Description

The "Natural Language Processing with Disaster Tweets" Kaggle competition was a data science challenge that tasked participants with developing machine learning models to classify tweets as either referring to real disasters or not. It highlighted the importance of natural language processing (NLP) techniques in analyzing unstructured text data and solving real-world problems. Participants used a variety of approaches, from traditional methods to advanced transformer models like BERT, to achieve high-performing models. The competition emphasized data preprocessing, feature engineering, and model evaluation, and it fostered collaboration and knowledge sharing within the data science community. Overall, it showcased the versatility of NLP and its applications in disaster response and management.

In [1]:

```
nt comes with many helpful analytics libraries installed
ggle/python Docker image: https://github.com/kaggle/docker-python
eral helpful packages to load

ar algebra
a processing, CSV file I/O (e.g. pd.read_csv)

ailable in the read-only "../input/" directory
is (by clicking run or pressing Shift+Enter) will list all files under t


 in os.walk('/kaggle/input'):
ames:
n(dirname, filename))

B to the current directory (/kaggle/working/) that gets preserved as out
orary files to /kaggle/temp/, but they won't be saved outside of the cur
```

```
/kaggle/input/nlp-getting-started/sample_submission.csv
/kaggle/input/nlp-getting-started/train.csv
/kaggle/input/nlp-getting-started/test.csv
```

Importing Libraries

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
%config Completer.use_jedi = False

import numpy as np
import pandas as pd
!pip install text_hammer
import text_hammer as th
import seaborn as sns
import matplotlib.pyplot as plt
import re
from wordcloud import STOPWORDS
from collections import defaultdict
#%%time
from tqdm._tqdm_notebook import tqdm_notebook
tqdm_notebook.pandas()
from transformers import AutoTokenizer,TFBertModel

max_len = 36

import tensorflow as tf
tf.config.experimental.list_physical_devices('GPU')

from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.initializers import TruncatedNormal
from tensorflow.keras.losses import CategoricalCrossentropy,BinaryCross
from tensorflow.keras.metrics import CategoricalAccuracy,BinaryAccuracy
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.utils import plot_model
from tensorflow.keras.layers import Input, Dense

from nltk.corpus import stopwords
from wordcloud import WordCloud
```

```
Collecting text_hammer
  Downloading text_hammer-0.1.5-py3-none-any.whl (7.6 kB)
Collecting beautifulsoup4==4.9.1 (from text_hammer)
  Downloading beautifulsoup4-4.9.1-py3-none-any.whl (115 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.1/115.1 kB 3.2 MB/s e
ta 0:00:00
Requirement already satisfied: pandas in /opt/conda/lib/python3.10/sit
e-packages (from text_hammer) (2.0.3)
Requirement already satisfied: numpy in /opt/conda/lib/python3.10/site
-packages (from text_hammer) (1.23.5)
Requirement already satisfied: spacy in /opt/conda/lib/python3.10/site
-packages (from text_hammer) (3.6.1)
Requirement already satisfied: TextBlob in /opt/conda/lib/python3.10/s
ite-packages (from text_hammer) (0.17.1)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python
3.10/site-packages (from beautifulsoup4==4.9.1->text_hammer) (2.3.2.po
st1)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/li
b/python3.10/site-packages (from pandas->text_hammer) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.
10/site-packages (from pandas->text_hammer) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in /opt/conda/lib/python
3.10/site-packages (from pandas->text_hammer) (2023.3)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /opt/con
da/lib/python3.10/site-packages (from spacy->text_hammer) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /opt/con
da/lib/python3.10/site-packages (from spacy->text_hammer) (1.0.4)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /opt/cond
a/lib/python3.10/site-packages (from spacy->text_hammer) (1.0.9)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /opt/conda/lib/p
ython3.10/site-packages (from spacy->text_hammer) (2.0.7)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /opt/conda/li
b/python3.10/site-packages (from spacy->text_hammer) (3.0.8)
Requirement already satisfied: thinc<8.2.0,>=8.1.8 in /opt/conda/lib/p
ython3.10/site-packages (from spacy->text_hammer) (8.1.12)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /opt/conda/lib/
python3.10/site-packages (from spacy->text_hammer) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /opt/conda/lib/p
ython3.10/site-packages (from spacy->text_hammer) (2.4.7)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /opt/conda/l
ib/python3.10/site-packages (from spacy->text_hammer) (2.0.9)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /opt/conda/lib/
python3.10/site-packages (from spacy->text_hammer) (0.9.0)
Requirement already satisfied: pathy>=0.10.0 in /opt/conda/lib/python
3.10/site-packages (from spacy->text_hammer) (0.10.2)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /opt/conda/
lib/python3.10/site-packages (from spacy->text_hammer) (6.3.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /opt/conda/lib/p
ython3.10/site-packages (from spacy->text_hammer) (4.66.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /opt/conda/l
ib/python3.10/site-packages (from spacy->text_hammer) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
/opt/conda/lib/python3.10/site-packages (from spacy->text_hammer) (1.1
0.9)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.10/sit
e-packages (from spacy->text_hammer) (3.1.2)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.1
```

