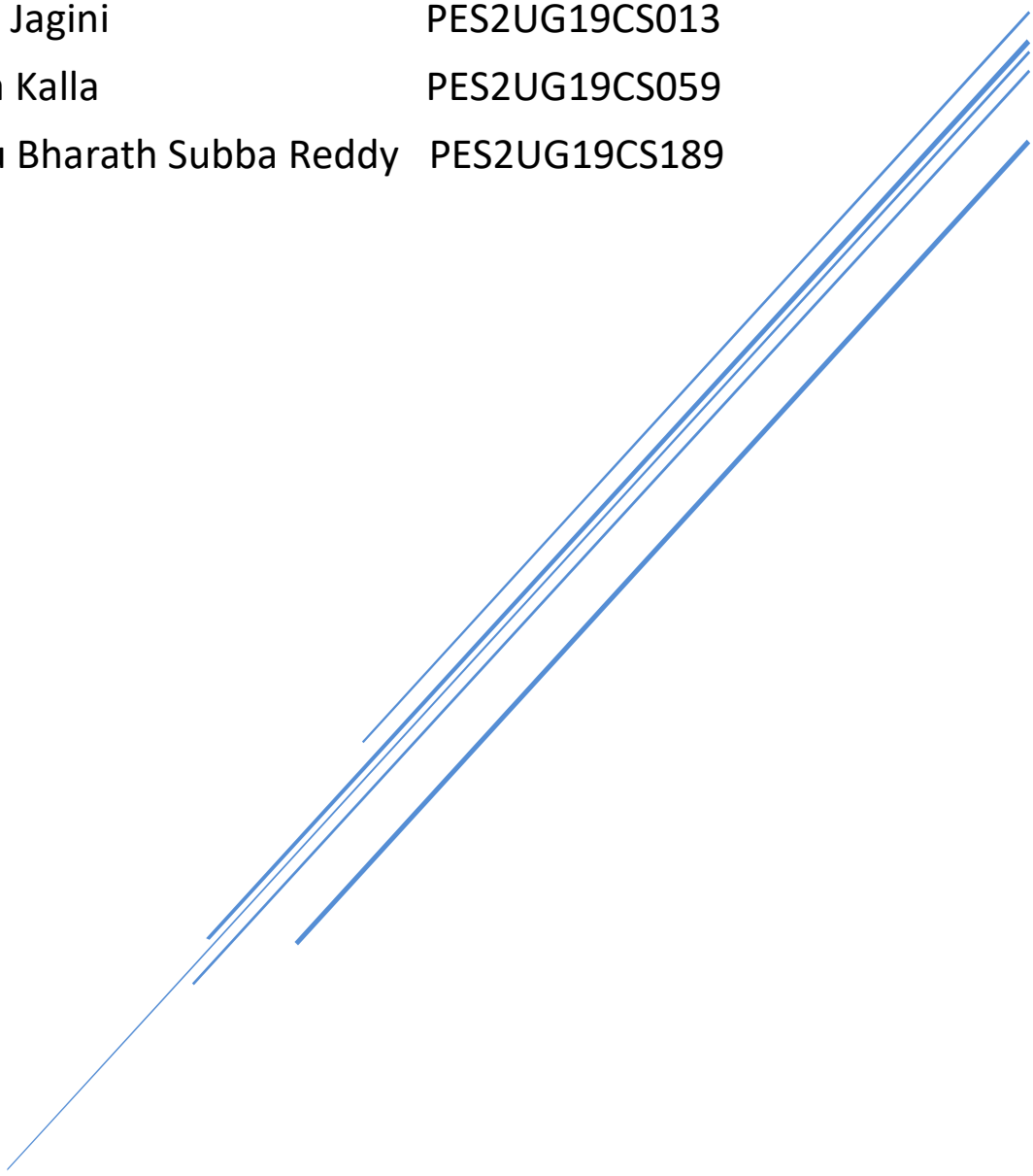
AIWIR ASSIGNMENT

DISASTER TWEET ANALYSIS

Batch-5

Group members

Achyut Jagini	PES2UG19CS013
Aparna Kalla	PES2UG19CS059
Koduru Bharath Subba Reddy	PES2UG19CS189



In [10]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Core packages for text processing.
#import wordcloud
import string
import re

# Libraries for text preprocessing.

import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
from nltk.corpus import stopwords, wordnet
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation, NMF
from sklearn.metrics import f1_score, accuracy_score

# Some packages for word clouds and NER.

#from wordcloud import WordCloud, STOPWORDS
from collections import Counter, defaultdict
from PIL import Image
#import spacy
!pip install https://github.com/explosion/spacy-models/releases/download/en_core_
import en_core_web_sm

# Core packages for general use throughout the notebook.

import random
import warnings
import time
import datetime
# For customizing our plots.

from matplotlib.ticker import MaxNLocator
import matplotlib.gridspec as gridspec
import matplotlib.patches as mpatches

# Loading pytorch packages.

import torch
from transformers import BertTokenizer, BertForSequenceClassification, AdamW, Ber
from torch.utils.data import TensorDataset, random_split, DataLoader, RandomSampl

# Setting some options for general use.
```

```
plt.style.use('fivethirtyeight')
sns.set(font_scale=1.5)
pd.options.display.max_columns = 250
pd.options.display.max_rows = 250
warnings.filterwarnings('ignore')


#Setting seeds for consistent results.
seed_val = 42
random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)
```

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

