Natural Language Processing of a Book

Team Name: ASH
Team Members: Ayush Gupta(17ucs042), Harshit Garg(17ucs064), Shubhi Rustagi(17ucs158)

This is part of NLP-Projects Phase-1. In this project, we have used the following libraries and toolkits:

- **nltk** - A leading platform in Python designed to provide easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries. These libraries are used for classification, tokenisation, lemmatisation etc.
- **string** - This module consists of basic operations that can be performed on string.
- **WordNetLemmatizer** - It is a package for lemmatization. Lemmatization is the process of grouping together the words having similar meaning to a particular word.
- **stopwords** - Set of commonly used words that are mostly ignored in text processing.
- **matplotlib** - Library used for data visualisation. In other words, it is used for plotting the relationship among different components.
- **re** - Library used for operations on regular expressions.

To begin, we import all the necssary libraries.

In [2]:
```
import nltk
import string
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import re
from nltk.corpus import brown
nltk.download('brown')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package brown to
[nltk_data]     C:\Users\Shubhi\AppData\Roaming\nltk_data...
[nltk_data]   Package brown is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Shubhi\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Shubhi\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Shubhi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Shubhi\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[2]: True

Now, open the book and read it in a variable *file*. Further, we read the contents of the book in another variable *T*. We declare a variable *test* equal to *T* in case we require the original text later in the process as *T* will undergo text processing.

In [3]:
```
#Reading Book
file = open("Sandy_Hook_NLP.txt",encoding="utf8")

# variable T
T = file.read()

# variable test
test=T

# Printing the contents of the book stored in T
print(T)
```

```
Project Gutenberg's Off Sandy Hook and other stories, by Richard Dehan

This eBook is for the use of anyone anywhere in the United States and most
other parts of the world at no cost and with almost no restrictions
whatsoever.  You may copy it, give it away or re-use it under the terms of
the Project Gutenberg License included with this eBook or online at
www.gutenberg.org.  If you are not located in the United States, you'll have
to check the laws of the country where you are located before using this ebook.

Title: Off Sandy Hook and other stories

Author: Richard Dehan

Release Date: October 8, 2019 [EBook #60452]

Language: English

Character set encoding: UTF-8

*** START OF THIS PROJECT GUTENBERG EBOOK OFF SANDY HOOK AND OTHER STORIES ***
```

Further, punctuation marks are removed using ***maketrans()*** function and ***translate()*** function. First, *maketrans()* maps the string.punctuation to None and creates a mapping table. Then, using *translate()*, all the punctuations in *T* are replaced by None according to the mapping table.

In [4]:
```
translator = str.maketrans("","",string.punctuation)
T=T.translate(translator)
print(T)
```

```
Project Gutenbergs Off Sandy Hook and other stories by Richard Dehan

This eBook is for the use of anyone anywhere in the United States and most
other parts of the world at no cost and with almost no restrictions
whatsoever  You may copy it give it away or reuse it under the terms of
the Project Gutenberg License included with this eBook or online at
wwwgutenbergorg  If you are not located in the United States youll have
to check the laws of the country where you are located before using this ebook

Title Off Sandy Hook and other stories

Author Richard Dehan

Release Date October 8 2019 EBook 60452

Language English

Character set encoding UTF8
```

Removal of the acknowledgement section is done by using the substitution function and regular expressions.

In [5]:
```
# Removing acknowledgement
T = re.sub("Project[\s\S]*CONTENTS","",T)
print(T)
```

```
                                  PAGE
                   OFF SANDY HOOK                   1

                   GEMINI                          15

                   A DISH OF MACARONI              31

                   “FREDDY  CIE”               44

                   UNDER THE ELECTRICS             60

                   “VALCOURT’S GRIN”               68

                   THE EVOLUTION OF THE FAIREST    81

                   THE REVOLT OF RUSTLETON         95
```

Similarly as above, we remove the chapter names from the index of the book.

In [6]:
```
# Removing Chapter Names
T=re.sub("[A-Z]{2,}","",T)
print(T)
```

