Experiment 1

Name: Raviraj Shinde

Branch: BE CMPN-A

Roll No: 18102A0010

Title: To perform Text pre-processing

Estimated time to complete this experiment: 2 hours

Objective: Understanding the various steps involved in Text preprocessing so that further analysis can be performed on the standardized text.

Expected Outcome of Experiment: To create a standardized text from the natural language text.

References:

1. https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/
2. https://www.geeksforgeeks.org/text-preprocessing-in-python-set-1/
3. https://www.kaggle.com/sudalairajkumar/getting-started-with-text-preprocessing
4. https://towardsdatascience.com/nlp-text-preprocessing-a-practical-guide-and-template-d80874676e79
5. https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html
6. https://analyticsindiamag.com/complete-tutorial-on-text-preprocessing-in-nlp/

Pre Lab/ Prior Concepts: Programming language – Python or R

Brief description:

There are 3 main components:

1. Tokenization

Tokenization is about splitting strings of text into smaller pieces, or “tokens”. Paragraphs can be tokenized into sentences and sentences can be tokenized into words.

2. Normalization

   Normalization aims to put all text on a level playing field, e.g., converting all characters to lowercase.

3. Noise removal

   Noise removal cleans up the text, e.g., remove extra whitespaces.

New Concepts to be learned: Text pre-processing Techniques

Requirements: PC with Python or R installed.

Flow Chart: -

Theory:

Steps in Text Pre-processing are:

- **Remove HTML tags**

  This is useful if we scrap the data from different websites. We might end up having html strings as part of our text. Since these tags are not useful for our NLP tasks, it is better to remove them.

- **Remove extra whitespaces**

  These are unwanted extra spaces between words. They have no significant meaning, hence it is better to remove them.

- **Convert accented characters**

Words with accent marks like “latté” and “café” can be converted and standardized to just “latte” and “cafe”, else our NLP model will treat “latté” and “latte” as different words even though they are referring to same thing.

- **Expand contractions**

  Contraction is the shortened form of a word like don’t stands for do not, aren’t stands for are not. Like this, we need to expand this contraction in the text data for better analysis. you can easily get the dictionary of contractions on google or create your own and use the re module to map the contractions.

- **Remove special characters**

  Special characters typically include any character that is not a letter or number, such as punctuation and whitespace. Removing special characters from a string results in a string containing only letters and numbers.

- **Lowercase all texts**

  If the text is in the same case, it is easy for a machine to interpret the words because the lower case and upper case are treated differently by the machine. So, we need to make the text in the same case and the most preferred case is a lower case to avoid such problems.

- **Convert number words to numeric form**

  This step involves the conversion of number words to numeric form, e.g., seven to 7, to standardize text.

- **Remove numbers**

  While analyzing text for certain applications, the numbers in the string are not of importance. In such cases we prefer to remove numbers and work on the words only.

- **Remove stopwords**

Stopwords are the most commonly occurring words in a text which do not provide any valuable information. stopwords like they, there, this, where, etc are some of the stopwords. NLTK library is a common library that is used to remove stopwords and include approximately 180 stopwords which it removes. If we want to add any new word to a set of words then it is easy using the add method.

- **Stemming**

  Stemming is a process to reduce the word to its root stem for example run, running, runs, runed derived from the same word as run. basically stemming do is remove the prefix or suffix from word like ing, s, es, etc. NLTK library is used to stem the words. The stemming technique is not used for production purposes because it is not so efficient technique and most of the time it stems the unwanted words. So, to solve the problem another technique came into the market as Lemmatization. there are various types of stemming algorithms like porter stemmer, snowball stemmer. Porter stemmer is widely used present in the NLTK library.

- **Lemmatization**

  Lemmatization is similar to stemming, used to stem the words into root word but differs in working. Actually, Lemmatization is a systematic way to reduce the words into their lemma by matching them with a language dictionary.

- **Sentence Tokenization**

  Splitting sentences in the paragraph

- **Word Tokenization**

  Splitting words in a sentence.

Program:

- **Remove HTML tags**

Scraping website:

```
import requests
URL = "https://vit.edu.in/"
page = requests.get(URL)
data = page.text
print(data)
```

Removing HTML tags:

```
from bs4 import BeautifulSoup
def remove_html(text):
    return BeautifulSoup(text, "lxml").text
print(remove_html(data))
```

- **Remove extra whitespaces**

```
def remove_whitespace(text):
    return  " ".join(text.split())

remove_whitespace(data)
```

- **Convert accented characters**

```
import unidecode
def remove_accented_chars(text):
    text = unidecode.unidecode(text)
    return text
```

- **Expand contractions**

```
def expand_contractions(text):
    text = contractions.fix(text)
    return text
```

- **Remove special characters**

```
for character in data:
    if character.isalnum():
        result += character

print(result)
```