```
Collecting https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-2.2.5/en_core_web_sm-2.2.5.tar.gz (https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-2.2.5/en_core_web_sm-2.2.5.tar.gz)
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-2.2.5/en_core_web_sm-2.2.5.tar.gz (https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-2.2.5/en_core_web_sm-2.2.5.tar.gz) (12.0 MB)
Requirement already satisfied: spacy>=2.2.2 in d:\conda\lib\site-packages (from en-core-web-sm==2.2.5) (3.2.4)
Requirement already satisfied: thinc<8.1.0,>=8.0.12 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (8.0.15)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (2.0.6)
Requirement already satisfied: pathy>=0.3.5 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (0.6.1)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (4.62.3)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.8 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (3.0.9)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (1.0.2)
Requirement already satisfied: typer<0.5.0,>=0.3.0 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (0.4.1)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (0.7.7)
Requirement already satisfied: packaging>=20.0 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (21.0)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (2.26.0)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in d:\conda\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (2.0.7)
```

```
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.9.0,>=1.7.4 in d:\conda
\lib\site-packages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (1.8.2)
Requirement already satisfied: jinja2 in d:\conda\lib\site-packages (from spacy
>=2.2.2->en-core-web-sm==2.2.5) (2.11.3)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in d:\conda\lib\site-p
ackages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (1.0.6)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in d:\conda\lib\site-pac
kages (from spacy>=2.2.2->en-core-web-sm==2.2.5) (3.3.0)
Requirement already satisfied: click<8.1.0 in d:\conda\lib\site-packages (from
 spacy>=2.2.2->en-core-web-sm==2.2.5) (8.0.3)
Requirement already satisfied: setuptools in d:\conda\lib\site-packages (from s
pacy>=2.2.2->en-core-web-sm==2.2.5) (58.0.4)
Requirement already satisfied: numpy>=1.15.0 in d:\conda\lib\site-packages (fro
m spacy>=2.2.2->en-core-web-sm==2.2.5) (1.20.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.1 in d:\conda\lib\site-package
s (from spacy>=2.2.2->en-core-web-sm==2.2.5) (2.4.2)
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in d:\conda\lib\site-packag
es (from spacy>=2.2.2->en-core-web-sm==2.2.5) (0.9.1)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in d:\conda\lib\site-packa
ges (from spacy>=2.2.2->en-core-web-sm==2.2.5) (3.0.6)
Requirement already satisfied: colorama in d:\conda\lib\site-packages (from cli
ck<8.1.0->spacy>=2.2.2->en-core-web-sm==2.2.5) (0.4.4)
Requirement already satisfied: pyparsing>=2.0.2 in d:\conda\lib\site-packages
 (from packaging>=20.0->spacy>=2.2.2->en-core-web-sm==2.2.5) (3.0.4)
Requirement already satisfied: smart-open<6.0.0,>=5.0.0 in d:\conda\lib\site-pa
ckages (from pathy>=0.3.5->spacy>=2.2.2->en-core-web-sm==2.2.5) (5.2.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in d:\conda\lib\site-
packages (from pydantic!=1.8,!=1.8.1,<1.9.0,>=1.7.4->spacy>=2.2.2->en-core-web-
sm==2.2.5) (3.10.0.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in d:\conda\lib\site-packa
ges (from requests<3.0.0,>=2.13.0->spacy>=2.2.2->en-core-web-sm==2.2.5) (1.26.
7)
Requirement already satisfied: certifi>=2017.4.17 in d:\conda\lib\site-packages
(from requests<3.0.0,>=2.13.0->spacy>=2.2.2->en-core-web-sm==2.2.5) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in d:\conda\lib\site-p
ackages (from requests<3.0.0,>=2.13.0->spacy>=2.2.2->en-core-web-sm==2.2.5) (2.
0.4)
Requirement already satisfied: idna<4,>=2.5 in d:\conda\lib\site-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.2->en-core-web-sm==2.2.5) (3.2)
Requirement already satisfied: MarkupSafe>=0.23 in d:\conda\lib\site-packages
 (from jinja2->spacy>=2.2.2->en-core-web-sm==2.2.5) (1.1.1)

D:\Conda\lib\site-packages\spacy\util.py:841: UserWarning: [W094] Model 'en_cor
e_web_sm' (2.2.5) specifies an under-constrained spaCy version requirement: >=
2.2.2. This can lead to compatibility problems with older versions, or as new s
paCy versions are released, because the model may say it's compatible when it's
not. Consider changing the "spacy_version" in your meta.json to a version rang
e, with a lower and upper pin. For example: >=3.2.4,<3.3.0
  warnings.warn(warn_msg)
```

In [31]:

```
trainv = pd.read_csv('train.csv')
testv = pd.read_csv('test.csv')
```

In [32]:
```
display(trainv.sample(5))
display(testv.sample(5))
```

	id	keyword	location	text	target
735	1065	bleeding	Gages Lake, IL	@beckyfeigin I defs will when it stops bleeding!	1
2152	3089	deaths	the void, U.S.A	@HighQualityBird a reverse situation (lol I do...	1
994	1444	body%20bagging	New York, NY	@bomairinge @EluTranscendent straight body bag...	0
6523	9330	survive	Gotham City	You can't fight fate and you can't survive alo...	0
1188	1710	bridge%20collapse	NaN	Mexico: construction of bridge collapse killsâ...	1

	id	keyword	location	text
2386	7975	razed	Try looking at the map?	@RazeD_ yea in this photo
2351	7858	quarantined	china	Top link: Reddit's new content policy goes int...
2704	9011	stretcher	Seattle, WA	@TheJasonTaylorR *EMS tries to stablize me and...
1152	3799	detonate	Nashville, Music City, USA	@HJudeBoudreaux Start your car with it! Or use...
2512	8376	ruin	NaN	He said he's 'gonna put a ring in my Harvey's ...

In [33]:
```
# Checking observation and feature numbers for train and test data.