```
                                  1

                                       15

                   A                 31

                   “  ”               44

                                 60

                   “’S ”               68

                          81

                              95
```

Now, we remove the Chapter Numbers using the substitution function and regular expression.

In [7]:
```
# Removing Chapter Numbers
T=re.sub("[0-9]+","",T)
print(T)
```

```
                   A

                   “  ”



                   “’S ”
```

Conversion of the text to lower case for the ease of analysis.

In [8]:
```
# Converting text to lowercase
T = T.lower()
print(T)
```

```
a

“  ”


“’s ”
```

In [9]:
```
# splitting the lines and joining them into one.
T = " ".join(T.split())
wordcloudT=T
print(T)
```

```
 a “ ” “’s ” a ’s a a a ’s ’s ’s a “– –” on board the rampatina liner eleven days and a half out from liverpool the usual ter
rific sensation created by the appearance of the pilotyacht prevailed necks were craned and toes were trodden on as the steam
er slackened speed and a line dexterously thrown by a bluejerseyed deckhand was caught by somebody aboard the yacht the pilot
not insensible to the fact of his being a personage of note carefully divested his bearded countenance of all expression as h
e saluted the captain and taking from the decksteward’s obsequiously proffered salver a glass containing fourfingers of neat
bourbon whisky concealed its contents about his person without perceptible emotion and went up with the first officer upon th
e upper bridge as the relieved skipper plunged below the telegraphs clicked their message—the leviathan hulk of the liner qui
vered and began to forge slowly ahead and an intelligentlooking thinlipped badlyshaved young man in a bowler tweeds and strip
ed necktie introduced himself to the second officer as an emissary of the press “mr cyrus k pillson new york yeller pleased t
o know you sir” said the second officer “step into the smokeroom this way barsteward a brandy cocktail for me and you sir ord
er whatever you are most in the habit of hoisting whisky straight now sir happy to afford you what information i can” “i pres
ume” observed the young gentleman of the press settling himself on the springy morocco cushions and accepting the second offi
cer’s polite offer of a green havana of the strongest kind “that you have had a smooth passage considerin’ the time of year”
“smooth” the second officer carefully reversed in his reply the pressman’s remark “well yes the time of year considered a smo
oth passage i take it we have had” “no fogs” interrogated the young gentleman clicking the elastic band of a notebook which p
rojected from his breastpocket “fogs no” said the second officer “you didn’t chance” pursued the young gentleman of the press
taking his short drink from the steward’s salver and throwing it contemptuously down his throat “to fall in with a berg off t
he bank did you” “not a smell of one” replied the second officer with decision “ran into a derelict hencoop perhaps” persiste
d the young gentleman concealing the worn sole of a wearied boot from the searching glare of the electric light by tucking it
```

Tokenisation is splitting a string into words, thus, forming a list of words known as tokens. For tokenisation, we use a library from *nltk* known as *tokenise*. The library consists of function *word_tokenize()* which takes the string, here *T* as the parameter and returns the list as shown in the output.

In [10]:
```
# Tokenisation
from nltk.tokenize import word_tokenize