```
0/site-packages (from spacy->text_hammer) (68.0.0)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/pytho
n3.10/site-packages (from spacy->text_hammer) (21.3)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /opt/conda/l
ib/python3.10/site-packages (from spacy->text_hammer) (3.3.0)
Requirement already satisfied: nltk>=3.1 in /opt/conda/lib/python3.10/
site-packages (from TextBlob->text_hammer) (3.2.4)
Requirement already satisfied: six in /opt/conda/lib/python3.10/site-p
ackages (from nltk>=3.1->TextBlob->text_hammer) (1.16.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/
lib/python3.10/site-packages (from packaging>=20.0->spacy->text_hamme
r) (3.0.9)
Requirement already satisfied: typing-extensions>=4.2.0 in /opt/conda/
lib/python3.10/site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.
4->spacy->text_hammer) (4.6.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/
lib/python3.10/site-packages (from requests<3.0.0,>=2.13.0->spacy->tex
t_hammer) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.
10/site-packages (from requests<3.0.0,>=2.13.0->spacy->text_hammer)
(3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/py
thon3.10/site-packages (from requests<3.0.0,>=2.13.0->spacy->text_hamm
er) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/py
thon3.10/site-packages (from requests<3.0.0,>=2.13.0->spacy->text_hamm
er) (2023.7.22)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /opt/conda/lib/py
thon3.10/site-packages (from thinc<8.2.0,>=8.1.8->spacy->text_hammer)
(0.7.10)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /opt/conda/
lib/python3.10/site-packages (from thinc<8.2.0,>=8.1.8->spacy->text_ha
mmer) (0.1.1)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /opt/conda/lib/p
ython3.10/site-packages (from typer<0.10.0,>=0.3.0->spacy->text_hamme
r) (8.1.7)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/pytho
n3.10/site-packages (from jinja2->spacy->text_hammer) (2.1.3)
Installing collected packages: beautifulsoup4, text_hammer
  Attempting uninstall: beautifulsoup4
    Found existing installation: beautifulsoup4 4.12.2
    Uninstalling beautifulsoup4-4.12.2:
      Successfully uninstalled beautifulsoup4-4.12.2
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of t
he following dependency conflicts.
momepy 0.6.0 requires shapely>=2, but you have shapely 1.8.5.post1 whi
ch is incompatible.
Successfully installed beautifulsoup4-4.12.2 text_hammer-0.1.5
```

Loading The Data

In [3]:
```
train_data = pd.read_csv('../input/nlp-getting-started/train.csv',useco
test_data = pd.read_csv('../input/nlp-getting-started/test.csv',usecols
sample_data = pd.read_csv('../input/nlp-getting-started/sample_submissi
```

In [6]:
```
test_data.head()
```

Out[6]:

	id	text
0	0	Just happened a terrible car crash
1	2	Heard about #earthquake is different cities, s...
2	3	there is a forest fire at spot pond, geese are...
3	9	Apocalypse lighting. #Spokane #wildfires
4	11	Typhoon Soudelor kills 28 in China and Taiwan

In [7]:
```
test_data.tail()
```

Out[7]:

	id	text
3258	10861	EARTHQUAKE SAFETY LOS ANGELES □ÛÒ SAFETY FASTE...
3259	10865	Storm in RI worse than last hurricane. My city...
3260	10868	Green Line derailment in Chicago http://t.co/U...
3261	10874	MEG issues Hazardous Weather Outlook (HWO) htt...
3262	10875	#CityofCalgary has activated its Municipal Eme...

In [8]:
```
train_data.head()
```

Out[8]:

	id	text	target
0	1	Our Deeds are the Reason of this #earthquake M...	1
1	4	Forest fire near La Ronge Sask. Canada	1
2	5	All residents asked to 'shelter in place' are ...	1
3	6	13,000 people receive #wildfires evacuation or...	1
4	7	Just got sent this photo from Ruby #Alaska as ...	1

In [9]:
```
train_data.tail()
```

Out[9]:

	id	text	target
7608	10869	Two giant cranes holding a bridge collapse int...	1
7609	10870	@aria_ahrary @TheTawniest The out of control w...	1
7610	10871	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	1
7611	10872	Police investigating after an e-bike collided ...	1
7612	10873	The Latest: More Homes Razed by Northern Calif...	1

In [10]:
```
train_data.shape
```

Out[10]:
```
(7613, 3)
```

In [11]:
```
def text_preprocessing(df,col_name):
    column = col_name
    df[column] = df[column].progress_apply(lambda x:str(x).lower())
#     df[column] = df[column].progress_apply(lambda x: th.cont_exp(x)) #
    df[column] = df[column].progress_apply(lambda x: th.remove_emails(x
    df[column] = df[column].progress_apply(lambda x: th.remove_html_tag
#     df[column] = df[column].progress_apply(lambda x: ps.remove_stopwor
    df[column] = df[column].progress_apply(lambda x: th.remove_special_
    df[column] = df[column].progress_apply(lambda x: th.remove_accented_
#     df[column] = df[column].progress_apply(lambda x: th.make_base(x))
    return(df)
```

In [12]:
```
train_cleaned_data = text_preprocessing(train_data,'text')
```

