Assignment 1.1 Text Preprocessing using NLP Techniques

Poetry Foundation Poems

`Link to the dataset:` https://www.kaggle.com/datasets/tgdivy/poetry-foundation-poems

Assignment Instructions

Objective

- The purpose of this assignment is to apply tokenization, stemming, lemmatization, and stop word removal techniques on an available dataset to preprocess the text data. Through this assignment, students will gain hands-on experience in text preprocessing using NLP techniques.

Instructions

- Prior to beginning work on this assignment, review the week's lab session.

Dataset Selection:

- Choose an available dataset for text preprocessing. You can use a publicly available dataset.
- The dataset can be in a file format such as CSV, JSON, TXT, or any other commonly used format.
- Ensure that the dataset contains textual data suitable for tokenization, stemming, lemmatization, and stop word removal.

Data Exploration:

- Begin by exploring the dataset to understand its structure, format, and attributes.
- Identify the specific column(s) or attribute(s) that contain the textual data you will be working with.

Text Preprocessing:

- Perform tokenization on the textual data. Split the text into individual tokens (words, punctuation, or other meaningful units) using an appropriate tokenization technique or library.
- Apply stemming to the tokens. Reduce each word to its base or root form using a suitable stemming algorithm or library.
- Implement lemmatization on the tokens. Transform each word into its canonical or dictionary form using a reliable lemmatization technique or library.
- Remove stop words from the tokenized text. Remove common words (e.g., "the," "is," "and") that do not carry significant meaning using a standard stop word list or library.

Documentation:

- Prepare a report documenting your approach and findings.
- Include an introduction to the dataset, describing its source and content.
- Present the code or scripts used for text preprocessing, clearly indicating the steps followed for tokenization, stemming, lemmatization, and stop word removal.
- Provide visualizations or statistics that demonstrate the impact of preprocessing on the dataset, such as word frequency before and after preprocessing.
- Discuss any insights or observations gained through the preprocessing process.

Conclusion:

- Summarize the importance of text preprocessing in NLP and how it enhances the quality and effectiveness of subsequent text analysis or modeling tasks.
- Explain the challenges encountered during the preprocessing process and discuss potential strategies to overcome them.
- Conclude with a discussion on the potential applications and benefits of text preprocessing techniques in various NLP domains.

Data Exploration

In [23]:
```
# NLP libraries and tools
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [24]:
```
# loading the dataset
df = pd.read_csv('PoetryFoundationData.csv')
df.head()
```

Out[24]:

	Unnamed: 0	Title	Poem	Poet	Tags
0	0	\r\r\n Objects Used to Prop...	\r\r\nDog bone, stapler,\r\r\ncribbage board, ...	Michelle Menting	NaN
1	1	\r\r\n The New Church\r\r\n...	\r\r\nThe old cupola glinted above the clouds,...	Lucia Cherciu	NaN
2	2	\r\r\n Look for Me\r\r\n ...	\r\r\nLook for me under the hood\r\r\nof that ...	Ted Kooser	NaN
3	3	\r\r\n Wild Life\r\r\n ...	\r\r\nBehind the silo, the Mother Rabbit\r\r\n...	Grace Cavalieri	NaN
4	4	\r\r\n Umbrella\r\r\n ...	\r\r\nWhen I push your button\r\r\nyou fly off...	Connie Wanek	NaN

In [25]:
```
# getting some general information about this dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13854 entries, 0 to 13853
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  13854 non-null  int64 
 1   Title       13854 non-null  object
 2   Poem        13854 non-null  object
 3   Poet        13854 non-null  object
 4   Tags        12899 non-null  object
dtypes: int64(1), object(4)
memory usage: 541.3+ KB
```

In [26]:
```
# getting the length of the dataset
len(df)
```

Out[26]: 13854

In [27]:
```
# dropping the column 'Unnamed: 0'
df.drop('Unnamed: 0', axis=1, inplace=True)
df.columns
```

Out[27]: Index(['Title', 'Poem', 'Poet', 'Tags'], dtype='object')

In [28]:
```
# getting the unique counts of the all the columns
for col in df.columns:
    print(f'{col} has {df[col].nunique()} unique vals')
```

```
Title has 13240 unique vals
Poem has 13754 unique vals
Poet has 3128 unique vals
Tags has 8297 unique vals
```

In [29]:
```
# checking for missing values or nan values
df.isnull().sum()
```

Out[29]:
```
Title      0
Poem       0
Poet       0
Tags     955
dtype: int64
```

In [30]:
```
# printing one full poem to see how the data looks like
print(df['Poem'][0])
```