- **Lowercase all texts**

```
print(data.lower())
```

- **Convert number from numeric form to 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 = num2words(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
```

- **Remove numbers**

```
res = ''.join([i for i in data if not i.isdigit()])
```

- **Remove stopwords**

```
import nltk
```

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
def remove_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in stop_words])
print(remove_stopwords(input_str))
```

- **Stemming**

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
def stem_words(text):
    return " ".join([stemmer.stem(word) for word in text.split()])
stem_words(input_str)
```

- **Lemmatization**

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    return " ".join([lemmatizer.lemmatize(word) for word in text.split()])
lemmatize_words(input_str)
```

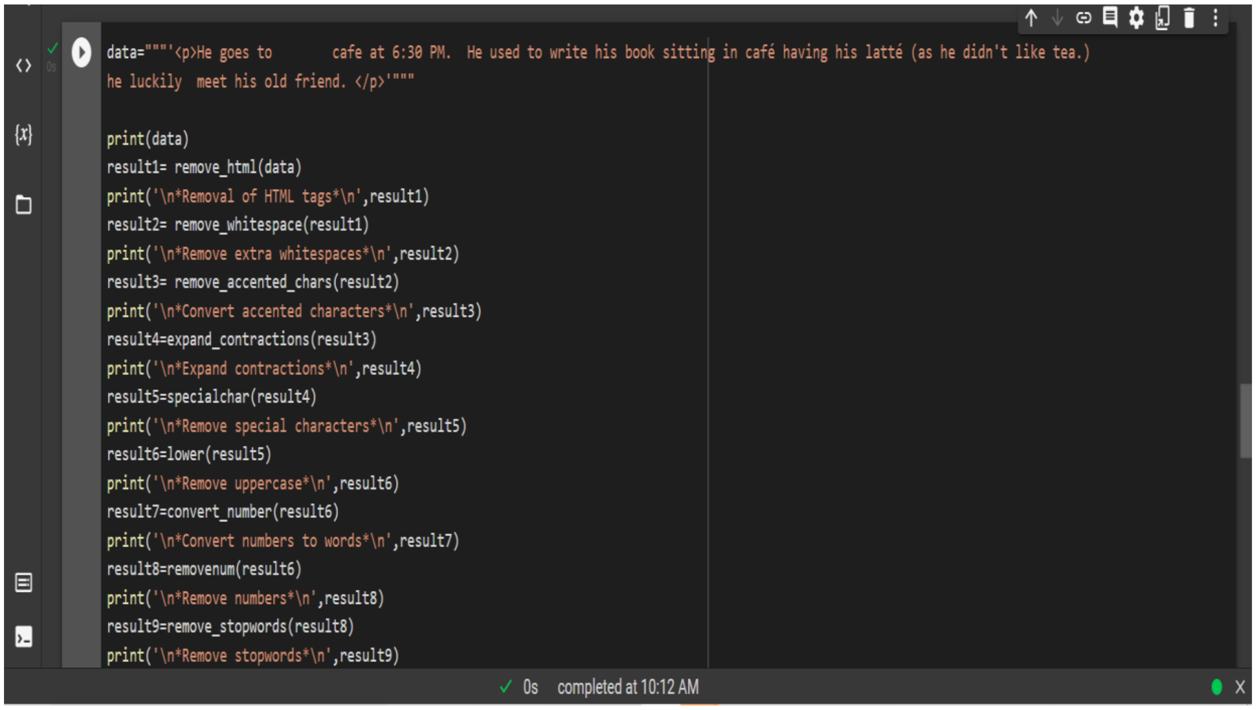
- **Tokenizing sentences**

```
from nltk.tokenize import sent_tokenize
def segment_sentence(sentence):
    return sent_tokenize(sentence)
```

- **Tokenize words**

```
from nltk import word_tokenize
def tokenize(sentences):
    words = word_tokenize(sentence)
    return words
```