```
  0%|          | 0/7613 [00:00<?, ?it/s]
  0%|          | 0/7613 [00:00<?, ?it/s]
  0%|          | 0/7613 [00:00<?, ?it/s]
  0%|          | 0/7613 [00:00<?, ?it/s]
  0%|          | 0/7613 [00:00<?, ?it/s]
```

In [13]:
```
train_cleaned_data[train_cleaned_data.target == 0]
```

Out[13]:

	id	text	target
15	23	whats up man	0
16	24	i love fruits	0
17	25	summer is lovely	0
18	26	my car is so fast	0
19	28	what a gooooooooaaaaaal	0
...	...	...	...
7581	10833	engineshed great atmosphere at the british lio...	0
7582	10834	cramer igers 3 words that wrecked disneys stoc...	0
7584	10837	these boxes are ready to explode exploding kit...	0
7587	10841	sirens everywhere	0
7593	10848	i just heard a really loud bang and everyone i...	0

4342 rows × 3 columns

In [14]:
```
train_data = train_cleaned_data.copy()
```

In [15]:
```
train_data.head(10)
```

Out[15]:

	id	text	target
0	1	our deeds are the reason of this earthquake ma...	1
1	4	forest fire near la ronge sask canada	1
2	5	all residents asked to shelter in place are be...	1
3	6	13000 people receive wildfires evacuation orde...	1
4	7	just got sent this photo from ruby alaska as s...	1
5	8	rockyfire update california hwy 20 closed in b...	1
6	10	flood disaster heavy rain causes flash floodin...	1
7	13	im on top of the hill and i can see a fire in ...	1
8	14	theres an emergency evacuation happening now i...	1
9	15	im afraid that the tornado is coming to our area	1

Here,

- Target 1 indicates any accident or disaster
- Target 0 indicates a formal tweets with not much attention

Word Cloud:

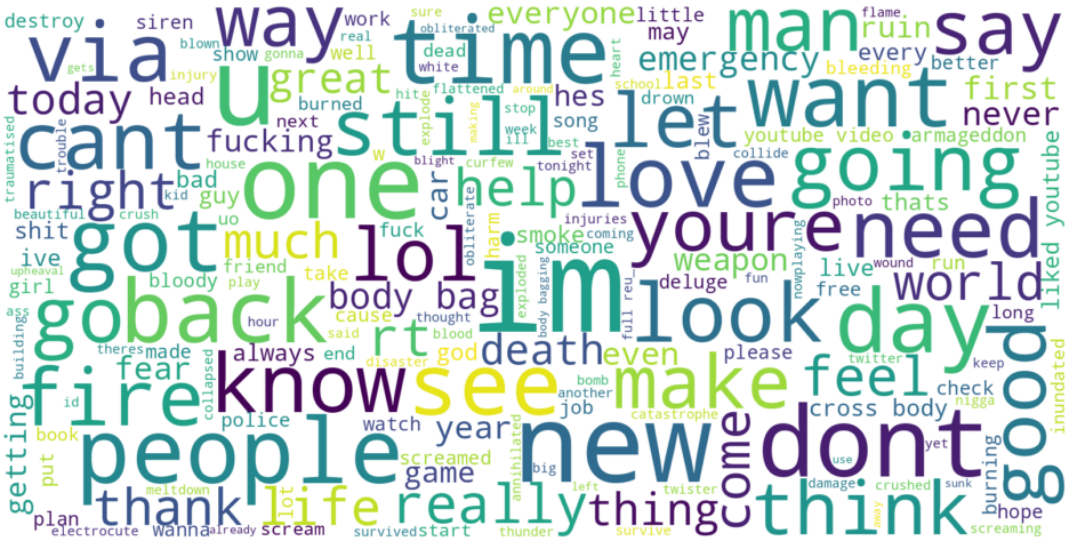
Type *Markdown* and LaTeX: α^2

In [16]:
```
stop_words = set(stopwords.words('english'))
train_data['text'] = train_data['text'].apply(lambda x: ' '.join([word
```

In [18]:
```
#Disaster Tweets wordcloud
disaster_tweets = train_data[train_data.target == 1]
disaster_string = []
for t in disaster_tweets.text:
    disaster_string.append(t)
disaster_string = pd.Series(disaster_string).str.cat(sep=' ')
wordcloud = WordCloud(width=1600, height=800,max_font_size=200, backgro
plt.figure(figsize=(12,10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