```
Dog bone, stapler,
cribbage board, garlic press
     because this window is loose—lacks
suction, lacks grip.
Bungee cord, bootstrap,
dog leash, leather belt
     because this window had sash cords.
They frayed. They broke.
Feather duster, thatch of straw, empty
bottle of Elmer's glue
     because this window is loud—its hinges clack
open, clack shut.
Stuffed bear, baby blanket,
single crib newel
     because this window is split. It's dividing
in two.
Velvet moss, sagebrush,
willow branch, robin's wing
     because this window, it's pane-less. It's only
a frame of air.
```

Text Preprocessing

In [31]:
```
# downloading the punkt tokenizer
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /home/parker/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[31]:
```
True
```

Tokenization

In [32]:
```
# tokenizing the poem column
df['Poem'] = df['Poem'].apply(word_tokenize)
df['Poem']
```

Out[32]:
```
0        [Dog, bone, ,, stapler, ,, cribbage, board, ,,...
1        [The, old, cupola, glinted, above, the, clouds...
2        [Look, for, me, under, the, hood, of, that, ol...
3        [Behind, the, silo, ,, the, Mother, Rabbit, hu...
4        [When, I, push, your, button, you, fly, off, t...
                              ...
13849    [We, 'd, like, to, talk, with, you, about, fea...
13850                                                   []
13851                                                   []
13852    [Philosophic, in, its, complex, ,, ovoid, empt...
13853    [Dear, Writers, ,, I, ', m, compiling, the, fi...
Name: Poem, Length: 13854, dtype: object
```

In [33]:
```
# making all the words to lower case
df['Poem'] = df['Poem'].apply(lambda x: [word.lower() for word in x])
```

Stemming

In [34]:
```
# init stemmer
stemmer = PorterStemmer()

# function to stem the words
def stem_words(tokens):
    return [stemmer.stem(token) for token in tokens]

# applying the funct
df['Poem_Stemmed'] = df['Poem'].apply(stem_words)

# display
df[['Poem', 'Poem_Stemmed']].head()
```

Out[34]:

	Poem	Poem_Stemmed
0	[dog, bone, ,, stapler, ,, cribbage, board, ,,...	[dog, bone, ,, stapler, ,, cribbag, board, ,, ...
1	[the, old, cupola, glinted, above, the, clouds...	[the, old, cupola, glint, abov, the, cloud, ,,...
2	[look, for, me, under, the, hood, of, that, ol...	[look, for, me, under, the, hood, of, that, ol...
3	[behind, the, silo, ,, the, mother, rabbit, hu...	[behind, the, silo, ,, the, mother, rabbit, hu...
4	[when, i, push, your, button, you, fly, off, t...	[when, i, push, your, button, you, fli, off, t...

In [35]:
```
# downloading word net data
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package wordnet to /home/parker/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /home/parker/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[35]:
```
True
```

Lemmatizing

In [36]:
```
# lemmatizing
lemmatizer = WordNetLemmatizer()

# function to lemmatize the words
def lemmatize_words(tokens):
    return [lemmatizer.lemmatize(token) for token in tokens]

# applying the function
df['Poem_Lemmatized'] = df['Poem'].apply(lemmatize_words)

# display
df[['Poem', 'Poem_Lemmatized']].head()
```

Out[36]:

	Poem	Poem_Lemmatized
0	[dog, bone, ,, stapler, ,, cribbage, board, ,,...	[dog, bone, ,, stapler, ,, cribbage, board, ,,...
1	[the, old, cupola, glinted, above, the, clouds...	[the, old, cupola, glinted, above, the, cloud,...
2	[look, for, me, under, the, hood, of, that, ol...	[look, for, me, under, the, hood, of, that, ol...
3	[behind, the, silo, ,, the, mother, rabbit, hu...	[behind, the, silo, ,, the, mother, rabbit, hu...
4	[when, i, push, your, button, you, fly, off, t...	[when, i, push, your, button, you, fly, off, t...

In [37]:
```
# displaying the Poem_Lemmatized column and the Poem_steemed column side

df[['Poem_Lemmatized', 'Poem_Stemmed']].head()
```

Out[37]:

	Poem_Lemmatized	Poem_Stemmed
0	[dog, bone, ,, stapler, ,, cribbage, board, ,,...	[dog, bone, ,, stapler, ,, cribbag, board, ,, ...
1	[the, old, cupola, glinted, above, the, cloud,...	[the, old, cupola, glint, abov, the, cloud, ,,...
2	[look, for, me, under, the, hood, of, that, ol...	[look, for, me, under, the, hood, of, that, ol...
3	[behind, the, silo, ,, the, mother, rabbit, hu...	[behind, the, silo, ,, the, mother, rabbit, hu...
4	[when, i, push, your, button, you, fly, off, t...	[when, i, push, your, button, you, fli, off, t...

In [38]:
```
# priniting the first poem to see the difference
print(df['Poem'][1])
print(df['Poem_Lemmatized'][1])
print(df['Poem_Stemmed'][1])
```

