In [1]:
```
#downloading the packages
# uncomment all the lines to install the packages

#import nltk
#nltk.download('punkt')
#nltk.download('averaged_perceptron_tagger')

# open the nltk downloader
# note that the downloader might be minimized in your toolbar
# the downloader is a modal window, so the Jupyter notebook will wait for you to do
#nltk.download()

#!pip install spacy
```

In [2]:
```
# 1. Tokenize the texts in the text files.
```

In [3]:
```
#using nltk
import nltk
from nltk import word_tokenize
import pickle
# Open the input file for reading
# change the path of the input file according to your path location
with open("E:\\MS_Course_Notes\\COMP_293C\\Assignments\\Assignment_1\\input_data.tx
    input_text = file.read()

tokenized_text = word_tokenize(input_text)

# Specify the path for the output file to save the tokenized text
output_file_path = "E:\\MS_Course_Notes\\COMP_293C\\Assignments\\Assignment_1\\q1_t

# Write the tokenized text to the output file
with open(output_file_path, 'w') as output_file:
    output_file.write(str(tokenized_text))

print("Tokenized text written to:", output_file_path)
#print the tokenized text
#print(tokenized_text)
```

```
Tokenized text written to: E:\MS_Course_Notes\COMP_293C\Assignments\Assignment_1\q
1_tokenized_output.txt
```

In [4]:
```
#2. Count word frequencies in the texts.
```

In [5]:
```
from nltk.probability import FreqDist

freq_dist = FreqDist(tokenized_text)

# Convert the frequency distribution to a dictionary
freq_dict = dict(freq_dist)


# Specify the path for the output file to save the tokenized text
output_file_path = "E:\\MS_Course_Notes\\COMP_293C\\Assignments\\Assignment_1\\q2_t

# Write the tokenized text to the output file
with open(output_file_path, 'w') as output_file:
    output_file.write(str(freq_dict))

print("Count word frequencies written to:", output_file_path)
FreqDist(tokenized_text)
```

```
Count word frequencies written to: E:\MS_Course_Notes\COMP_293C\Assignments\Assign
ment_1\q2_frequency_output.txt
```

Out[5]:
```
FreqDist({',': 74, 'the': 70, 'to': 55, ':': 52, '.': 44, 'of': 41, 'your': 40, 'f
or': 35, 'are': 26, 'and': 23, ...})
```

In [6]:
```
#3. Perform part-of-speech (POS) tagging on the tokenized words.
```

In [7]:
```
pos_tag_text = nltk.pos_tag(tokenized_text)

# Specify the path for the output file to save the tokenized text
output_file_path = "E:\\MS_Course_Notes\\COMP_293C\\Assignments\\Assignment_1\\q3_p

# Write the tokenized text to the output file
with open(output_file_path, 'w') as output_file:
    output_file.write(str(pos_tag_text))

print("Perform part-of-speech (POS) tagging on the tokenized words written to:", ou

#print(pos_tag_text)
```

```
Perform part-of-speech (POS) tagging on the tokenized words written to: E:\MS_Cour
se_Notes\COMP_293C\Assignments\Assignment_1\q3_pos_output.txt
```

In [8]:
```
#4. Perform named entity recognition (NER) on the texts.
```

In [9]:
```
# Use NLTK's NER chunker
ner_tag_texts = nltk.chunk.ne_chunk(pos_tag_text)

# 'ner_tag_texts' now contains a tree structure with named entities recognized
# Extract named entities as a list:
named_entities = []
for subtree in ner_tag_texts:
    if isinstance(subtree, nltk.Tree):
        entity = " ".join([word for word, tag in subtree.leaves()])
        named_entities.append((entity, subtree.label()))

# Specify the path for the output file to save the tokenized text
output_file_path = "E:\\MS_Course_Notes\\COMP_293C\\Assignments\\Assignment_1\\q4_p

# Write the tokenized text to the output file
with open(output_file_path, 'w') as output_file:
    output_file.write(str(named_entities))

print("Perform named entity recognition (NER) on the texts written to:", output_fil
#print(named_entities)
```

```
Perform named entity recognition (NER) on the texts written to: E:\MS_Course_Notes
\COMP_293C\Assignments\Assignment_1\q4_pos_output.txt
```