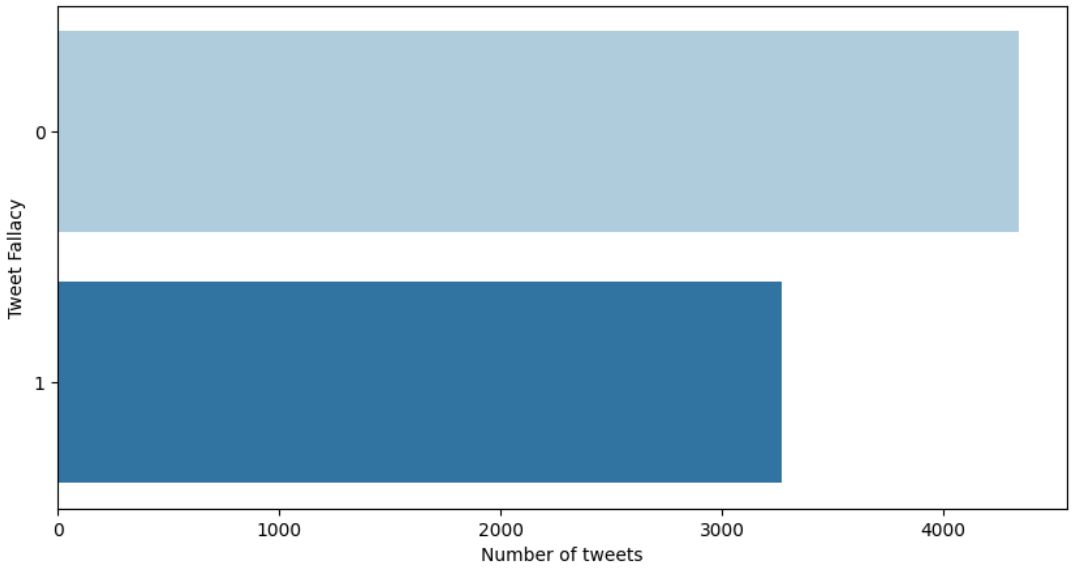
In [20]:

```
# Positive tweets wordcloud
formal_tweets = train_data[train_data.target == 0]
formal_string = []
for t in formal_tweets.text:
    formal_string.append(t)
formal_string = pd.Series(formal_string).str.cat(sep=' ')
wordcloud = WordCloud(width=1600, height=800,max_font_size=200, backgrou
plt.figure(figsize=(12,10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



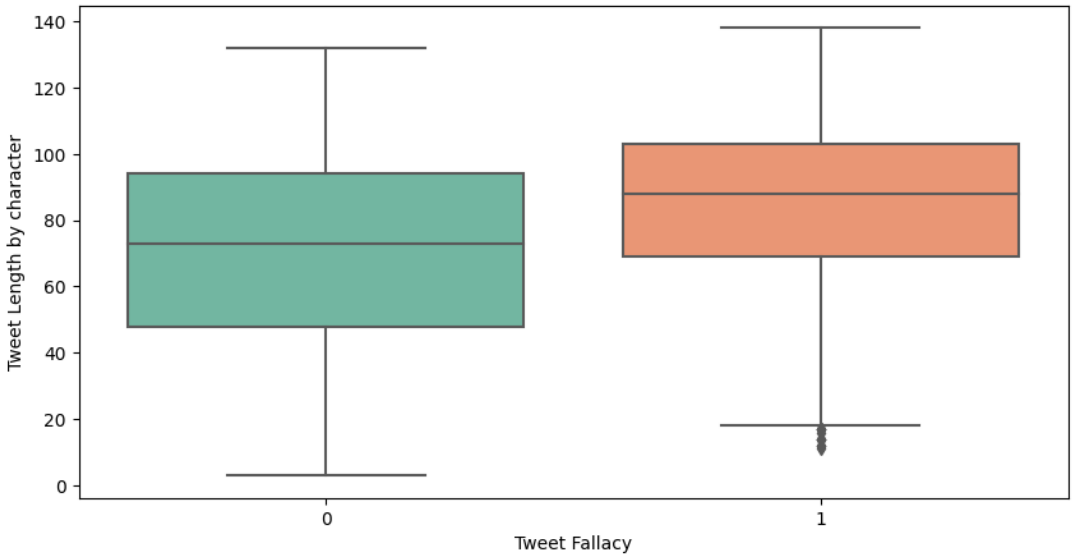
In [21]:

```
#Visualizing class distribution
plt.figure(figsize=(10,5))
sns.countplot(y='target',data = train_data,palette="Paired")
plt.ylabel("Tweet Fallacy")
plt.xlabel("Number of tweets")
plt.show()
```



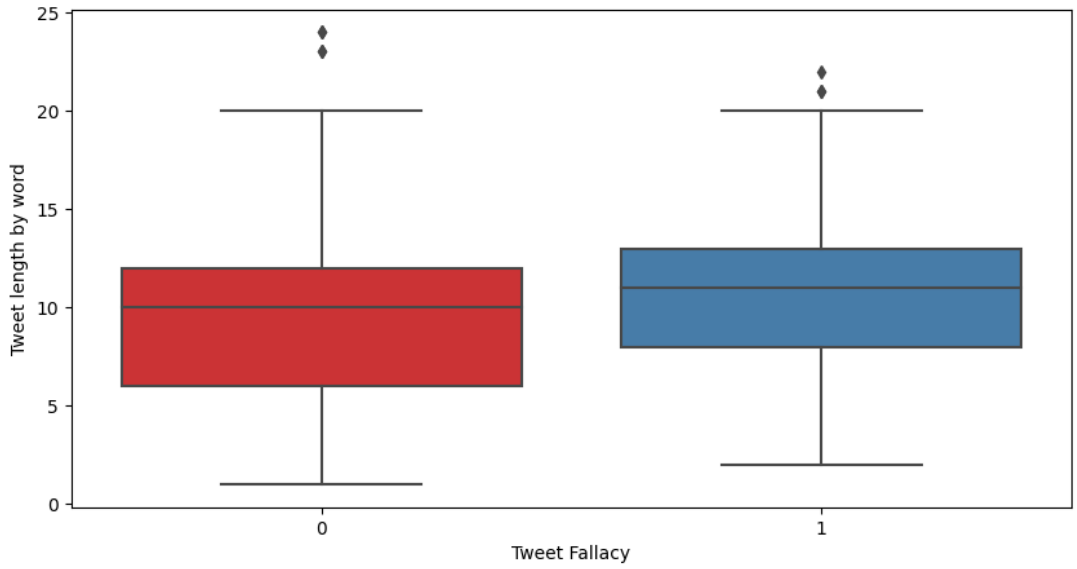
In [22]:

```
#Visualizing tweet length by characaters
plt.figure(figsize=(10,5))
train_sent = train_data['text'].str.len()
sns.boxplot(x="target",y=train_sent,data=train_data,palette="Set2")
plt.xlabel("Tweet Fallacy")
plt.ylabel("Tweet Length by character")
plt.show()
```



In [23]:

```
#Visualizing tweet length by words
plt.figure(figsize=(10,5))
train_sent = train_data['text'].str.split().map(lambda x : len(x))
sns.boxplot(x="target",y=train_sent,data=train_data,palette="Set1")
plt.xlabel("Tweet Fallacy")
plt.ylabel("Tweet length by word")
plt.show()
```

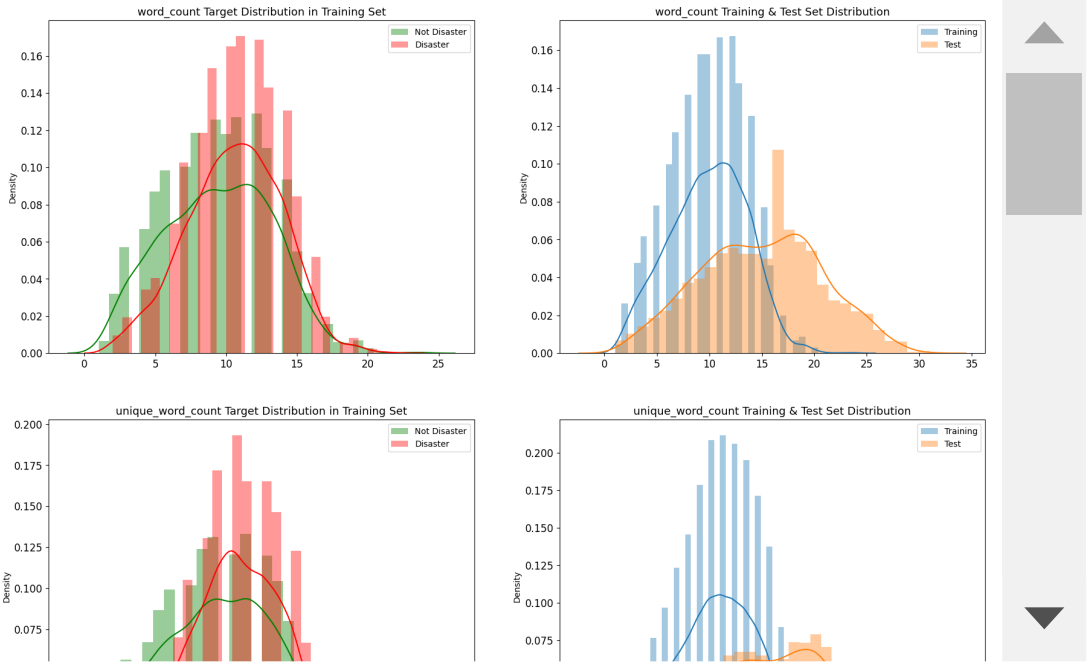


In [24]:

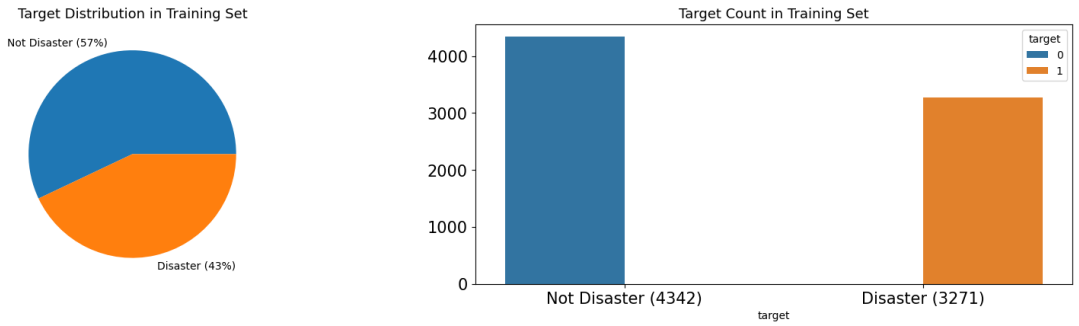
```
# word_count
train_data['word_count'] = train_data['text'].apply(lambda x: len(str(x
test_data['word_count'] = test_data['text'].apply(lambda x: len(str(x).