print(trainv.shape)
print(testv.shape)
```

```
(7613, 5)
(3263, 4)
```

In [34]:
```
# Some basic helper functions to clean text by removing urls, emojis, html tags c

def remove_URL(text):
    url = re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r'', text)


def remove_emoji(text):
    emoji_pattern = re.compile(
        '['
        u'\U0001F600-\U0001F64F'  # emoticons
        u'\U0001F300-\U0001F5FF'  # symbols & pictographs
        u'\U0001F680-\U0001F6FF'  # transport & map symbols
        u'\U0001F1E0-\U0001F1FF'  # flags (iOS)
        u'\U00002702-\U000027B0'
        u'\U000024C2-\U0001F251'
        ']+',
        flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)


def remove_html(text):
    html = re.compile(r'<.*?>|&([a-z0-9]+|#[0-9]{1,6}|#x[0-9a-f]{1,6});')
    return re.sub(html, '', text)


def remove_punct(text):
    table = str.maketrans('', '', string.punctuation)
    return text.translate(table)

# Applying helper functions

trainv['text_clean'] = trainv['text'].apply(lambda x: remove_URL(x))
trainv['text_clean'] = trainv['text_clean'].apply(lambda x: remove_emoji(x))
trainv['text_clean'] = trainv['text_clean'].apply(lambda x: remove_html(x))
trainv['text_clean'] = trainv['text_clean'].apply(lambda x: remove_punct(x))
```

In [35]:
```
# Tokenizing the tweet base texts.

trainv['tokenized'] = trainv['text_clean'].apply(word_tokenize)

trainv.head()
```

Out[35]:

	id	keyword	location	text	target	text_clean	tokenized
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	Our Deeds are the Reason of this earthquake Ma...	[Our, Deeds, are, the, Reason, of, this, earth...
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	Forest fire near La Ronge Sask Canada	[Forest, fire, near, La, Ronge, Sask, Canada]
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	All residents asked to shelter in place are be...	[All, residents, asked, to, shelter, in, place...
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	13000 people receive wildfires evacuation orde...	[13000, people, receive, wildfires, evacuation...
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	Just got sent this photo from Ruby Alaska as s...	[Just, got, sent, this, photo, from, Ruby, Ala...

In [36]:
```
# Lower casing clean text.

trainv['lower'] = trainv['tokenized'].apply(
    lambda x: [word.lower() for word in x])

trainv.head()
```

Out[36]:

	id	keyword	location	text	target	text_clean	tokenized	lower
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	Our Deeds are the Reason of this earthquake Ma...	[Our, Deeds, are, the, Reason, of, this, earth...	[our, deeds, are, the, reason, of, this, earth...
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	Forest fire near La Ronge Sask Canada	[Forest, fire, near, La, Ronge, Sask, Canada]	[forest, fire, near, la, ronge, sask, canada]
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	All residents asked to shelter in place are be...	[All, residents, asked, to, shelter, in, place...	[all, residents, asked, to, shelter, in, place...
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	13000 people receive wildfires evacuation orde...	[13000, people, receive, wildfires, evacuation...	[13000, people, receive, wildfires, evacuation...
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	Just got sent this photo from Ruby Alaska as s...	[Just, got, sent, this, photo, from, Ruby, Ala...	[just, got, sent, this, photo, from, ruby, ala...

In [37]:
```
# Removing stopwords.

stops = set(stopwords.words('english'))
trainv['stopwords_removed'] = trainv['lower'].apply(lambda x: [word for word in x

trainv.head()
```

Out[37]:

	id	keyword	location	text	target	text_clean	tokenized	lower	stopwords_rem
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	Our Deeds are the Reason of this earthquake Ma...	[Our, Deeds, are, the, Reason, of, this, earth...	[our, deeds, are, the, reason, of, this, earth...	[deeds, rea earthquake, allah, for
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	Forest fire near La Ronge Sask Canada	[Forest, fire, near, La, Ronge, Sask, Canada]	[forest, fire, near, la, ronge, sask, canada]	[forest, fire, nea ronge, sask, car
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	All residents asked to shelter in place are be...	[All, residents, asked, to, shelter, in, place...	[all, residents, asked, to, shelter, in, place...	[residents, as shelter, p notifiec
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	13000 people receive wildfires evacuation orde...	[13000, people, receive, wildfires, evacuation...	[13000, people, receive, wildfires, evacuation...	[13000, pe receive, wild evacuat
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	Just got sent this photo from Ruby Alaska as s...	[Just, got, sent, this, photo, from, Ruby, Ala...	[just, got, sent, this, photo, from, ruby, ala...	[got, sent, p ruby, alaska, sn wi

In [38]:
```
# Applying part of speech tags.

trainv['pos_tags'] = trainv['stopwords_removed'].apply(nltk.tag.pos_tag)

trainv.head()
```

Out[38]:

	id	keyword	location	text	target	text_clean	tokenized	lower	stopwords_rem
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	Our Deeds are the Reason of this earthquake Ma...	[Our, Deeds, are, the, Reason, of, this, earth...	[our, deeds, are, the, reason, of, this, earth...	[deeds, rea earthquake, allah, for
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	Forest fire near La Ronge Sask Canada	[Forest, fire, near, La, Ronge, Sask, Canada]	[forest, fire, near, la, ronge, sask, canada]	[forest, fire, nea ronge, sask, car
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	All residents asked to shelter in place are be...	[All, residents, asked, to, shelter, in, place...	[all, residents, asked, to, shelter, in, place...	[residents, as shelter, p notifiec
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	13000 people receive wildfires evacuation orde...	[13000, people, receive, wildfires, evacuation...	[13000, people, receive, wildfires, evacuation...	[13000, pe receive, wild evacuat
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	Just got sent this photo from Ruby Alaska as s...	[Just, got, sent, this, photo, from, Ruby, Ala...	[just, got, sent, this, photo, from, ruby, ala...	[got, sent, p ruby, alaska, sn wi

In [40]:
```
nltk.download('omw-1.4')
# Converting part of speeches to wordnet format.