T = word_tokenize(T)
print(T)
```

```
['\ufeff', 'a', '“', '”', '“', '’', 's', '”', 'a', '’', 's', 'a', 'a', 'a', '’', 's', '’', 's', '’', 's', 'a', '“', '–', '–',
'”', 'on', 'board', 'the', 'rampatina', 'liner', 'eleven', 'days', 'and', 'a', 'half', 'out', 'from', 'liverpool', 'the', 'us
ual', 'terrific', 'sensation', 'created', 'by', 'the', 'appearance', 'of', 'the', 'pilotyacht', 'prevailed', 'necks', 'were',
'craned', 'and', 'toes', 'were', 'trodden', 'on', 'as', 'the', 'steamer', 'slackened', 'speed', 'and', 'a', 'line', 'dexterou
sly', 'thrown', 'by', 'a', 'bluejerseyed', 'deckhand', 'was', 'caught', 'by', 'somebody', 'aboard', 'the', 'yacht', 'the', 'p
ilot', 'not', 'insensible', 'to', 'the', 'fact', 'of', 'his', 'being', 'a', 'personage', 'of', 'note', 'carefully', 'diveste
d', 'his', 'bearded', 'countenance', 'of', 'all', 'expression', 'as', 'he', 'saluted', 'the', 'captain', 'and', 'taking', 'fr
om', 'the', 'decksteward', '’', 's', 'obsequiously', 'proffered', 'salver', 'a', 'glass', 'containing', 'fourfingers', 'of',
'neat', 'bourbon', 'whisky', 'concealed', 'its', 'contents', 'about', 'his', 'person', 'without', 'perceptible', 'emotion',
'and', 'went', 'up', 'with', 'the', 'first', 'officer', 'upon', 'the', 'upper', 'bridge', 'as', 'the', 'relieved', 'skipper',
'plunged', 'below', 'the', 'telegraphs', 'clicked', 'their', 'message—the', 'leviathan', 'hulk', 'of', 'the', 'liner', 'quive
red', 'and', 'began', 'to', 'forge', 'slowly', 'ahead', 'and', 'an', 'intelligentlooking', 'thinlipped', 'badlyshaved', 'youn
g', 'man', 'in', 'a', 'bowler', 'tweeds', 'and', 'striped', 'necktie', 'introduced', 'himself', 'to', 'the', 'second', 'offic
er', 'as', 'an', 'emissary', 'of', 'the', 'press', '“', 'mr', 'cyrus', 'k', 'pillson', 'new', 'york', 'yeller', 'pleased', 't
o', 'know', 'you', 'sir', '”', 'said', 'the', 'second', 'officer', '“', 'step', 'into', 'the', 'smokeroom', 'this', 'way', 'b
arsteward', 'a', 'brandy', 'cocktail', 'for', 'me', 'and', 'you', 'sir', 'order', 'whatever', 'you', 'are', 'most', 'in', 'th
e', 'habit', 'of', 'hoisting', 'whisky', 'straight', 'now', 'sir', 'happy', 'to', 'afford', 'you', 'what', 'information',
'i', 'can', '”', '“', 'i', 'presume', '”', 'observed', 'the', 'young', 'gentleman', 'of', 'the', 'press', 'settling', 'himsel
f', 'on', 'the', 'springy', 'morocco', 'cushions', 'and', 'accepting', 'the', 'second', 'officer', '’', 's', 'polite', 'offe
```

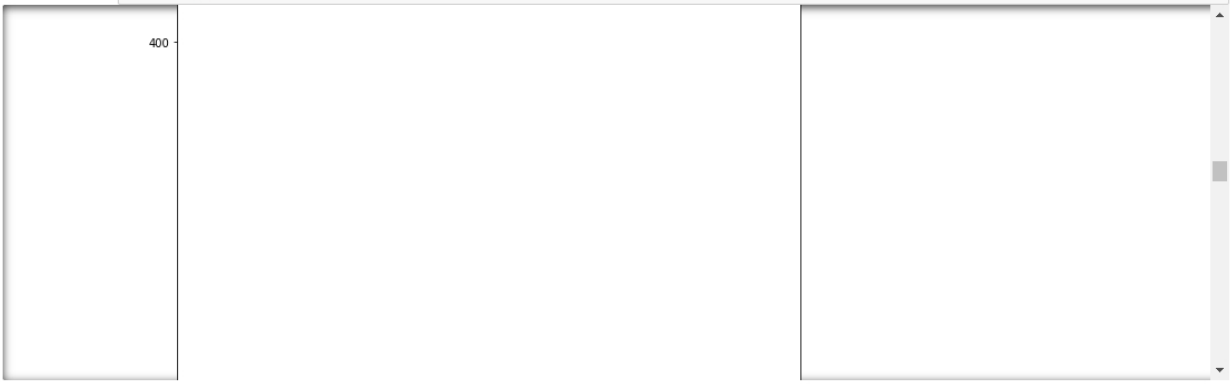
Lemmatization is grouping together of the different forms of a word having similar meaning.

In [11]:
```
# Lemmatisation
lemmatizer = WordNetLemmatizer()

