In [82]:
```
# !pip install nltk
# !pip install --upgrade pip
# !pip install --upgrade tensorflow
```

In [83]:
```
import re
import os
import string
import unicodedata

import numpy as np
import pandas as pd
import nltk
nltk.download('words')
import seaborn as sns
import matplotlib.pyplot as plt

from nltk.stem.porter import PorterStemmer
from nltk.corpus import words
word_dict = words.words()
stemmer = PorterStemmer()
```

```
[nltk_data] Downloading package words to /usr/share/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

In [84]:
```
import warnings

def fxn():
    warnings.warn("deprecated", DeprecationWarning)

with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    fxn()
```

Read data

In [85]:
```
train_df = pd.read_csv('/kaggle/input/pg-final-project-datasets/train.csv')
test_df = pd.read_csv('/kaggle/input/pg-final-project-datasets/test.csv')
print('Train dataset shape:', train_df.shape)
print('Test dataset shape:', test_df.shape)
print()

train_df.head()
display(train_df.iloc[35:40])
```

```
Train dataset shape: (7613, 5)
Test dataset shape: (3263, 4)
```

	id	keyword	location	text	target
35	53	ablaze	London, UK	On plus side LOOK AT THE SKY LAST NIGHT IT WAS...	0
36	54	ablaze	Pretoria	@PhDSquares #mufc they've built so much hype a...	0
37	55	ablaze	World Wide!!	INEC Office in Abia Set Ablaze - http://t.co/3...	1
38	56	ablaze	NaN	Barbados #Bridgetown JAMAICA ÛÒ Two cars set ...	1
39	57	ablaze	Paranaque City	Ablaze for you Lord :D	0

Missing values

In [86]:
```
def print_missing_values(df, name):
    print(f'Missing values in { name}')
    for col in df.columns:
        col_missing = df[col].isna().sum()
        print(f'{col}: {100*col_missing/len(df):.2f}%',)
    print()
print_missing_values(train_df, 'Training dataset')
print_missing_values(test_df, 'Test dataset')

print()

# fill the word 'unk' to fill missing fields

train_df = train_df.fillna('unk')
test_df = test_df.fillna('unk')
```

```
Missing values in Training dataset
id: 0.00%
keyword: 0.80%
location: 33.27%
text: 0.00%
target: 0.00%

Missing values in Test dataset
id: 0.00%
keyword: 0.80%
location: 33.86%
text: 0.00%
```

Duplicate records in dataset

In [87]:
```
cols = ['text', 'target', 'keyword', 'location']
dups_df = train_df[train_df.duplicated(subset = cols,)].sort_values(by = cols)
print(f'There are {len(dups_df)} duplicated records in training dataset.')

cols = ['text', 'keyword', 'location']
dups_df = test_df[test_df.duplicated(subset = cols,)].sort_values(by = cols)
print(f'There are {len(dups_df)} duplicated records in test dataset.')
```

```
There are 52 duplicated records in training dataset.
There are 11 duplicated records in test dataset.
```

We should remove those duplicated records from training set.

In [88]:
```
cols = ['text', 'target', 'keyword', 'location']
train_df = train_df.drop_duplicates(cols).reset_index(drop = True)
```

Delete Duplicate tweet text

In [89]:
```
cols = ['text']
dups_df = train_df[train_df.duplicated(subset = cols)].sort_values(by = cols)
print(f'There is {len(dups_df)} duplicated tweets in training dataset.')
print()
dups_df = dups_df.groupby(cols).agg({
    'id':'count',
    'keyword': 'nunique',
    'target': 'nunique',
    'location': 'nunique',
}).reset_index().rename(columns = {'id': 'text_dup_times'})

dups_df.groupby('text_dup_times').agg({
    'keyword': 'sum',
    'target': 'sum',
    'location': 'sum',
}).reset_index()
```