```
['the', 'old', 'cupola', 'glinted', 'above', 'the', 'clouds', ',', 'shon
e', 'among', 'fir', 'trees', ',', 'but', 'it', 'took', 'him', 'an', 'hou
r', 'for', 'the', 'half', 'mile', 'all', 'the', 'way', 'up', 'the', 'hil
l', '.', 'as', 'he', 'trailed', ',', 'the', 'village', 'passed', 'him', 'b
y', ',', 'greeted', 'him', ',', 'asked', 'about', 'his', 'health', ',', 'b
ut', 'everybody', 'hurried', 'to', 'catch', 'the', 'mass', ',', 'left', 'h
im', 'leaning', 'against', 'fences', ',', 'measuring', 'the', 'road', 'wit
h', 'the', 'walking', 'stick', 'he', 'sculpted', '.', 'he', 'yearned', 'fo
r', 'the', 'day', 'when', 'the', 'new', 'church', 'would', 'be', 'built-ri
ght', 'across', 'the', 'road', '.', 'now', 'it', 'rises', 'above', 'the',
'moon', ':', 'saints', 'in', 'frescoes', 'meet', 'the', 'eye', ',', 'and',
'only', 'the', 'rain', 'has', 'started', 'to', 'cut', 'through', 'the', 's
hingles', 'on', 'the', 'roof', 'of', 'his', 'empty', 'house', '.', 'the',
'apple', 'trees', 'have', 'taken', 'over', 'the', 'sky', ',', 'sequestere
d', 'the', 'gate', ',', 'sidled', 'over', 'the', 'porch', '.']
['the', 'old', 'cupola', 'glinted', 'above', 'the', 'cloud', ',', 'shone',
'among', 'fir', 'tree', ',', 'but', 'it', 'took', 'him', 'an', 'hour', 'fo
r', 'the', 'half', 'mile', 'all', 'the', 'way', 'up', 'the', 'hill', '.',
'a', 'he', 'trailed', ',', 'the', 'village', 'passed', 'him', 'by', ',',
'greeted', 'him', ',', 'asked', 'about', 'his', 'health', ',', 'but', 'eve
rybody', 'hurried', 'to', 'catch', 'the', 'mass', ',', 'left', 'him', 'lea
ning', 'against', 'fence', ',', 'measuring', 'the', 'road', 'with', 'the',
'walking', 'stick', 'he', 'sculpted', '.', 'he', 'yearned', 'for', 'the',
'day', 'when', 'the', 'new', 'church', 'would', 'be', 'built-right', 'acro
ss', 'the', 'road', '.', 'now', 'it', 'rise', 'above', 'the', 'moon', ':',
'saint', 'in', 'fresco', 'meet', 'the', 'eye', ',', 'and', 'only', 'the',
'rain', 'ha', 'started', 'to', 'cut', 'through', 'the', 'shingle', 'on',
'the', 'roof', 'of', 'his', 'empty', 'house', '.', 'the', 'apple', 'tree',
'have', 'taken', 'over', 'the', 'sky', ',', 'sequestered', 'the', 'gate',
',', 'sidled', 'over', 'the', 'porch', '.']
['the', 'old', 'cupola', 'glint', 'abov', 'the', 'cloud', ',', 'shone', 'a
mong', 'fir', 'tree', ',', 'but', 'it', 'took', 'him', 'an', 'hour', 'fo
r', 'the', 'half', 'mile', 'all', 'the', 'way', 'up', 'the', 'hill', '.',
'as', 'he', 'trail', ',', 'the', 'villag', 'pass', 'him', 'by', ',', 'gree
t', 'him', ',', 'ask', 'about', 'hi', 'health', ',', 'but', 'everybodi',
'hurri', 'to', 'catch', 'the', 'mass', ',', 'left', 'him', 'lean', 'agains
t', 'fenc', ',', 'measur', 'the', 'road', 'with', 'the', 'walk', 'stick',
'he', 'sculpt', '.', 'he', 'yearn', 'for', 'the', 'day', 'when', 'the', 'n
ew', 'church', 'would', 'be', 'built-right', 'across', 'the', 'road', '.',
'now', 'it', 'rise', 'abov', 'the', 'moon', ':', 'saint', 'in', 'fresco',
'meet', 'the', 'eye', ',', 'and', 'onli', 'the', 'rain', 'ha', 'start', 't
o', 'cut', 'through', 'the', 'shingl', 'on', 'the', 'roof', 'of', 'hi', 'e
mpti', 'hous', '.', 'the', 'appl', 'tree', 'have', 'taken', 'over', 'the',
'sky', ',', 'sequest', 'the', 'gate', ',', 'sidl', 'over', 'the', 'porch',
'.']
```

Removing any stop words

In [39]:
```
# download
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /home/parker/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[39]: True