# forming a set of stopwords from english language
stop_words = set(stopwords.words('english'))

# creating an empty list
lemmatized_T=[]

# traversing all the words
for word in T:
    # if the word has a length greater than or equal to 2 and is not a stopword
    if len(word) >= 2 and word not in stop_words:
        # then we append the word into the list lemmatized_T after performing lemmatization
        # using the lemmatize() function
        lemmatized_T.append(lemmatizer.lemmatize(word))

# printing the list of lemmatized words
print(lemmatized_T)
```

```
[ board , rampatina , liner , eleven , day , half , liverpool , usual , terrific , sensation , created , appearanc
e', 'pilotyacht', 'prevailed', 'neck', 'craned', 'toe', 'trodden', 'steamer', 'slackened', 'speed', 'line', 'dexterously', 't
hrown', 'bluejerseyed', 'deckhand', 'caught', 'somebody', 'aboard', 'yacht', 'pilot', 'insensible', 'fact', 'personage', 'not
e', 'carefully', 'divested', 'bearded', 'countenance', 'expression', 'saluted', 'captain', 'taking', 'decksteward', 'obsequio
usly', 'proffered', 'salver', 'glass', 'containing', 'fourfingers', 'neat', 'bourbon', 'whisky', 'concealed', 'content', 'per
son', 'without', 'perceptible', 'emotion', 'went', 'first', 'officer', 'upon', 'upper', 'bridge', 'relieved', 'skipper', 'plu
nged', 'telegraph', 'clicked', 'message—the', 'leviathan', 'hulk', 'liner', 'quivered', 'began', 'forge', 'slowly', 'ahead',
'intelligentlooking', 'thinlipped', 'badlyshaved', 'young', 'man', 'bowler', 'tweed', 'striped', 'necktie', 'introduced', 'se
cond', 'officer', 'emissary', 'press', 'mr', 'cyrus', 'pillson', 'new', 'york', 'yeller', 'pleased', 'know', 'sir', 'said',
'second', 'officer', 'step', 'smokeroom', 'way', 'barsteward', 'brandy', 'cocktail', 'sir', 'order', 'whatever', 'habit', 'ho
isting', 'whisky', 'straight', 'sir', 'happy', 'afford', 'information', 'presume', 'observed', 'young', 'gentleman', 'press',
'settling', 'springy', 'morocco', 'cushion', 'accepting', 'second', 'officer', 'polite', 'offer', 'green', 'havana', 'stronge
st', 'kind', 'smooth', 'passage', 'considerin', 'time', 'year', 'smooth', 'second', 'officer', 'carefully', 'reversed', 'repl
y', 'pressman', 'remark', 'well', 'yes', 'time', 'year', 'considered', 'smooth', 'passage', 'take', 'fog', 'interrogated', 'y
oung', 'gentleman', 'clicking', 'elastic', 'band', 'notebook', 'projected', 'breastpocket', 'fog', 'said', 'second', 'office
r', 'chance', 'pursued', 'young', 'gentleman', 'press', 'taking', 'short', 'drink', 'steward', 'salver', 'throwing', 'contemp
tuously', 'throat', 'fall', 'berg', 'bank', 'smell', 'one', 'replied', 'second', 'officer', 'decision', 'ran', 'derelict', 'h
encoop', 'perhaps', 'persisted', 'young', 'gentleman', 'concealing', 'worn', 'sole', 'wearied', 'boot', 'searching', 'glare',
'electric', 'light', 'tucking', 'underneath', 'old', 'lady', 'bonnetbox', 'rubber', 'doll', 'woman', 'baby', 'lost', 'overboa
```

Analysis of the frequency of various tokens

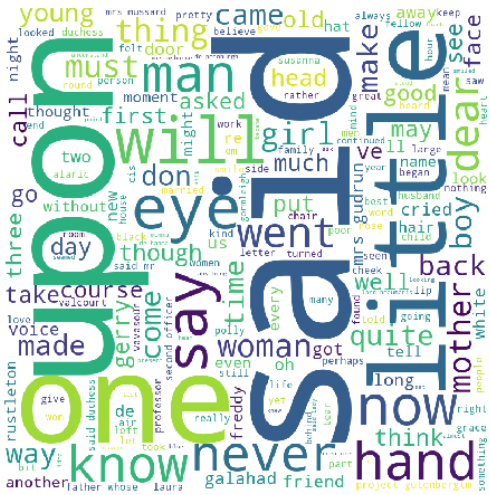
In [13]:
```
# Evaluating frequency distribution of tokens
freq_dist = nltk.FreqDist(lemmatized_T)

