Figures

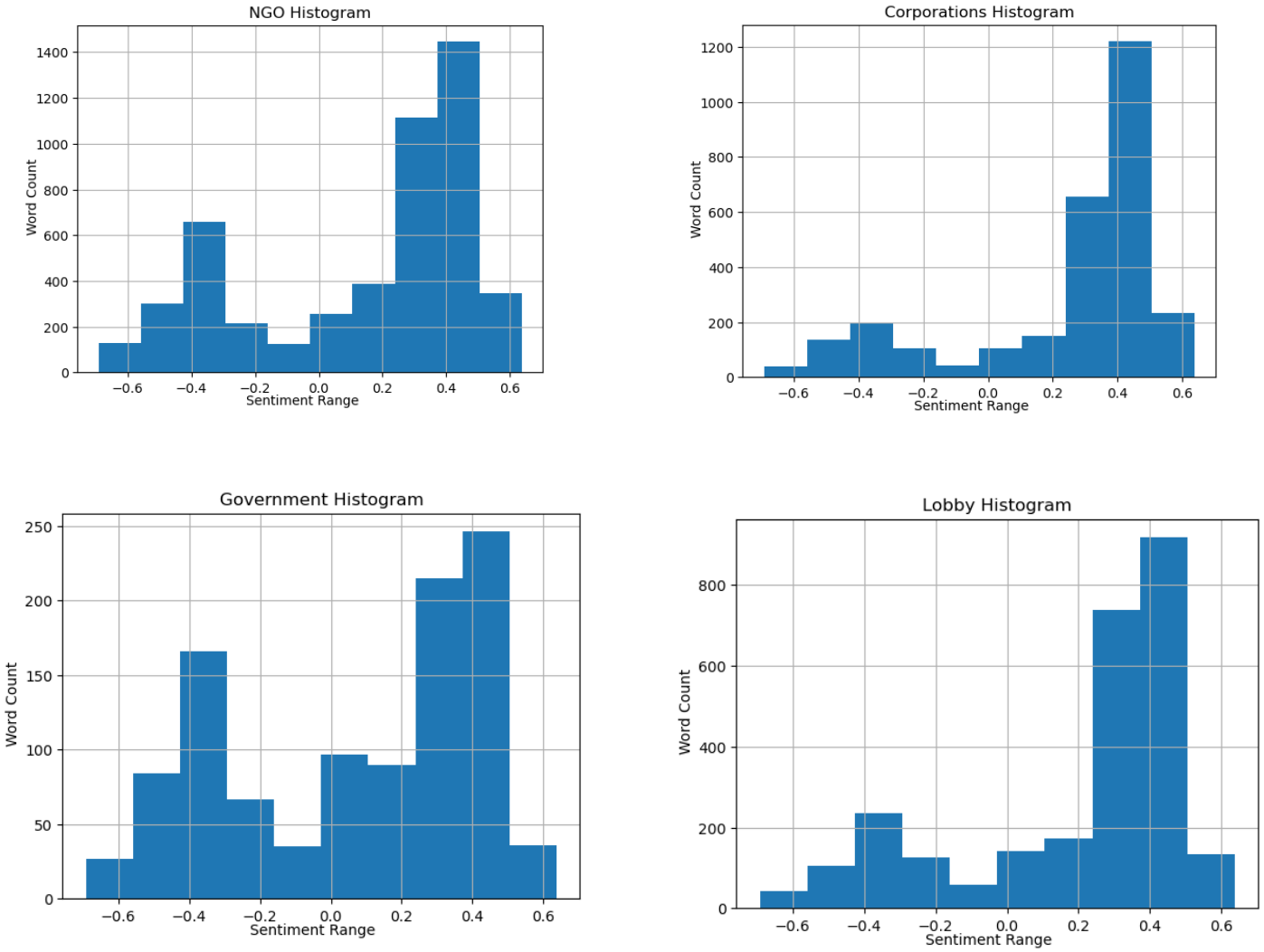
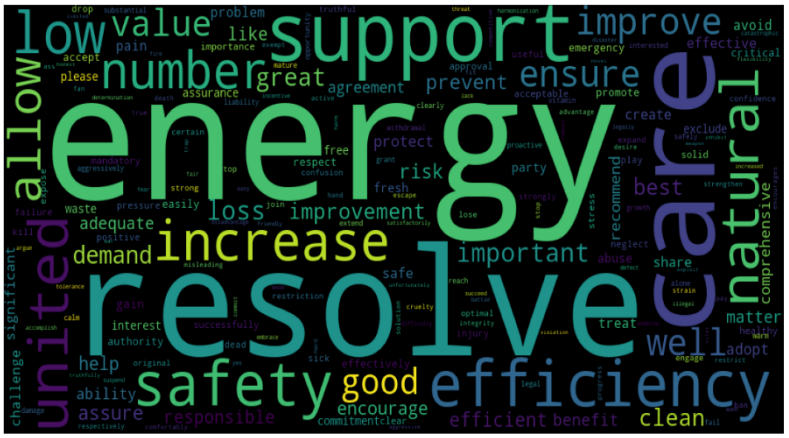


Figure 1.1: Distribution of sentiment range with accompanied word count for each of the four categories of actors.