def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN


trainv['wordnet_pos'] = trainv['pos_tags'].apply(
    lambda x: [(word, get_wordnet_pos(pos_tag)) for (word, pos_tag) in x])

trainv.head()
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\achyu\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[40]:

	id	keyword	location	text	target	text_clean	tokenized	lower	stopwords_rem
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	Our Deeds are the Reason of this earthquake Ma...	[Our, Deeds, are, the, Reason, of, this, earth...	[our, deeds, are, the, reason, of, this, earth...	[deeds, rea earthquake, allah, for
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	Forest fire near La Ronge Sask Canada	[Forest, fire, near, La, Ronge, Sask, Canada]	[forest, fire, near, la, ronge, sask, canada]	[forest, fire, nea ronge, sask, car
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	All residents asked to shelter in place are be...	[All, residents, asked, to, shelter, in, place...	[all, residents, asked, to, shelter, in, place...	[residents, as shelter, p notifiec
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	13000 people receive wildfires evacuation orde...	[13000, people, receive, wildfires, evacuation...	[13000, people, receive, wildfires, evacuation...	[13000, pe receive, wild evacuat
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	Just got sent this photo from Ruby Alaska as s...	[Just, got, sent, this, photo, from, Ruby, Ala...	[just, got, sent, this, photo, from, ruby, ala...	[got, sent, p ruby, alaska, sn wi

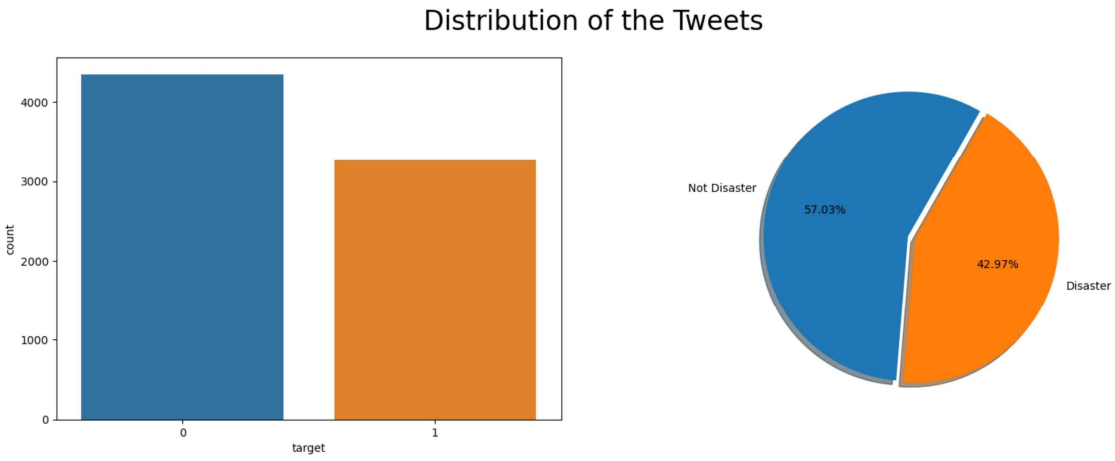
In [41]:
```
# Applying word lemmatizer.

wnl = WordNetLemmatizer()

trainv['lemmatized'] = trainv['wordnet_pos'].apply(
    lambda x: [wnl.lemmatize(word, tag) for word, tag in x])

trainv['lemmatized'] = trainv['lemmatized'].apply(
    lambda x: [word for word in x if word not in stops])

trainv['lemma_str'] = [' '.join(map(str, l)) for l in trainv['lemmatized']]

trainv.head()
```

Out[41]:

	id	keyword	location	text	target	text_clean	tokenized	lower	stopwords_rem
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	Our Deeds are the Reason of this earthquake Ma...	[Our, Deeds, are, the, Reason, of, this, earth...	[our, deeds, are, the, reason, of, this, earth...	[deeds, rea earthquake, allah, for
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	Forest fire near La Ronge Sask Canada	[Forest, fire, near, La, Ronge, Sask, Canada]	[forest, fire, near, la, ronge, sask, canada]	[forest, fire, nea ronge, sask, car
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	All residents asked to shelter in place are be...	[All, residents, asked, to, shelter, in, place...	[all, residents, asked, to, shelter, in, place...	[residents, as shelter, p notifiec
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	13000 people receive wildfires evacuation orde...	[13000, people, receive, wildfires, evacuation...	[13000, people, receive, wildfires, evacuation...	[13000, pe receive, wild evacuat
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	Just got sent this photo from Ruby Alaska as s...	[Just, got, sent, this, photo, from, Ruby, Ala...	[just, got, sent, this, photo, from, ruby, ala...	[got, sent, p ruby, alaska, sn wi