# creating a list of the tokens
keys = list(freq_dist.keys())

# creating a list of the frequency of the various tokens
values = list(freq_dist.values())

# plotting a scatter diagram of the frequency distribution
plt.figure(figsize=(10,100))
plt.scatter(keys,values)
plt.xlabel("Tokens")
plt.ylabel("Frequency")
plt.title("Frequency Distribution of tokens")
plt.show()
```



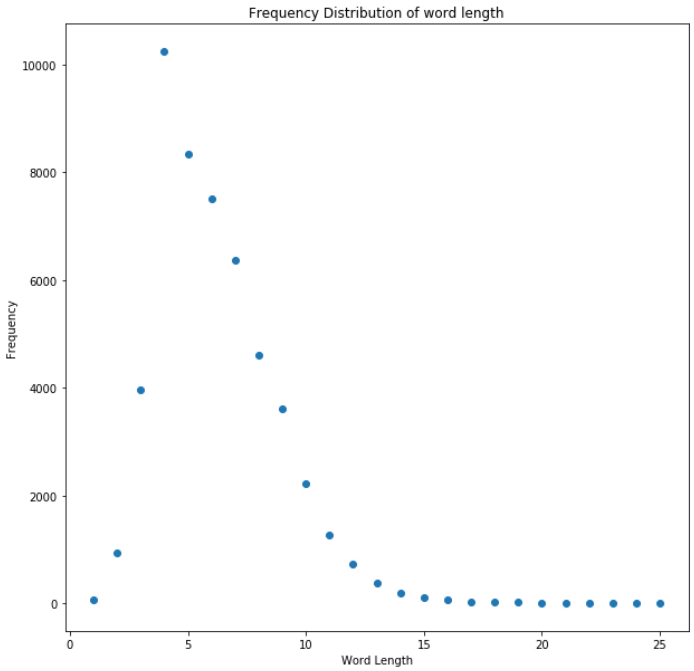
WordCloud is a representation of the textual data according to their frequency. It basically emphasizes the keywords used in the text.

In [14]:
```
# WordCloud without removing Stopwords
wordcloud = WordCloud(width = 800, height=800,
                      background_color='white',
                      min_font_size=10).generate(wordcloudT)

plt.figure(figsize=(8,8),facecolor=None)
plt.imshow(wordcloud)
plt.axis("off")

plt.show()
```

In [15]:
```
# WordCloud after removing Stopwords
wordcloud = WordCloud(width = 800, height=800,
                      background_color='white',
                       stopwords = set(STOPWORDS),
                      min_font_size=10).generate(wordcloudT)

plt.figure(figsize=(8,8),facecolor=None)
plt.imshow(wordcloud)
plt.axis("off")