In [40]:
```
# making a list of stopwords
stop_words = set(stopwords.words('english'))
```

In [41]:
```
# function to remove stopwords from the tokenized text
def remove_stopwords(tokens):
    return [token for token in tokens if token.lower() not in stop_words]

# applying the function
df['Poem_No_Stopwords'] = df['Poem'].apply(remove_stopwords)

# display
df[['Poem', 'Poem_No_Stopwords']].head()
```

Out[41]:

	Poem	Poem_No_Stopwords
0	[dog, bone, ,, stapler, ,, cribbage, board, ,,...	[dog, bone, ,, stapler, ,, cribbage, board, ,,...
1	[the, old, cupola, glinted, above, the, clouds...	[old, cupola, glinted, clouds, ,, shone, among...
2	[look, for, me, under, the, hood, of, that, ol...	[look, hood, old, chevrolet, settled, weeds, e...
3	[behind, the, silo, ,, the, mother, rabbit, hu...	[behind, silo, ,, mother, rabbit, hunches, lik...
4	[when, i, push, your, button, you, fly, off, t...	[push, button, fly, handle, ,, old, skin, bone...

In [42]:
```
# displaying the Poem_No_Stopwords column and the Poem column side by sid
print(df['Poem'][1])
print(df['Poem_No_Stopwords'][1])
```

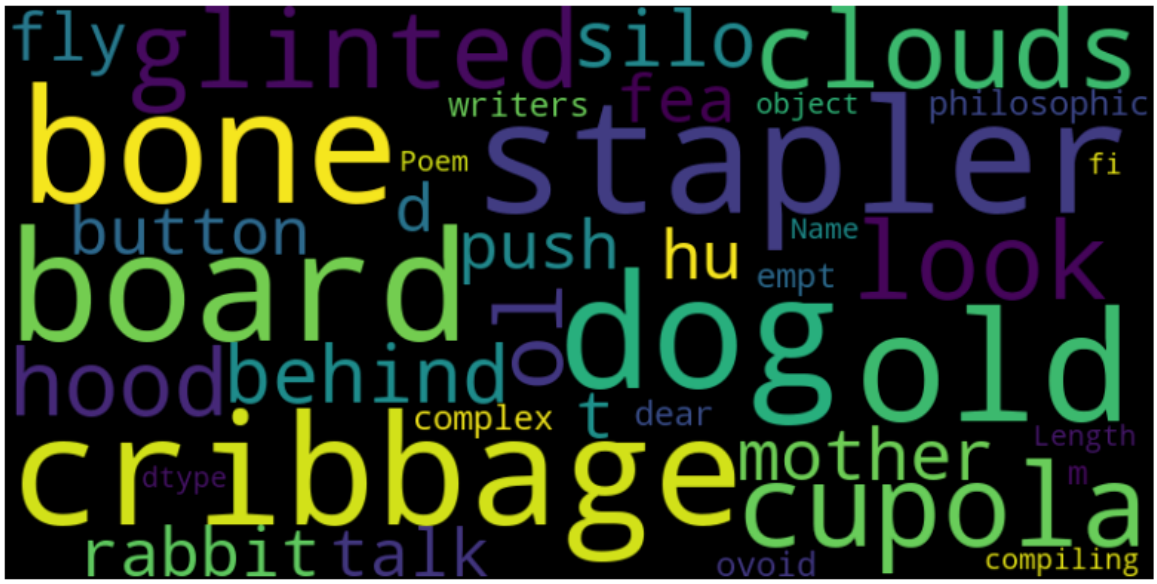
```
['the', 'old', 'cupola', 'glinted', 'above', 'the', 'clouds', ',', 'shon
e', 'among', 'fir', 'trees', ',', 'but', 'it', 'took', 'him', 'an', 'hou
r', 'for', 'the', 'half', 'mile', 'all', 'the', 'way', 'up', 'the', 'hil
l', '.', 'as', 'he', 'trailed', ',', 'the', 'village', 'passed', 'him', 'b
y', ',', 'greeted', 'him', ',', 'asked', 'about', 'his', 'health', ',', 'b
ut', 'everybody', 'hurried', 'to', 'catch', 'the', 'mass', ',', 'left', 'h
im', 'leaning', 'against', 'fences', ',', 'measuring', 'the', 'road', 'wit
h', 'the', 'walking', 'stick', 'he', 'sculpted', '.', 'he', 'yearned', 'fo
r', 'the', 'day', 'when', 'the', 'new', 'church', 'would', 'be', 'built-ri
ght', 'across', 'the', 'road', '.', 'now', 'it', 'rises', 'above', 'the',
'moon', ':', 'saints', 'in', 'frescoes', 'meet', 'the', 'eye', ',', 'and',
'only', 'the', 'rain', 'has', 'started', 'to', 'cut', 'through', 'the', 's
hingles', 'on', 'the', 'roof', 'of', 'his', 'empty', 'house', '.', 'the',
'apple', 'trees', 'have', 'taken', 'over', 'the', 'sky', ',', 'sequestere
d', 'the', 'gate', ',', 'sidled', 'over', 'the', 'porch', '.']
['old', 'cupola', 'glinted', 'clouds', ',', 'shone', 'among', 'fir', 'tree
s', ',', 'took', 'hour', 'half', 'mile', 'way', 'hill', '.', 'trailed',
',', 'village', 'passed', ',', 'greeted', ',', 'asked', 'health', ',', 'ev
erybody', 'hurried', 'catch', 'mass', ',', 'left', 'leaning', 'fences',
',', 'measuring', 'road', 'walking', 'stick', 'sculpted', '.', 'yearned',
'day', 'new', 'church', 'would', 'built-right', 'across', 'road', '.', 'ri
ses', 'moon', ':', 'saints', 'frescoes', 'meet', 'eye', ',', 'rain', 'star
ted', 'cut', 'shingles', 'roof', 'empty', 'house', '.', 'apple', 'trees',
'taken', 'sky', ',', 'sequestered', 'gate', ',', 'sidled', 'porch', '.']
```