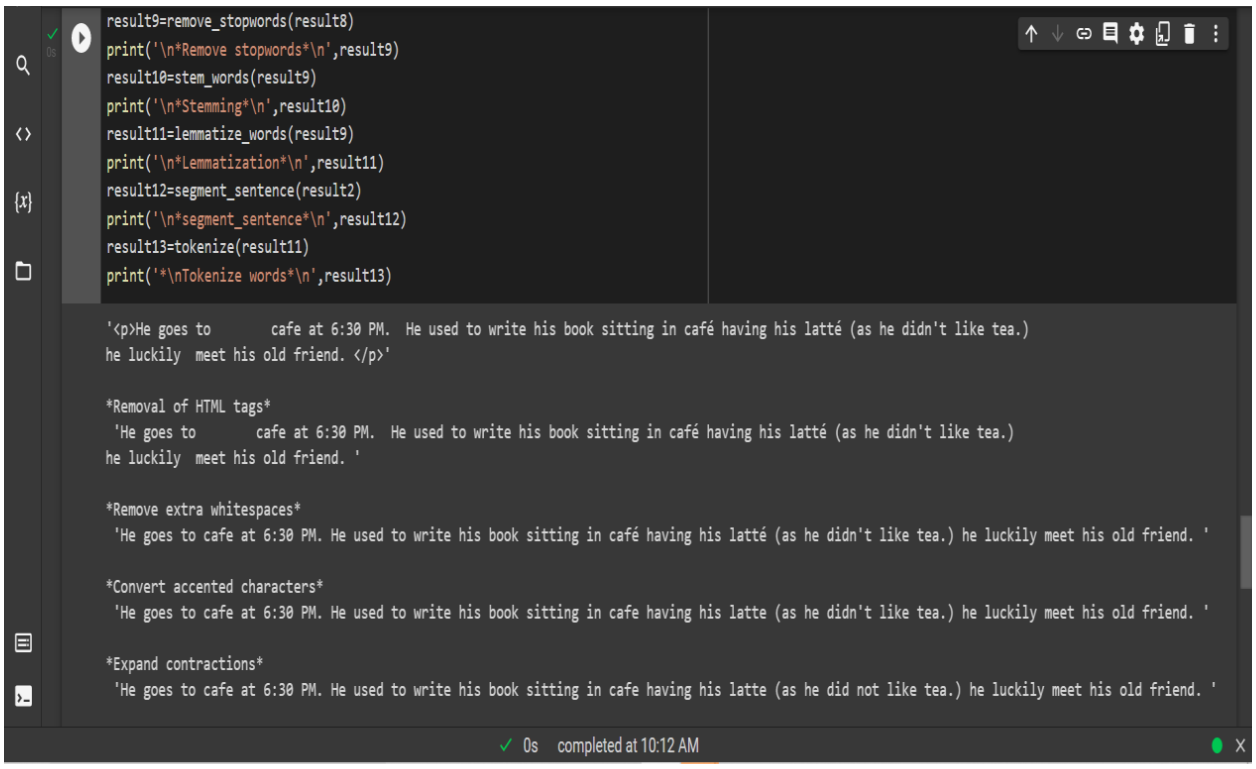
Output:



```
data="""'<p>He goes to        cafe at 6:30 PM.  He used to write his book sitting in café having his latté (as he didn't like tea.)
he luckily  meet his old friend. </p>'"""

print(data)
result1= remove_html(data)
print('\n*Removal of HTML tags*\n',result1)
result2= remove_whitespace(result1)
print('\n*Remove extra whitespaces*\n',result2)
result3= remove_accented_chars(result2)
print('\n*Convert accented characters*\n',result3)
result4=expand_contractions(result3)
print('\n*Expand contractions*\n',result4)
result5=specialchar(result4)
print('\n*Remove special characters*\n',result5)
result6=lower(result5)
print('\n*Remove uppercase*\n',result6)
result7=convert_number(result6)
print('\n*Convert numbers to words*\n',result7)
result8=removenum(result6)
print('\n*Remove numbers*\n',result8)
result9=remove_stopwords(result8)
print('\n*Remove stopwords*\n',result9)
```

✓ 0s completed at 10:12 AM



```
result9=remove_stopwords(result8)
print('\n*Remove stopwords*\n',result9)
result10=stem_words(result9)
print('\n*Stemming*\n',result10)
result11=lemmatize_words(result9)
print('\n*Lemmatization*\n',result11)
result12=segment_sentence(result2)
print('\n*segment_sentence*\n',result12)
result13=tokenize(result11)
print('*\nTokenize words*\n',result13)
```