NGO Cloud

Lobby Cloud



Corporations Cloud

Government Cloud

Figure 1.2: Word clouds for each of the four categories of actors, displaying the most frequently sentimented words of the output.

Appendix A

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.tag import pos_tag
from nltk.stem.wordnet import WordNetLemmatizer
import nltk
import re
import string
import pandas as pd
import matplotlib.pyplot as plt
import warnings
from wordcloud import WordCloud
from collections import Counter

if __name__ == '__main__':
    document = open("All.txt", encoding="utf8").read()
    nltk.download('vader_lexicon')
    analyzer = SentimentIntensityAnalyzer()
    warnings.filterwarnings("ignore")
    stop_words = list(set(stopwords.words('english')))
    tokenized = word_tokenize(document) #tokenized words
    tokenized = [w.lower() for w in tokenized] #make everything lower case
    table = str.maketrans('', '', string.punctuation)
    stripped = [w.translate(table) for w in tokenized] #Strip the words of
punctuation
    words = [word for word in stripped if word.isalpha()]
    words = [w for w in words if not w in stop_words] #Take out stop words

    def lemmatize_sentence(tokens): #Links words into similar word meanings (as in it
removes plurals and groups them together)
        lemmatizer = WordNetLemmatizer()
        lemmatized_words = []
        for word, tag in pos_tag(tokens):
            if tag.startswith('NN'):
                pos = 'n'
            elif tag.startswith('VB'):
                pos = 'v'
            else:
                pos = 'a'
            lemmatized_words.append(lemmatizer.lemmatize(word, pos))
        return lemmatized_words

    lemmatized_words = lemmatize_sentence(words)

    def remove_noise(lemmatized_words, stop_words=()): #Cleans the data, removes all
useless characters, tags with POS and lemmatizes

        cleaned_tokens = []

        for token, tag in pos_tag(lemmatized_words):
            token = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[對]|[$-_@.&+#]|[!*\(\),]|'
```

```
\
                            '(?:%[0-9a-fA-F][0-9a-fA-F]))+', '', token)
            token = re.sub("(@[A-Za-z0-9_]+)", "", token)

            token = re.sub("對", "", token)

            if tag.startswith("NN"):
                pos = 'n'
            elif tag.startswith('VB'):
                pos = 'v'
            else:
                pos = 'a'

            lemmatizer = WordNetLemmatizer()
            token = lemmatizer.lemmatize(token, pos)

            if len(token) > 0 and token not in string.punctuation and token.lower()
not in stop_words:
                cleaned_tokens.append(token.lower())
        return cleaned_tokens

    cleaned_words = remove_noise(lemmatized_words)

    documentSentiments = 0.0 #Sets initial sentiment to 0
    for sentences in cleaned_words: #Uses Vader polarity score, excludes values > or
< than 0 and outputs list.
        vs = analyzer.polarity_scores(sentences)
        if vs['compound'] > 0 or vs['compound'] < 0: #This information was input into
an excel file
            output = "{:-<0} {}".format(sentences, str(vs["compound"]))
            documentSentiments += vs["compound"]
            #print(output)
    print("-----------------------------------------------------")

Excel_file = pd.read_excel('Government.xlsx') # Reads output from vs analyzer from an
excel file and puts in DF
sentiment_anal = pd.DataFrame(Excel_file)
#print(sentiment_anal)
print(sentiment_anal.describe()) #Gives us information about the distribution as
shown in Table 2.1

fig, axes = plt.subplots(sharex=True, sharey=True)  #This section specifies graph
information and saves the picture as  a png file
sentiment_anal.hist(ax=axes)
fig.text(0.5, 0.04, 'Sentiment Range', ha='center')
fig.text(0.04, 0.5, 'Word Count', va='center', rotation='vertical')
plt.title("NGO Histogram")
#plt.show()
plt.savefig('NGOHist.png', bbox_inches='tight')
plt.close()

word_cloud_dict = Counter(sentiment_anal['Words']) #Word cloud draws from the word
section of DF and generates a cloud
wordcloud = WordCloud(width=1000,
height=500).generate_from_frequencies(word_cloud_dict)
```

```
plt.figure(figsize=(15, 8)) #Cloud specifications and saved as a png file
plt.imshow(wordcloud)
plt.axis("off")
plt.suptitle("Government Cloud", fontsize=30)
#plt.show()
plt.savefig('GovCloud.png', bbox_inches='tight')
plt.close()


fd = nltk.FreqDist(sentiment_anal['Words']) #Distribution of words as shown in Table
2.2

print(fd.most_common(50))
```