# unique_word_count
train_data['unique_word_count'] = train_data['text'].apply(lambda x: le
test_data['unique_word_count'] = test_data['text'].apply(lambda x: len(

# stop_word_count
train_data['stop_word_count'] = train_data['text'].apply(lambda x: len(
test_data['stop_word_count'] = test_data['text'].apply(lambda x: len([w

# url_count
train_data['url_count'] = train_data['text'].apply(lambda x: len([w for
test_data['url_count'] = test_data['text'].apply(lambda x: len([w for w

# mean_word_length
train_data['mean_word_length'] = train_data['text'].apply(lambda x: np.
test_data['mean_word_length'] = test_data['text'].apply(lambda x: np.me

# char_count
train_data['char_count'] = train_data['text'].apply(lambda x: len(str(x
test_data['char_count'] = test_data['text'].apply(lambda x: len(str(x))
```

In [25]:

```
METAFEATURES = ['word_count', 'unique_word_count', 'stop_word_count', '
                'char_count']
DISASTER_TWEETS = train_data['target'] == 1

fig, axes = plt.subplots(ncols=2, nrows=len(METAFEATURES), figsize=(20,

for i, feature in enumerate(METAFEATURES):
    sns.distplot(train_data.loc[~DISASTER_TWEETS][feature], label='Not
    sns.distplot(train_data.loc[DISASTER_TWEETS][feature], label='Disas

    sns.distplot(train_data[feature], label='Training', ax=axes[i][1])
    sns.distplot(test_data[feature], label='Test', ax=axes[i][1])

    for j in range(2):
        axes[i][j].set_xlabel('')
        axes[i][j].tick_params(axis='x', labelsize=12)
        axes[i][j].tick_params(axis='y', labelsize=12)
        axes[i][j].legend()

    axes[i][0].set_title(f'{feature} Target Distribution in Training Se
    axes[i][1].set_title(f'{feature} Training & Test Set Distribution',

plt.show()
```



In [26]:
```
fig, axes = plt.subplots(ncols=2, figsize=(17, 4), dpi=100)
plt.tight_layout()

train_data.groupby('target').count()['id'].plot(kind='pie', ax=axes[0],
sns.countplot(x=train_data['target'], hue=train_data['target'], ax=axes

axes[0].set_ylabel('')
axes[1].set_ylabel('')
axes[1].set_xticklabels(['Not Disaster (4342)', 'Disaster (3271)'])
axes[0].tick_params(axis='x', labelsize=15)
axes[0].tick_params(axis='y', labelsize=15)
axes[1].tick_params(axis='x', labelsize=15)
axes[1].tick_params(axis='y', labelsize=15)

axes[0].set_title('Target Distribution in Training Set', fontsize=13)
axes[1].set_title('Target Count in Training Set', fontsize=13)

plt.show()
```



BERT - Bidirectional Encoder Representations from Transformers

- BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.
- The best part about BERT is that we can use the BERT models to extract high quality language features from our text data.

LOADING THE BERT MODEL:

In [ ]:
```
tokenizer = AutoTokenizer.from_pretrained('bert-large-uncased')
bert = TFBertModel.from_pretrained('bert-large-uncased')
```

In [30]:
```
tokenizer('Happy learning and keep kaggling &*&*&&')
```

Out[30]:
```
{'input_ids': [101, 3407, 4083, 1998, 2562, 10556, 13871, 2989, 1004,
1008, 1004, 1008, 1004, 1004, 102], 'token_type_ids': [0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'attention_mask': [1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1]}
```

CONVERSION OF OUR TEXT DATA INTO BERT INPUT FORMAT:

In [31]:
```
print("max len of tweets",max([len(x.split()) for x in train_data.text]
max_length = 36
```