In [43]:
```
# now making a word map to see the most common words in the dataset

from wordcloud import WordCloud

# making a word map for the Poem column
wordcloud = WordCloud(width = 800, height = 400, random_state=21, max_fon
plt.figure(figsize=(10, 7))
```

```
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```



Documentation

Introduction to the Dataset

The dataset that I decided to use for this assignment is a collection of poems from the Poetry Foundation. The dataset contains 13,854 poems and includes the following columns, Title, Poem, Poet, and Tags. The dataset was downloaded from Kaggle, and comes in a CSV file format. The data set is used for Poet classification. I did not have to clean the data, as it was already clean and ready to use, containing no missing values, except in the Tags column, which was not needed for the analysis, and text processing outlined in the instuctions.

Working with the Dataset

Working with the dataset was fairly easy as mentioned earlier, there was no missing values from the data set, and the data was already clean. The only thing that I had to do was to remove the Tags column, as it was not needed for the analysis. I also had to do some text processing on the Poem column, which included removing punctuation, making all the text lowercase, and tokenizing the text. Consider that the data set was fairly easy to work with I was able to use the library NLTK to do the text processing and tokenization.

During the cleaning process, I used the NLTK 'putkt' function to remove punctuation from the Poem column. I also used the 'lower' function to make all the text lowercase. I then used the 'word_tokenize' function to tokenize the text.

When working with the data set, I did not really run into any sorts of challanges other than having to pip install and do some research on the NLTK library to figure out how to

use the functions that I needed to use, but the documentation was helpful and I was able to eventually figure out how to properly use them.

Conclusion and Final Thoughts

The importance of text pre-processing is critical when working with text data, as it will help the model to better understand the text data when making predictions. The importance of text pre-processing and tokenization when working with text data is important because it helps the model better understand the underlying context of the text data, and breaks down more complex wording into smaller more manageable pieces for the LLM to be able to understand. I think that even though there are multiple kinds of tokenization methods, they all end up doing the same thing with minor changes. After tokenizing the data, we then stem the data to remove any suffixes from the words, and then we can use the data to train a model. We are essentially breaking down the text futher and further with every step that we take so our data is smaller. Removing stoppwords is also important because it helps the model to better understand the text data, and helps to remove any words that are not needed for the analysis and prevents the model from being confused by the data.

In my case doing all of the data preprocesssing steps would help make my model better at predicting the poet, based off of their sequencing and word choice. However, I would like to see and I am curious how if the words were spoken, if the model would be able to predict the poet based off of their voice, or if it would be heavily reliant on the text data, before the data was preprocessed. I think that this would be an interesting experiment to try, and I would like to see how the model would perform in this scenario.