing is an essential task in NLP as it helps to clean and transform raw data into a suitable format used for analysis or modeling.

In this article, we will learn by using various Python Libraries and Techniques that are involved in Text Processing.

> **Prerequisites**: Introduction to NLP

Whenever we have textual data, we need to apply several processing and pre-processing steps to the data to transform words into numerical features that work with machine learning algorithms. The pre-processing steps for a problem depend mainly on the domain and the problem itself, hence, we don't need to apply all steps to every problem.
In this article, we are going to see text preprocessing in Python. We will be using the NLTK (Natural Language Toolkit) library here.

Python` ```

import the necessary libraries

import nltk import string import re

```
      `


  Text Lowercase
  --------------

  We lowercase the text to reduce the size of the vocabulary of our text data.


  Python`    ```
  def text_lowercase(text):
      return text.lower()

  input_str = "Hey, did you know that the summer break is coming? Amazing right !! It's only 5 m
  text_lowercase(input_str)
```

`

Example:

> **Input:** “Hey, did you know that the summer break is coming? Amazing right!! It’s only 5 more days!!”
> **Output:** “hey, did you know that the summer break is coming? amazing right!! it’s only 5 more days!!”

Remove numbers

We can either remove numbers or convert the numbers into their textual representations.
We can use regular expressions to remove the numbers.

Python` ```

Remove numbers

def remove_numbers(text): result = re.sub(r'\d+', '', text) return result

input_str = "There are 3 balls in this bag, and 12 in the other one." remove_numbers(input_str)

`

```
****Example:****

> ****Input:**** “There are 3 balls in this bag, and 12 in the other one.”
> ****Output:**** ‘There are balls in this bag, and in the other one.’
>

We can also convert the numbers into words. This can be done by using the inflect library.

Python`   ```
# import the inflect library
import inflect
p = inflect.engine()

# convert number into words
def convert_number(text):
```

```
        # split string into list of words
        temp_str = text.split()
        # initialise empty list
        new_string = []

        for word in temp_str:
            # if word is a digit, convert the digit
            # to numbers and append into the new_string list
            if word.isdigit():
                temp = p.number_to_words(word)
                new_string.append(temp)

            # append the word as it is
            else:
                new_string.append(word)

        # join the words of new_string to form a string
        temp_str = ' '.join(new_string)
        return temp_str

input_str = 'There are 3 balls in this bag, and 12 in the other one.'
convert_number(input_str)
```

`

Example:

> **Input:** “There are 3 balls in this bag, and 12 in the other one.”
> **Output:** “There are three balls in this bag, and twelve in the other one.”

Remove punctuation

We remove punctuations so that we don’t have different forms of the same word. If we don’t remove the punctuation, then been. been, been! will be treated separately.

Python` ```

remove punctuation

def remove_punctuation(text): translator = str.maketrans('', '', string.punctuation) return text.translate(translator) input_str = "Hey, did you know that the summer break is coming? Amazing right !! It's only 5 more days !!" remove_punctuation(input_str)

```
      `

****Example:****


> ****Input:**** “Hey, did you know that the summer break is coming? Amazing right!! It’s only
> ****Output:**** “Hey did you know that the summer break is coming Amazing right Its only 5 m
>

### Remove whitespace

We can use the join and split function to remove all the white spaces in a string.


Python`   ```
# remove whitespace from text
def remove_whitespace(text):
    return  " ".join(text.split())
input_str = "we don't need   the given questions"
remove_whitespace(input_str)
```

`

Example:

```
Input: "   we don't need   the given questions"
Output: "we don't need the given questions"
```

Remove default stopwords

Stopwords are words that do not contribute to the meaning of a sentence. Hence, they can safely be removed without causing any change in the meaning of the sentence. The NLTK library has a set of stopwords and we can use these to remove stopwords from our text and return a list of word tokens.

```
LIST OF ENGLISH STOPWORDS IN NLTK:

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

Example:

Python` ``` from nltk.corpus import stopwords from nltk.tokenize import word_tokenize

remove stopwords function

def remove_stopwords(text): stop_words = set(stopwords.words("english")) word_tokens = word_tokenize(text) filtered_text = [word for word in word_tokens if word not in stop_words] return filtered_text

example_text = "This is a sample sentence and we are going to remove the stopwords from this." remove_stopwords(example_text)

`

****Example:****

> ****Input:**** “This is a sample sentence and we are going to remove the stopwords from this
> ****Output:**** \[‘This’, ‘sample’, ‘sentence’, ‘going’, ‘remove’, ‘stopwords’\]
>

Stemming

Stemming is the process of getting the root form of a word. Stem or root is the part to which

```
### ****Example:****
```

books ---> book looked ---> look denied ---> deni flies ---> fli

```
![](https://media.geeksforgeeks.org/wp-content/uploads/20190524003811/Screenshot-1014.png)

![](https://media.geeksforgeeks.org/wp-content/uploads/20190524004043/Screenshot-1033.png)

If the text is not in tokens, then we need to convert it into tokens. After we have converted

Python`   ```
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer = PorterStemmer()

# stem words in the list of tokenized words
def stem_words(text):
    word_tokens = word_tokenize(text)
    stems = [stemmer.stem(word) for word in word_tokens]
    return stems

text = 'data science uses scientific methods algorithms and many types of processes'
stem_words(text)
```

`

Example:

> **Input:** 'data science uses scientific methods algorithms and many types of processes'
> **Output:** ['data', 'scienc', 'use', 'scientif', 'method', 'algorithm', 'and', 'mani', 'type', 'of', 'process']

Lemmatization

Like stemming, lemmatization also converts a word to its root form. The only difference is that lemmatization ensures that the root word belongs to the language. We will get valid words if we use

lemmatization. In NLTK, we use the WordNetLemmatizer to get the lemmas of words. We also need to provide a context for the lemmatization. So, we add the part-of-speech as a parameter.

Python` ``` from nltk.stem import WordNetLemmatizer from nltk.tokenize import word_tokenize lemmatizer = WordNetLemmatizer()

def lemma_words(text): word_tokens = word_tokenize(text) lemmas = [lemmatizer.lemmatize(word) for word in word_tokens] return lemmas

input_str = "data science uses scientific methods algorithms and many types of processes" lemma_words(input_str)

`

****Example:****

> ****Input:**** ‘data science uses scientific methods algorithms and many types of processes’
> ****Output:**** \[‘data’, ‘science’, ‘use’, ‘scientific’, ‘methods’, ‘algorithms’, ‘and’, ‘m

FAQs on Text-Processing in Python

Q1. What are the testing frameworks are commonly used for text processing in python?

> * ****unittest:**** There are some built-in testing frameworks in Python
> * ****pytest:**** This is the third-party testing frameworks that are known for its simpli
> * ****nose2:**** this is nose testing frameworks with additional features.
> * ****doctest:**** A module for testing docstrings by running examples embedded in documen

Q2. How do i run tests in Python?

> You can run tests using the following commands:
>
> * ****For unittest: ‘python -m unittest <test_module>’****
> * ****For Pytest: ‘pytest<test_module>****

Q3. How can i mock objects in Python tests?

> Python provides the ‘****unittest.mock’**** module, that allow to create mock objects for te

Q4. How do i write basic test in Python?

> You can create a test class which inherits from ‘****unittest.TestCase’**** and write test m