MES Wadia College of Engineering Pune-01
Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Subject: LP_VI (EL-V NLP)**

Assignment No. 3

Aim: Perform text cleaning, perform lemmatization (any method), remove stop words (any method), label encoding. Create representations using TF-IDF. Save outputs.

Dataset: https://github.com/PICT-NLP/BE-NLP-Elective/blob/main/3-Preprocessing/News_dataset.pickle

Theory:

Text Cleaning:

Lowercase / UpperCase:

It helps to maintain the consistency flow during the NLP tasks and text mining. The **lower()** function makes the whole process quite straightforward.

Punctuation Removal:

in this step we will be removing all punctuations, because the punctuation to the sentence adds up noise that brings ambiguity while training the model.

HTML Code and URL Links:

we can simply use the following code :

text = re.sub(r'^https?:\/\/.*[\r\n]*', '', text, flags=re.MULTILINE) text = re.sub('<.*?>+', '', text)

Spell Checks :

This guide uses the pyspellchecker package for spelling correction.
Here's the code that helps us!

`from spellchecker import SpellChecker

```
spelling = SpellChecker()
def spelling_checks(text):
correct_result = []
typo_words = spelling.unknown(text.split())
for word in text.split():
if word in typo_words:
correct_result.append(spelling.correction(word))
else:
```

```
correct_result.append(word)
return " ".join(correct_result)
`
```

Tokenization:

Tokenizing is like splitting a whole sentence into words. You can consider a simple separator for this purpose. Most of these problems can be solved by using the nltk library. The word_tokenize module breaks the words into tokens and these words act as an input for the normalization and cleaning process. It can further be used to convert a string (text) into numeric data so that machine learning models can digest it.

Removing Stop Words:

English is one of the most common languages, especially in the world of social media. For instance, "a," "our," "for," "in," etc. are in the set of most commonly used words. Removing these words helps the model to consider only key features. These words also don't carry much information. By eliminating them, data scientists can focus on the important words.

Check out the list of the stop words nltk provides.

```
stopwordslist = stopwords.words('english')

print(stopwordslist)

print('Total:',len(stopwordslist))
```

Normalization:

Normalization is an advanced step in cleaning to maintain uniformity. It brings all the words under the roof by adding stemming and lemmatization.

Lemmatization is quite similar to the Stemming.

Lemmatization:

Lemmatization is a text normalization technique used in Natural Language Processing (NLP), that switches any kind of a word to its base root mode. Lemmatization is responsible for grouping different inflected forms of words into the root form, having the same meaning.

Lemma: A basic word form (e.g. infinitive or singular nominative noun) that is used to represent all forms of the same word

An inflected form of a word has a changed spelling or ending that shows the way it is used in sentences:

"Finds" and "found" are inflected forms of "find".

The main difference between Stemming and lemmatization is that it produces the root word, which has a meaning.

9 different approaches to perform Lemmatization:

1. WordNet
2. WordNet (with POS tag)
3. TextBlob

4. TextBlob (with POS tag)
5. spaCy
6. TreeTagger
7. Pattern
8. Gensim
9. Stanford CoreNLP

Stop Words:

Stop words are common words that don't play a big role in classification of text. Search engines often ignore them because they don't really help narrow down the results for a given search phrase. *A, the, it, he, she,* and *an* are common stop words in English.

Because they don't provide much value, it's beneficial to remove stop words before processing text for natural language processing (NLP) tasks. Imagine how much larger your databases would have to be if all these common words were included?

The Python NLTK library contains a default list of stop words. To remove stop words, you need to divide your text into tokens (words), and then check if each token matches words in your list of stop words. If the token matches a stop word, you ignore the token. Otherwise you add the token to the list of *valid* words.

The NLTK library supports stop word removal from a variety of languages.

To see the list of languages supported by NLTK for stop word removal, call the fieldids() function from the stopwords class of the nltk.corpus module as shown in the following script.

```
from nltk.corpus import stopwords

print(stopwords.fieldids())
```

Here is a list of all the languages supported by the NLKT library for stop words removal.

['arabic', 'azerbaijani', 'danish', 'dutch', 'english', 'finnish', 'french', 'german', 'greek', 'hungarian', 'indonesian', 'italian', 'kazakh', 'nepali', 'norwegian', 'portuguese', 'romanian', 'russian', 'slovene', 'spanish', 'swedish', 'tajik', 'turkish']

To get a list of English stop words, you have to pass 'english' as a parameter to the stopwords.words() function as shown below.

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

Conclusion:

Questions:

1. **What is label encoding?**
2. **Which are the lemmatization methods? Explain any one of them.**
3. **What is the need of text cleaning? How it is done?**