plt.show()
```

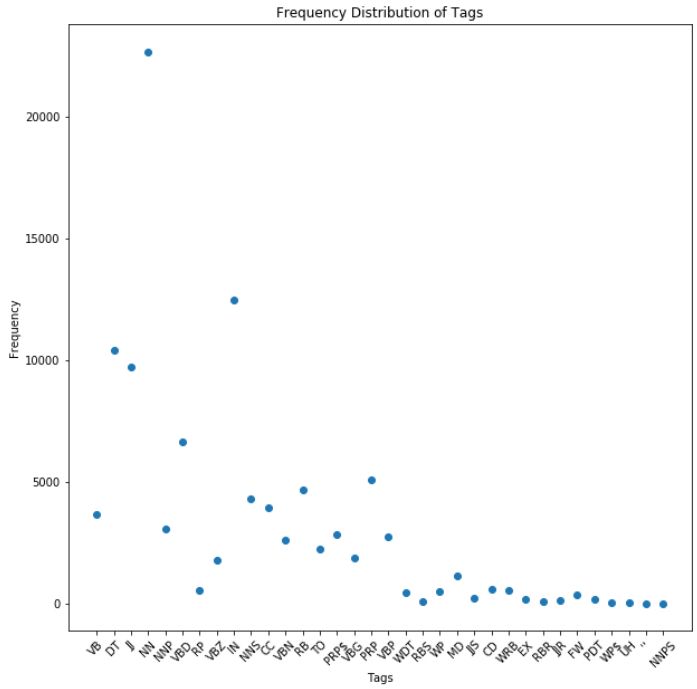


In [16]:
```
#Frequency distribution of Word Length
len_list={}

for i in range(len(lemmatized_T)):

    if len(lemmatized_T[i]) not in len_list:
        len_list[len(lemmatized_T[i])] = 1

    else:
        len_list[len(lemmatized_T[i])] += 1

keys = list(len_list.keys())
values = list(len_list.values())

plt.figure(figsize=(10,10))
plt.scatter(keys,values)
plt.xlabel("Word Length")
plt.ylabel("Frequency")
plt.title("Frequency Distribution of word length")
plt.show()
```



In [17]:
```
#POS Tagging
PosTags = nltk.pos_tag(T)

tag_list={}

for i in range(len(PosTags)):

    if PosTags[i][1] not in tag_list:
        tag_list[PosTags[i][1]] = 1

    else:
        tag_list[PosTags[i][1]] += 1

keys = list(tag_list.keys())
values = list(tag_list.values())

# Plotting frequency distribution of the tags
plt.figure(figsize=(10,10))
plt.scatter(keys,values)
plt.xlabel("Tags")
```

```
plt.xlabel("Tags")
plt.xticks(rotation = 45)
plt.ylabel("Frequency")
plt.title("Frequency Distribution of Tags")
plt.show()
```



In [18]:
```
# using brown corpus for POS tagging
brown_news_words = brown.tagged_words(categories='news', tagset='universal')
brown_news_words
```

Out[18]:
```
[('The', 'DET'), ('Fulton', 'NOUN'), ...]
```

In [19]:
```
# Frequency distribution of POS tags
fdistw = nltk.FreqDist([w for (w, t) in brown_news_words])
fdistw
```

Out[19]:
```
FreqDist({'the': 5580, ',': 5188, '.': 4030, 'of': 2849, 'and': 2146, 'to': 2116, 'a': 1993, 'in': 1893, 'for': 943, 'The': 806, ...})
```

In [21]:
```
# Representation of frequrncy distribution of POS tags using brown corpus
# creating a list of the tokens
keys = list(fdistw.keys())

# creating a list of the frequency of the various tokens
values = list(fdistw.values())

# plotting a scatter diagram of the frequency distribution
plt.figure(figsize=(10,100))
plt.scatter(keys,values)
plt.xlabel("Tags")
plt.ylabel("Frequency")
plt.title("Frequency Distribution of tags")
plt.show()
```