In [10]:
```
# 5. Displaying the most frequent 10 words.
```

In [11]:
```
# NLP imports
import nltk
import spacy
from spacy import displacy
# general numerical and visualization imports
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
import numpy as np
```

In [12]:
```
#displaying the most frequent 10 words

from nltk import FreqDist
import matplotlib
matplotlib.use('Agg')  # 'Agg' backend for saving plots
import matplotlib.pyplot as plt

# Assuming 'tokens' is your list of tokens
word_freq = FreqDist(tokenized_text)

# Get the 10 most common words
most_common_words = word_freq.most_common(10)

all_fdist = pd.Series(dict(most_common_words))
# Setting fig and ax into variables
fig, ax = plt.subplots(figsize=(5,5))
# Plot with Seaborn plotting tools
plt.xticks(rotation = 60)
plt.title("Frequency -- Top 10 Words in the input text file",
fontsize = 25)
plt.xlabel("Words", fontsize = 25)
plt.ylabel("Frequency", fontsize = 25)
all_plot = sns.barplot(x = all_fdist.index, y = all_fdist.values,
ax=ax)
plt.xticks(rotation=50)
#to display in UI
#plt.show()

# Specify the path for the output image file where you want to save the plot
output_image_path = "E:\\MS_Course_Notes\\COMP_293C\\Assignments\\Assignment_1\\q5_

# Save the plot as an image file
plt.savefig(output_image_path, bbox_inches='tight')  # 'bbox_inches' prevents trimm

print("5. Displaying the most frequent 10 words saved as:", output_image_path)
```

```
5. Displaying the most frequent 10 words saved as: E:\MS_Course_Notes\COMP_293C\As
signments\Assignment_1\q5_frequent_10_words_bar_plot.png
```

In [13]:
```
#6. Compute a word cloud from the word frequency distribution.
```

In [14]:
```
# displaying a WordCloud
from wordcloud import WordCloud
wordcloud = WordCloud(background_color = 'white',

max_words = 25,
relative_scaling = 0,
width = 600,height = 300,
max_font_size = 150,
colormap = 'Dark2',
min_font_size = 10).generate_from_frequencies(all_fdist)
# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
#to display in UI
#plt.show()
# Specify the path for the output image file where you want to save the plot
output_image_path = "E:\\MS_Course_Notes\\COMP_293C\\Assignments\\Assignment_1\\q6_

# Save the plot as an image file
plt.savefig(output_image_path, bbox_inches='tight')  # 'bbox_inches' prevents trimm

print("6. Compute a word cloud from the word frequency distribution saved as:", out
```

```
6. Compute a word cloud from the word frequency distribution saved as: E:\MS_Cours
e_Notes\COMP_293C\Assignments\Assignment_1\q6_word_cloud_frequency.png
```

In [15]:
```
#7. Display the frequencies of the parts of speech.
```

In [16]:
```
from collections import Counter
import matplotlib.pyplot as plt
import seaborn as sns

# Count the occurrences of each POS tag
pos_freq = Counter(tag for word, tag in pos_tag_text)

# Extract the tags and their frequencies
tags = list(pos_freq.keys())
frequencies = list(pos_freq.values())

# Define a custom color palette using seaborn
custom_palette = sns.color_palette("Set1", len(tags))

# Create a bar plot
plt.figure(figsize=(12, 6))

# Iterate through tags and frequencies and assign custom colors
for i, (tag, freq) in enumerate(zip(tags, frequencies)):
    plt.bar(tag, freq, color=custom_palette[i])

plt.xlabel("Part of Speech")
plt.ylabel("Frequency")
plt.title("Part of Speech Frequency in input text file")
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.tight_layout()  # Ensure labels are not cut off
#to show in the front end
#plt.show()

# Specify the path for the output image file where you want to save the plot
output_image_path = "E:\\MS_Course_Notes\\COMP_293C\\Assignments\\Assignment_1\\q7_

# Save the plot as an image file
plt.savefig(output_image_path, bbox_inches='tight')  # 'bbox_inches' prevents trimm

print("7. Display the frequencies of the parts of speech. saved as:", output_image_
```

```
7. Display the frequencies of the parts of speech. saved as: E:\MS_Course_Notes\CO
MP_293C\Assignments\Assignment_1\q7_frequency_pos.png
```

In [ ]:
```

```