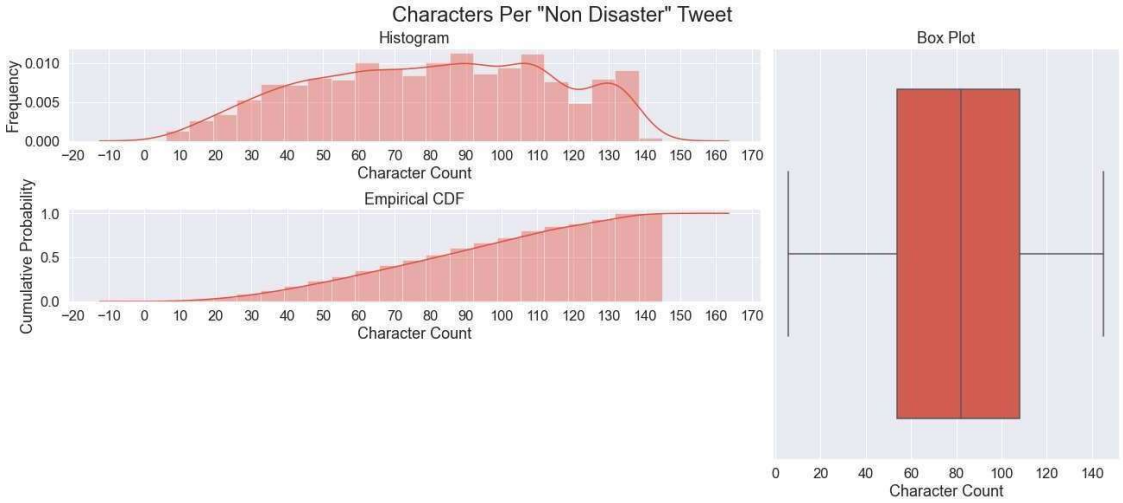
In [42]:
```
# Displaying target distribution.

fig, axes = plt.subplots(ncols=2, nrows=1, figsize=(18, 6), dpi=100)
sns.countplot(trainv['target'], ax=axes[0])
axes[1].pie(trainv['target'].value_counts(),
            labels=['Not Disaster', 'Disaster'],
            autopct='%1.2f%%',
            shadow=True,
            explode=(0.05, 0),
            startangle=60)
fig.suptitle('Distribution of the Tweets', fontsize=24)
plt.show()
```

```
D:\Conda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positi
onal argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

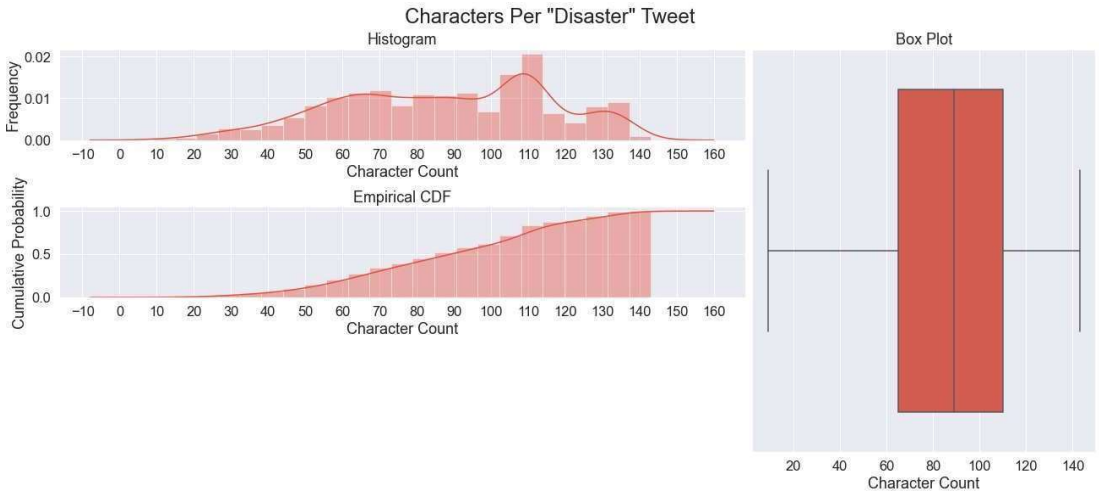


In [19]:
```
# Creating a new feature for the visualization.

trainv['Character Count'] = trainv['text_clean'].apply(lambda x: len(str(x)))


def plot_dist3(df, feature, title):
    # Creating a customized chart. and giving in figsize and everything.
    fig = plt.figure(constrained_layout=True, figsize=(18, 8))
    # Creating a grid of 3 cols and 3 rows.
    grid = gridspec.GridSpec(ncols=3, nrows=3, figure=fig)

    # Customizing the histogram grid.
    ax1 = fig.add_subplot(grid[0, :2])
    # Set the title.
    ax1.set_title('Histogram')
    # plot the histogram.
    sns.distplot(df.loc[:, feature],
                 hist=True,
                 kde=True,
                 ax=ax1,
                 color='#e74c3c')
    ax1.set(ylabel='Frequency')
    ax1.xaxis.set_major_locator(MaxNLocator(nbins=20))

    # Customizing the ecdf_plot.
    ax2 = fig.add_subplot(grid[1, :2])
    # Set the title.
    ax2.set_title('Empirical CDF')
    # Plotting the ecdf_Plot.
    sns.distplot(df.loc[:, feature],
                 ax=ax2,
                 kde_kws={'cumulative': True},
                 hist_kws={'cumulative': True},
                 color='#e74c3c')
    ax2.xaxis.set_major_locator(MaxNLocator(nbins=20))
    ax2.set(ylabel='Cumulative Probability')
    # Customizing the Box Plot.
    ax3 = fig.add_subplot(grid[:, 2])
    # Set title.
    ax3.set_title('Box Plot')
    # Plotting the box plot.
    sns.boxplot(x=feature, data=df, orient='v', ax=ax3, color='#e74c3c')
    ax3.yaxis.set_major_locator(MaxNLocator(nbins=25))