```
max len of tweets 24
```

In [32]:
```
x_train = tokenizer(
    text=train_data.text.tolist(),
    add_special_tokens=True,
    max_length=36,
    truncation=True,
    padding=True,
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
```

In [33]:
```
x_train['input_ids'].shape
```

Out[33]:
```
TensorShape([7613, 36])
```

In [34]:
```
x_train['attention_mask'].shape
```

Out[34]:
```
TensorShape([7613, 36])
```

In [35]:
```
y_train = train_data.target.values
y_train
```

Out[35]:
```
array([1, 1, 1, ..., 1, 1, 1])
```

In [36]:
```
train_data.target.value_counts()
```

Out[36]:
```
target
0    4342
1    3271
Name: count, dtype: int64
```

BUILDING THE MODEL ARCHITECTURE:

In [37]:
```
input_ids = Input(shape=(max_len,), dtype=tf.int32, name="input_ids")
input_mask = Input(shape=(max_len,), dtype=tf.int32, name="attention_ma
# embeddings = dbert_model(input_ids,attention_mask = input_mask)[0]


embeddings = bert(input_ids,attention_mask = input_mask)[1] #(0 is the
# out = tf.keras.layers.GlobalMaxPool1D()(embeddings)
out = tf.keras.layers.Dropout(0.1)(embeddings)

out = Dense(128, activation='relu')(out)
out = tf.keras.layers.Dropout(0.1)(out)
out = Dense(32,activation = 'relu')(out)

y = Dense(1,activation = 'sigmoid')(out)

model = tf.keras.Model(inputs=[input_ids, input_mask], outputs=y)
model.layers[2].trainable = True

# for training bert our lr must be so small
```

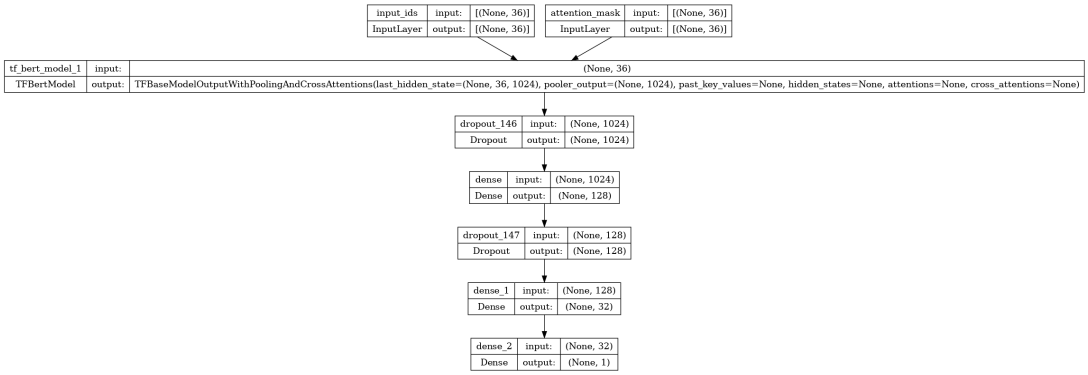
In [38]:
```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
Connected to                  
=================================================================
 input_ids (InputLayer)      [(None, 36)]              0         
[]                            

 attention_mask (InputLayer  [(None, 36)]              0         
)                             []                            

 tf_bert_model_1 (TFBertMod  TFBaseModelOutputWithPo  335141888 
el)                           olingAndCrossAttentions(last_hidde
n_state=(None, 36, 1024),
 pooler_output=(None, 1024),
 past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)
['input_ids[0][0]',
                              'attention_mask[0][0]']

 dropout_146 (Dropout)       (None, 1024)              0         
['tf_bert_model_1[0][1]']     

 dense (Dense)               (None, 128)               131200    
['dropout_146[0][0]']         

 dropout_147 (Dropout)       (None, 128)               0         
['dense[0][0]']               

 dense_1 (Dense)             (None, 32)                4128      
['dropout_147[0][0]']         

 dense_2 (Dense)             (None, 1)                 33        
['dense_1[0][0]']             

=================================================================
Total params: 335,277,249
Trainable params: 335,277,249
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
optimizer = Adam(
    learning_rate=6e-06, # this learning rate is for bert model.
    epsilon=1e-08,
    decay=0.01,
    clipnorm=1.0)

# Set loss and metrics
loss = BinaryCrossentropy(from_logits = True)
metric = BinaryAccuracy('accuracy'),
# Compile the model
model.compile(
    optimizer = optimizer,
    loss = loss,
    metrics = metric)
```

In [41]:
```
plot_model(model, show_shapes = True)
```

Out[41]:



In [ ]:
```
# Fit the model
final = model.fit(
    x ={'input_ids':x_train['input_ids'],'attention_mask':x_train['atte
    y = y_train,
#   validation_split = 0.1,
  epochs=9,
    batch_size=10
)
```

Epoch 1/9 762/762 [==============================] - 215s 233ms/step - loss: 0.5084 - accuracy: 0.7644 Epoch 2/9 762/762 [==============================] - 178s 234ms/step - loss: 0.4231 - accuracy: 0.8235 Epoch 3/9 762/762 [==============================] - 178s 234ms/step - loss: 0.4058 - accuracy: 0.8382 Epoch 4/9 762/762 [==============================] - 179s 234ms/step - loss: 0.4004 - accuracy: 0.8394 Epoch 5/9 762/762 [==============================] - 179s 235ms/step - loss: 0.3897 - accuracy: 0.8400 Epoch 6/9 762/762 [==============================] - 179s 235ms/step - loss: 0.3862 - accuracy: 0.8449 Epoch 7/9 762/762 [==============================] - 179s 235ms/step - loss: 0.3809 - accuracy:

0.8501 Epoch 8/9 762/762 [==============================] - 180s 237ms/step - loss: 0.3788 - accuracy: 0.8487 Epoch 9/9 762/762 [==============================] - 179s 235ms/step - loss: 0.3695 - accuracy: 0.8550

VISUALIZATION OF LOSS AND ACCURACY CURVE:

In [49]:

```
def visual_accuracy_and_loss(final):
    acc = final.history['accuracy']
    loss = final.history['loss']
    epochs_plot = np.arange(1, len(loss) + 1)
    plt.clf()
    plt.plot(epochs_plot, acc, 'r', label='Accuracy')
    plt.plot(epochs_plot, loss, 'b:', label='Loss')
    plt.title('VISUALIZATION OF LOSS AND ACCURACY CURVE')
    plt.xlabel('Epochs')
    plt.legend()
    plt.show()
```

In [50]:
```
test_data
```

Out[50]:

	id	text	word_count	unique_word_count	stop_word_count	url_cou
0	0	Just happened a terrible car crash	6	6	2	
1	2	Heard about #earthquake is different cities, s...	9	9	2	
2	3	there is a forest fire at spot pond, geese are...	19	19	10	
3	9	Apocalypse lighting. #Spokane #wildfires	4	4	0	
4	11	Typhoon Soudelor kills 28 in China and Taiwan	8	8	2	
...	...	...	...	...	...	
3258	10861	EARTHQUAKE SAFETY LOS ANGELES □ÛÒ SAFETY FASTE...	8	7	0	
3259	10865	Storm in RI worse than last hurricane. My city...	23	22	7	
3260	10868	Green Line derailment in Chicago http://t.co/U...	6	6	1	
3261	10874	MEG issues Hazardous Weather Outlook (HWO) htt...	7	7	0	
3262	10875	#CityofCalgary has activated its Municipal Eme...	8	8	2	

3263 rows × 8 columns

In [51]:
```
x_test = tokenizer(
    text=test_data.text.tolist(),
    add_special_tokens=True,
    max_length=36,
    truncation=True,
    padding=True,
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
```

PREDICTION:

In [52]:
```
predicted = model.predict({'input_ids':x_test['input_ids'],'attention_m
```

```
102/102 [==============================] - 615s 6s/step
```

In [53]:
```
y_predicted = np.where(predicted>0.5,1,0)
```

In [54]:
```
y_predicted = y_predicted.reshape((1,3263))[0]
```

In [55]:
```
sample_data.head()
```

Out[55]:

	id	target
0	0	0
1	2	0
2	3	0
3	9	0
4	11	0

In [56]:
```
sample_data.to_csv('submission.csv',index = False)
print(" Successfully completed! ")
```

```
 Successfully completed!
```

Conclusion

In conclusion, the "Natural Language Processing with Disaster Tweets" Kaggle competition has been a remarkable journey into the world of text analysis and machine learning. This competition provided a platform for data scientists, researchers, and enthusiasts from around the world to come together and tackle a critical problem: identifying tweets that refer to real disasters. As we reflect on this competition, several key takeaways and observations come to the forefront.

Firstly, the competition underscored the significance of natural language processing (NLP) techniques in addressing real-world challenges. NLP has the potential to extract valuable insights from unstructured textual data, and this competition demonstrated its applicability in disaster response and management.

Secondly, the diverse approaches and methodologies used by participants highlighted the versatility of NLP models and techniques. From traditional models like TF-IDF and logistic regression to cutting-edge transformer models like BERT and GPT, the competition showcased the power of innovation and creativity in solving complex NLP tasks.

Furthermore, the competition emphasized the importance of data preprocessing, feature engineering, and model evaluation. Clean and well-structured data, along with thoughtful feature engineering, played a crucial role in achieving high-performing models. Rigorous evaluation metrics such as F1-score and precision-recall curves helped participants fine-tune their models and ensure robust performance.

Lastly, collaboration and knowledge sharing were pivotal in this competition's success. Participants shared their insights, code snippets, and strategies on forums and platforms, contributing to a vibrant and collaborative data science community.

As we move forward from this competition, it is essential to recognize that NLP continues to evolve rapidly. New techniques and models will emerge, further expanding our capabilities in understanding and extracting insights from textual data. The skills and knowledge gained from this competition will undoubtedly prove valuable in addressing various other NLP challenges and real-world applications.

In closing, the "Natural Language Processing with Disaster Tweets" Kaggle competition not only pushed the boundaries of NLP but also brought together a community of data enthusiasts dedicated to making a positive impact. It serves as a testament to the power of data science and collaborative problem-solving in addressing critical global issues.

Type *Markdown* and LaTeX: α^2