```
There is 58 duplicated tweets in training dataset.
```

Out[89]:

	text_dup_times	keyword	target	location
0	1	40	40	40
1	2	7	11	14
2	4	1	1	4

It looks to me that:

- When the same tweet is duplicated 4 time, it refers to disaster/not disaster with the same keyword in 4 locations (this can't be true). those duplicated tweets should be deleted
- When the same tweet is duplicated 2 or 1 time, it was classified as disaster sometimes and sometimes as not a disastor. This is also noise in the data. We should delete those records

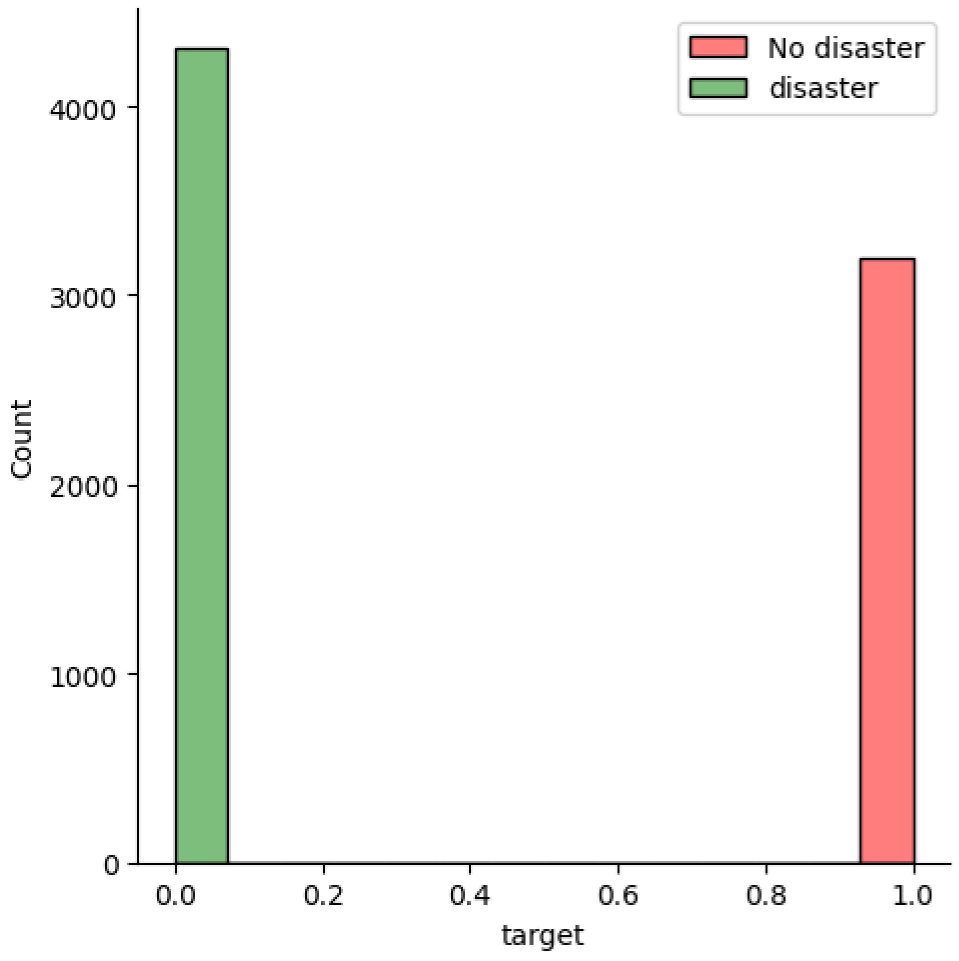
but we should be careful about the label. We can elelmiate those duplicated tweets by voting on thier labels

In [90]:
```
train_df = train_df.groupby(by = ['text']).agg({
    'id': 'first',
    'location': lambda x:x.value_counts().index[0],
    'keyword':lambda x:x.value_counts().index[0],
    'target': lambda x:x.value_counts().index[0],
}).reset_index()
```

Target Column

In [91]:
```
sns.displot(x = 'target', hue = 'target', data = train_df, palette = ['green',
plt.legend(['No disaster', 'disaster'])
plt.show()
```

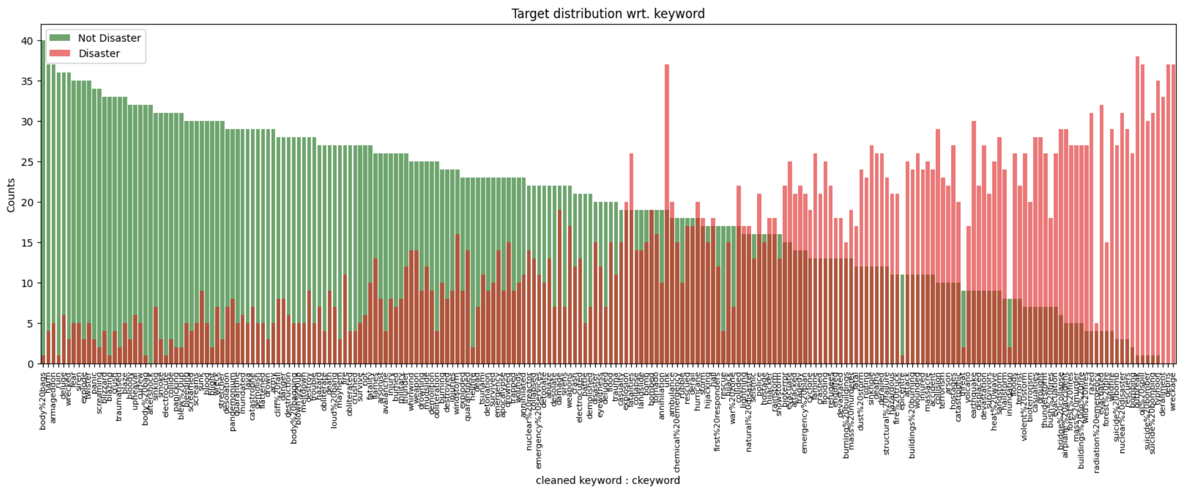
```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarni
ng: use_inf_as_na option is deprecated and will be removed in a future versio
n. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarni
ng: When grouping with a length-1 list-like, you will need to pass a length-1
tuple to get_group in a future version of pandas. Pass `(name,)` instead of `
name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarni
ng: When grouping with a length-1 list-like, you will need to pass a length-1
tuple to get_group in a future version of pandas. Pass `(name,)` instead of `
name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarni
ng: When grouping with a length-1 list-like, you will need to pass a length-1
tuple to get_group in a future version of pandas. Pass `(name,)` instead of `
name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
```



Target distribution based on keyword

In [92]:
```