    plt.suptitle(f'{title}', fontsize=24)
```

In [20]:
```
plot_dist3(trainv[trainv['target'] == 0], 'Character Count','Characters Per "Non
```

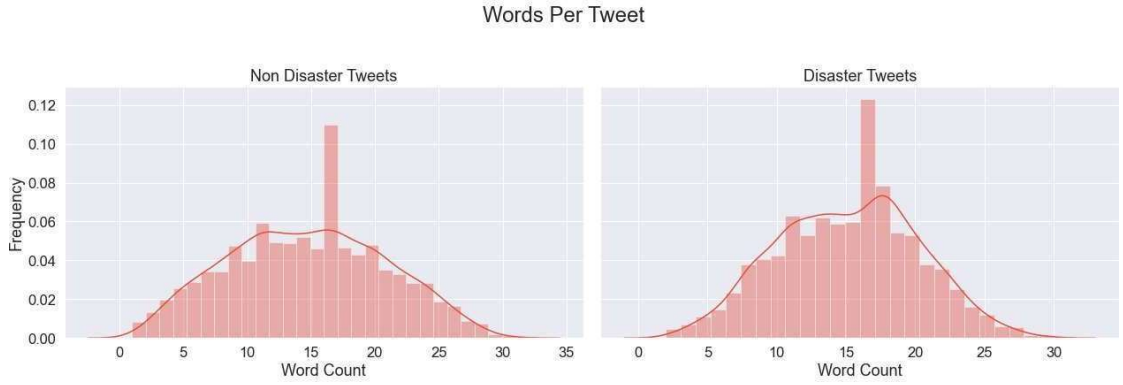


In [21]:
```
plot_dist3(trainv[trainv['target'] == 1], 'Character Count','Characters Per "Disa
```



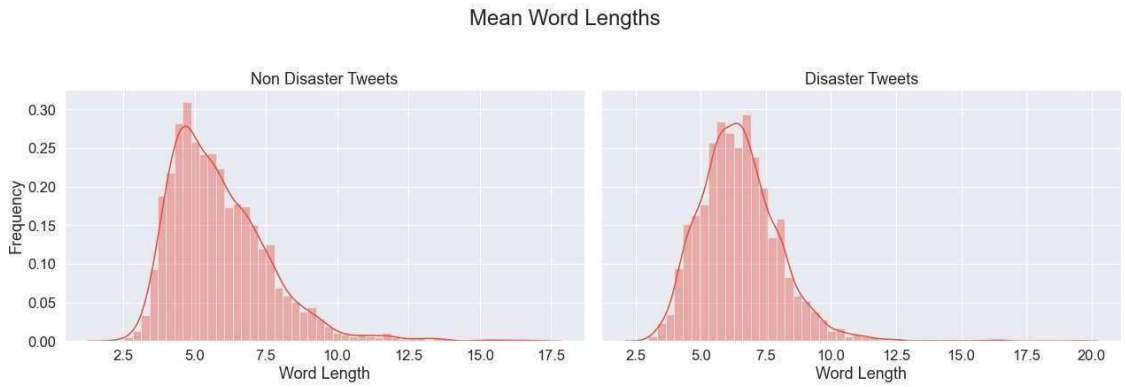
In [22]:
```
def plot_word_number_histogram(textno, textye):

    """A function for comparing word counts"""

    fig, axes = plt.subplots(ncols=2, nrows=1, figsize=(18, 6), sharey=True)
    sns.distplot(textno.str.split().map(lambda x: len(x)), ax=axes[0], color='#e
    sns.distplot(textye.str.split().map(lambda x: len(x)), ax=axes[1], color='#e

    axes[0].set_xlabel('Word Count')
    axes[0].set_ylabel('Frequency')
    axes[0].set_title('Non Disaster Tweets')
    axes[1].set_xlabel('Word Count')
    axes[1].set_title('Disaster Tweets')

    fig.suptitle('Words Per Tweet', fontsize=24, va='baseline')

    fig.tight_layout()
```

In [23]:
```
plot_word_number_histogram(trainv[trainv['target'] == 0]['text'],
                           trainv[trainv['target'] == 1]['text'])
```



In [24]:
```
def plot_word_len_histogram(textno, textye):

    """A function for comparing average word length"""

    fig, axes = plt.subplots(ncols=2, nrows=1, figsize=(18, 6), sharey=True)
    sns.distplot(textno.str.split().apply(lambda x: [len(i) for i in x]).map(
        lambda x: np.mean(x)),
                 ax=axes[0], color='#e74c3c')
    sns.distplot(textye.str.split().apply(lambda x: [len(i) for i in x]).map(
        lambda x: np.mean(x)),
                 ax=axes[1], color='#e74c3c')

    axes[0].set_xlabel('Word Length')
    axes[0].set_ylabel('Frequency')
    axes[0].set_title('Non Disaster Tweets')
    axes[1].set_xlabel('Word Length')
    axes[1].set_title('Disaster Tweets')

    fig.suptitle('Mean Word Lengths', fontsize=24, va='baseline')
    fig.tight_layout()
```

In [25]:
```
plot_word_len_histogram(trainv[trainv['target'] == 0]['text'],
                        trainv[trainv['target'] == 1]['text'])
```

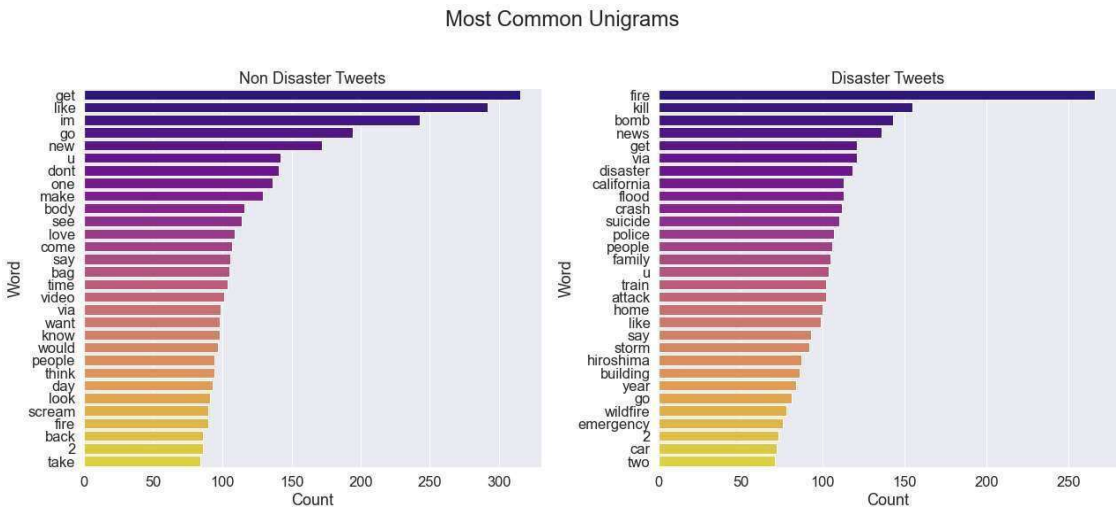


In [26]:
```
lis = [
    trainv[trainv['target'] == 0]['lemma_str'],
    trainv[trainv['target'] == 1]['lemma_str']
]
```

In [27]:
```
fig, axes = plt.subplots(1, 2, figsize=(18, 8))
axes = axes.flatten()

