**Text Pre-processing Application Exercise**

In this exercise you will complete steps in order to pre-process text for ML models. You may use Jupyter Notebook and Python to complete this assignment.

You are recommended to submit your notebook (.ipynb). Please note your code must be **well and clearly commented** for each exercise or there will be mark deduction.

Assignment essential packages:

*import nltk*

*import pandas as pd*

*import re*

*import string*

The YouTube Spam Collection Dataset in this exercise comes from *Machine Learning Repository*. Read more about the dataset:

[https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection#](https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection)

You must at least **use one of the files provided at Moodle** in this exercise. Files are also available at the following link:

<https://archive.ics.uci.edu/ml/machine-learning-databases/00380/>

1. **Read your data (100/10)**
2. Read in your text data (files provided at Moodle) and save it into a data frame called “dataset”.

Tip: You may use read\_csv in Pandas to read in the file.

1. Figure out how many rows and columns are in *“dataset”.*

Note: Study the dataset. Then, you must choose the column(s) needed for pre-processing.

Note: Since, you are not fitting ML model in this exercise, you may only proceed with a column containing text messages.

1. Check for any missing data in your data frame and display the result.

Tip: You may use isnull() to check for missing values in a column.

1. **Remove Punctuations (100/10)**
2. Write a Python function to remove punctuations from text and save the transformed text in a new column in your data frame called “*punct\_text*”.
3. Then, printout the first few rows of your data frame.

Note: Make sure the text is in the format of sentences NOT separated characters after this step.

1. **Tokenization (100/20)**
2. Write a function to tokenize the text and save the tokenized text in a new column in your data frame called “*token\_text*”.

Tip: You can specify a Regex to split the sentences based on non-word characters or use *word\_tokenize* from *nltk.tokenize*.

1. Then, printout the first few rows.
2. You may use *lower()* function such that all the text converts to lower case and **explain why this is important**.
3. **Remove Stopwords (100/10)**
4. Write a function to remove stopwords from “token\_text” column in your data frame and save the cleaned text in a new column in your data frame called “*nostop\_text*”.

Note: Alternatively, you can write a function that performs all the above-mentioned steps (2 to 4) and clean the text, make it ready for further steps.

1. Then, printout the first few rows.
2. **Stem (100/10)**
3. From *NLTK* *PorterStemmer*, use *stem* function in order to stem “*nonstop\_text*”.
4. You may write a function to stem each word in “*nonstop\_text*”. Then, save the stemmed text in a new column called “*stemmed\_text*”.
5. Printout the first few rows of your data frame.
6. **Lemmatize (100/10)**
7. From *NLTK WordNetLemmatizer,* use *lemmatize* function to lemmatize “*nonstop\_text*”.
8. You may write a function to lemmatize each word in “*nonstop\_text*”. Then, save the lemmatized text in a new column called “*lemmatized\_text*”.
9. Printout the first few rows of your data frame.
10. **Vectorizing (100/30)**
11. Import *CountVectorizer* from *sklearn*.

*from sklearn.feature\_extraction.text import CountVectorizer*

1. Then, instanciate CountVectorizer object.
2. Within *CountVectorizer* object you can set *analyzer* parameter to pass on your function from the previous steps to clean the text first.
3. Within *CountVectorizer* object use *fit\_transform* to vectorize the text and save it to counts\_text (use either *stemmed\_text or lemmatized\_text*).
4. Print the number of rows and columns in *counts\_text* and feature names:

*Print(counts\_text.shape)*

*Print(counts\_text.get\_feature\_names())*

1. Wihtin *CountVectorizer* set *ngram\_range* parameter such that only **two adjacent words** are being vectorized in the text (use either *stemmed\_text or lemmatized\_text*) and save the result into *ngramcounts\_text*.
2. Print the number of rows and columns in *ngramcounts\_text* and feature names:

*Print(ngramcounts\_text.shape)*

*Print(ngramcounts\_text.get\_feature\_names())*