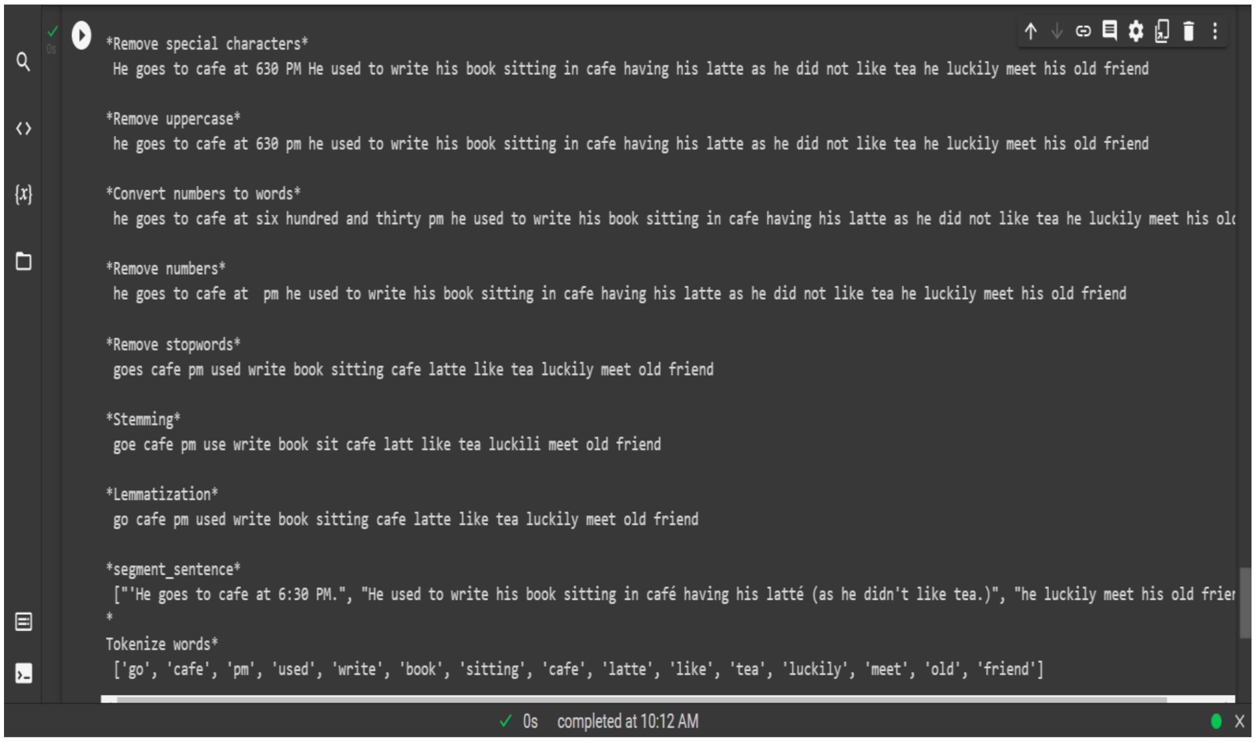
```
'<p>He goes to       cafe at 6:30 PM.  He used to write his book sitting in café having his latté (as he didn't like tea.)
he luckily  meet his old friend. </p>'

*Removal of HTML tags*
 'He goes to       cafe at 6:30 PM.  He used to write his book sitting in café having his latté (as he didn't like tea.)
he luckily  meet his old friend. '

*Remove extra whitespaces*
 'He goes to cafe at 6:30 PM. He used to write his book sitting in café having his latté (as he didn't like tea.) he luckily meet his old friend. '

*Convert accented characters*
 'He goes to cafe at 6:30 PM. He used to write his book sitting in cafe having his latte (as he didn't like tea.) he luckily meet his old friend. '

*Expand contractions*
 'He goes to cafe at 6:30 PM. He used to write his book sitting in cafe having his latte (as he did not like tea.) he luckily meet his old friend. '
```



```
*Remove special characters*
 He goes to cafe at 630 PM He used to write his book sitting in cafe having his latte as he did not like tea he luckily meet his old friend

*Remove uppercase*
 he goes to cafe at 630 pm he used to write his book sitting in cafe having his latte as he did not like tea he luckily meet his old friend

*Convert numbers to words*
 he goes to cafe at six hundred and thirty pm he used to write his book sitting in cafe having his latte as he did not like tea he luckily meet his old

*Remove numbers*
 he goes to cafe at  pm he used to write his book sitting in cafe having his latte as he did not like tea he luckily meet his old friend

*Remove stopwords*
 goes cafe pm used write book sitting cafe latte like tea luckily meet old friend

*Stemming*
 goe cafe pm use write book sit cafe latt like tea luckili meet old friend

*Lemmatization*
 go cafe pm used write book sitting cafe latte like tea luckily meet old friend

*segment_sentence*
 ["'He goes to cafe at 6:30 PM.", "He used to write his book sitting in café having his latté (as he didn't like tea.)", "he luckily meet his old frie
*
Tokenize words*
 ['go', 'cafe', 'pm', 'used', 'write', 'book', 'sitting', 'cafe', 'latte', 'like', 'tea', 'luckily', 'meet', 'old', 'friend']
```

Conclusion: Hence we have performed preprocessing of text and converted the given text into standardized text which can be used for further analysis.

Real Life Application:

- **Speech recognition – The task of converting voice data to text data.**
- **Sentiment analysis- The task of extracting qualities like attitudes, emotions, etc. The most basic task in sentiment analysis is to classify the polarity of a sentence that is positive, negative, or neutral.**
- **Natural language generation – The task of producing text data from some structured data.**
- **Part-of-speech (POS) tagging – The task of tagging the Part of Speech of a particular word in a sentence based on its definition and context.**