for i, j in zip(lis, axes):
    try:
        new = i.str.split()
        new = new.values.tolist()
        corpus = [word.lower() for i in new for word in i]
        dic = defaultdict(int)
        for word in corpus:
            if word in stop:
                dic[word] += 1

        top = sorted(dic.items(), key=lambda x: x[1], reverse=True)[:15]
        x, y = zip(*top)
        df = pd.DataFrame([x, y]).T
        df = df.rename(columns={0: 'Stopword', 1: 'Count'})
        sns.barplot(x='Count', y='Stopword', data=df, palette='plasma', ax=j)
        plt.tight_layout()
    except:
        plt.close()
        print('No stopwords left in texts.')
        break
```

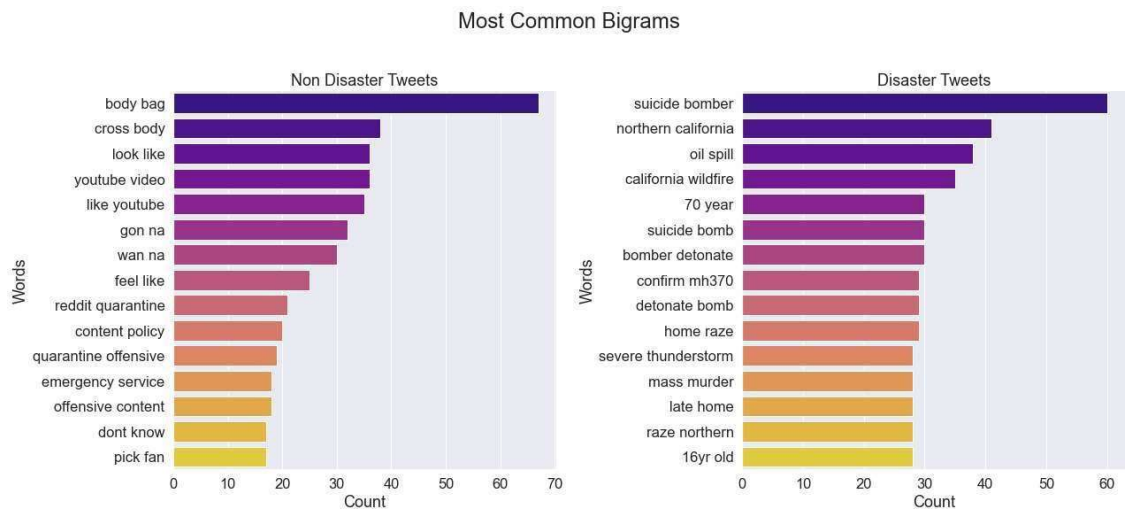
```
No stopwords left in texts.
```

In [28]:
```
# Displaying most common words.

fig, axes = plt.subplots(1, 2, figsize=(18, 8))
axes = axes.flatten()

for i, j in zip(lis, axes):

    new = i.str.split()
    new = new.values.tolist()
    corpus = [word for i in new for word in i]

    counter = Counter(corpus)
    most = counter.most_common()
    x, y = [], []
    for word, count in most[:30]:
        if (word not in stops):
            x.append(word)
            y.append(count)

    sns.barplot(x=y, y=x, palette='plasma', ax=j)
axes[0].set_title('Non Disaster Tweets')

axes[1].set_title('Disaster Tweets')
axes[0].set_xlabel('Count')
axes[0].set_ylabel('Word')
axes[1].set_xlabel('Count')
axes[1].set_ylabel('Word')

fig.suptitle('Most Common Unigrams', fontsize=24, va='baseline')
plt.tight_layout()
```

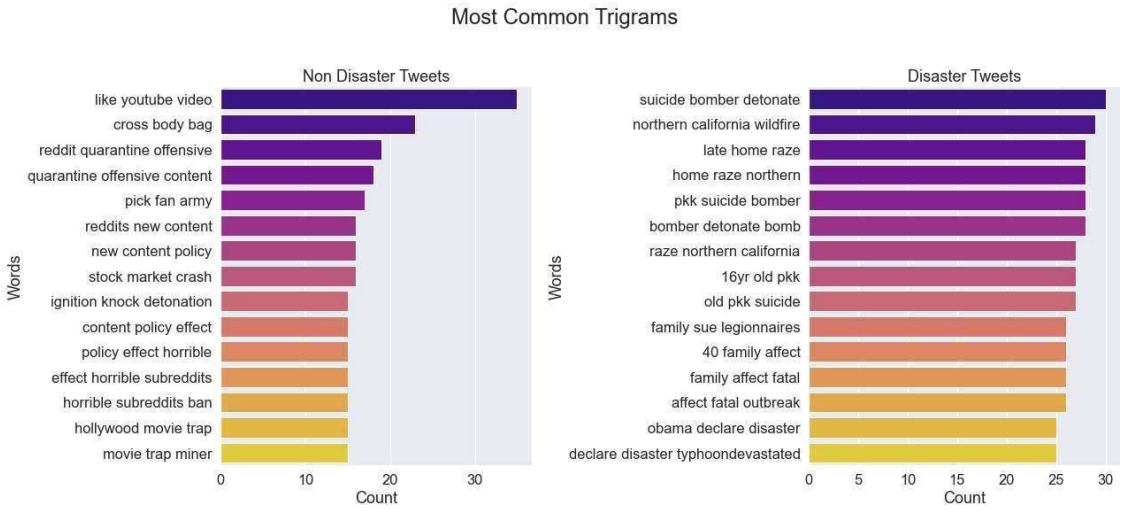


In [29]:
```
def ngrams(n, title):
    """A Function to plot most common ngrams"""
    fig, axes = plt.subplots(1, 2, figsize=(18, 8))
    axes = axes.flatten()
    for i, j in zip(lis, axes):

        new = i.str.split()
        new = new.values.tolist()
        corpus = [word for i in new for word in i]

        def _get_top_ngram(corpus, n=None):
            #getting top ngrams
            vec = CountVectorizer(ngram_range=(n, n),
                                  max_df=0.9,
                                  stop_words='english').fit(corpus)
            bag_of_words = vec.transform(corpus)
            sum_words = bag_of_words.sum(axis=0)
            words_freq = [(word, sum_words[0, idx])
                          for word, idx in vec.vocabulary_.items()]
            words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
            return words_freq[:15]

        top_n_bigrams = _get_top_ngram(i, n)[:15]
        x, y = map(list, zip(*top_n_bigrams))
        sns.barplot(x=y, y=x, palette='plasma', ax=j)

        axes[0].set_title('Non Disaster Tweets')
        axes[1].set_title('Disaster Tweets')
        axes[0].set_xlabel('Count')
        axes[0].set_ylabel('Words')
        axes[1].set_xlabel('Count')
        axes[1].set_ylabel('Words')
        fig.suptitle(title, fontsize=24, va='baseline')
        plt.tight_layout()
```

In [30]:
```
ngrams(2, 'Most Common Bigrams')
```



In [31]:
```
ngrams(3, 'Most Common Trigrams')
```



In [32]:
```
def display_topics(text, no_top_words, topic):

    """ A function for determining the topics present in our corpus with nmf "

    no_top_words = no_top_words
    tfidf_vectorizer = TfidfVectorizer(
        max_df=0.90, min_df=25, max_features=5000, use_idf=True)
    tfidf = tfidf_vectorizer.fit_transform(text)
    tfidf_feature_names = tfidf_vectorizer.get_feature_names()
    doc_term_matrix_tfidf = pd.DataFrame(
        tfidf.toarray(), columns=list(tfidf_feature_names))
    nmf = NMF(n_components=10, random_state=0,
              alpha=.1, init='nndsvd').fit(tfidf)
    print(topic)
    for topic_idx, topic in enumerate(nmf.components_):
        print('Topic %d:' % (topic_idx+1))
        print(' '.join([tfidf_feature_names[i]
                        for i in topic.argsort()[:-no_top_words - 1:-1]]))


display_topics(lis[0], 10, 'Non Disaster Topics\n')
```

```
Non Disaster Topics

Topic 1:
get lol blow good bomb first day demolish someone play
Topic 2:
like video youtube look feel back fire fatality sink mudslide
Topic 3:
im traumatise still disaster na gon attack drown dead weapon
Topic 4:
new emergency full read content post quarantine many storm re
Topic 5:
body bag cross shoulder woman full lady read ebay really
Topic 6:
dont one see know come say make want think fire
Topic 7:
scream fuck phone face good song loud hit baby time
Topic 8:
via youtube god change obliteration news story stop service video
Topic 9:
go content quarantine many explode make reddit let top deluge
Topic 10:
love crush collide woman much death military armageddon would check
```

In [33]:
```
display_topics(lis[1], 10,'Disaster Topics\n')
```

```
Disaster Topics

Topic 1:
fire forest building truck evacuate wild burn california service set
Topic 2:
suicide bomb kill bomber saudi mosque detonate pkk old 16yr
Topic 3:
california wildfire home northern late news raze abc collapse burn

Topic 4:
flood storm rain people train issue severe weather rescue violent
Topic 5:
hiroshima atomic bomb year japan still anniversary 70 war bombing
Topic 6:
via attack wave israeli police evacuation heat post wound car
      7:
Topic family         malaysia wreckage pm debris conclusively investigator find
mh370 8:     confirm
disaster nuclear
orthern          obama natural declare saipan typhoondevastated sign collapse n

Topic 9:
Topic 10:
crash kill accident helicopter air train fear say police car
get watch minute sandstorm swallow airport go im like dont
```

In [43]:
```
# Setting mask for wordcloud.

mask = np.array(Image.open('twittermask.png'))
mask[mask.sum(axis=2) == 0] = 255
```

In [44]:
```
def plot_wordcloud(text, title, title_size):
    """ A function for creating wordcloud images """
    words = text
    allwords = []
    for wordlist in words:
        allwords += wordlist
    mostcommon = FreqDist(allwords).most_common(140)
    wordcloud = WordCloud(
        width=1200,
        height=800,
        background_color='black',
        stopwords=set(STOPWORDS),
        max_words=150,
        scale=3,
        mask=mask,
        contour_width=0.1,
        contour_color='grey',
    ).generate(str(mostcommon))

    def grey_color_func(word,
                        font_size,
                        position,
                        orientation,
                        random_state=None,
                        **kwargs):
        # A definition for creating grey color shades.
        return 'hsl(0, 0%%, %d%%)' % random.randint(60, 100)

    fig = plt.figure(figsize=(18, 18), facecolor='white')
    plt.imshow(wordcloud.recolor(color_func=grey_color_func, random_state=42),
               interpolation='bilinear')
    plt.axis('off')
    plt.title(title,
              fontdict={
                  'size': title_size,
                  'verticalalignment': 'bottom'
              })
    plt.tight_layout(pad=0)
    plt.show()
```

In [47]:
```
from wordcloud import WordCloud, STOPWORDS
plot_wordcloud(trainv[trainv['target'] == 0]['lemmatized'],
               'Most Common Words in Non-Disaster Tweets',
               title_size=30)
```

Most Common Words in Non-Disaster Tweets