df = pd.pivot_table(train_df, index = ['keyword'], columns = ['target'], value
df = df.sort_values(by = [0], ascending = False)
plt.figure(figsize = (20, 6))
sns.barplot(y = 0, x = 'keyword', data = df, color = "g", alpha = 0.6, label =
sns.barplot(y = 1, x = 'keyword', data = df, color = 'r', alpha = 0.6, label =
plt.title('Target distribution wrt. keyword')
plt.xlabel('cleaned keyword : ckeyword')
plt.ylabel('Counts')
plt.xticks(rotation = 90, fontsize = 8)
plt.legend()
```

Out[92]:
```
<matplotlib.legend.Legend at 0x7824e86a49d0>
```

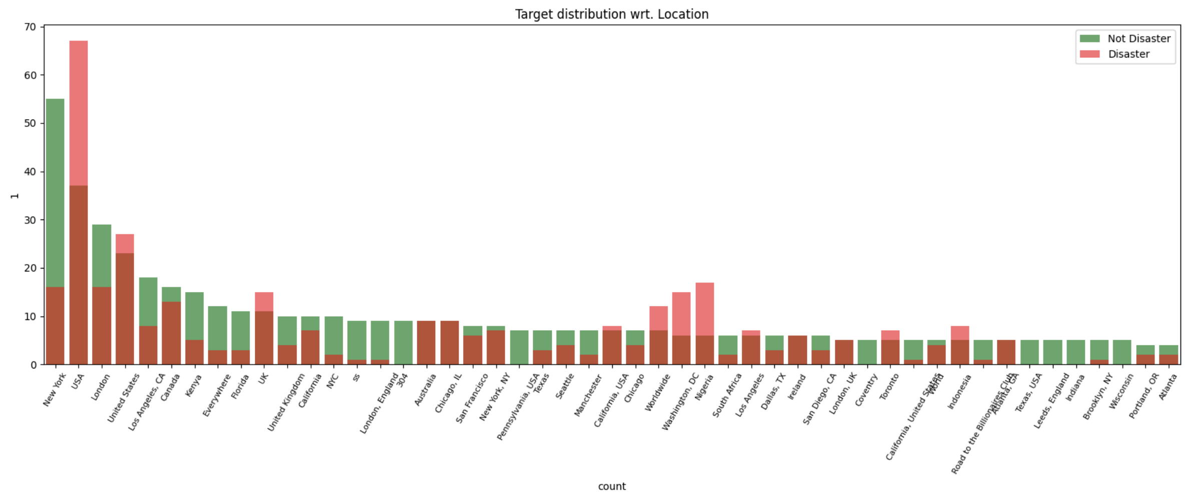


In [93]:
```
train_df['keyword'].value_counts()[0: 10]
```

Out[93]:
```
keyword
unk            56
fatalities     45
deluge         42
armageddon     42
damage         41
body%20bags    41
harm           41
evacuate       40
twister        40
windstorm      40
Name: count, dtype: int64
```

Target distribution based on Location

In [94]:
```
df = pd.pivot_table(train_df, index = 'location', columns = 'target', values =
df = df.sort_values(by = 0, ascending = False)
lim = 50
plt.figure(figsize = (20, 6))
sns.barplot(x = 'location', y = 0, data = df.iloc[1:lim], color = 'g', alpha =
sns.barplot(x = 'location', y = 1, data = df.iloc[1:lim], color = 'r', alpha =
plt.xlabel('count')
plt.title('Target distribution wrt. Location')
plt.xticks(rotation = 60, fontsize = 8)
plt.legend()
plt.show()


# This is just to know that dataset contains various different locations & is
train_df['location'].value_counts()[0:20]
```



Out[94]:
```
location
unk                  2482
USA                   104
New York               71
United States          50
London                 45
Canada                 29
Los Angeles, CA        26
UK                     26
Nigeria                23
India                  21
Washington, DC         21
Kenya                  20
Mumbai                 20
Worldwide              19
Australia              18
Chicago, IL            18
California             17
California, USA        15
New York, NY           15
Everywhere             15
Name: count, dtype: int64
```

Text Normalisation of tweets

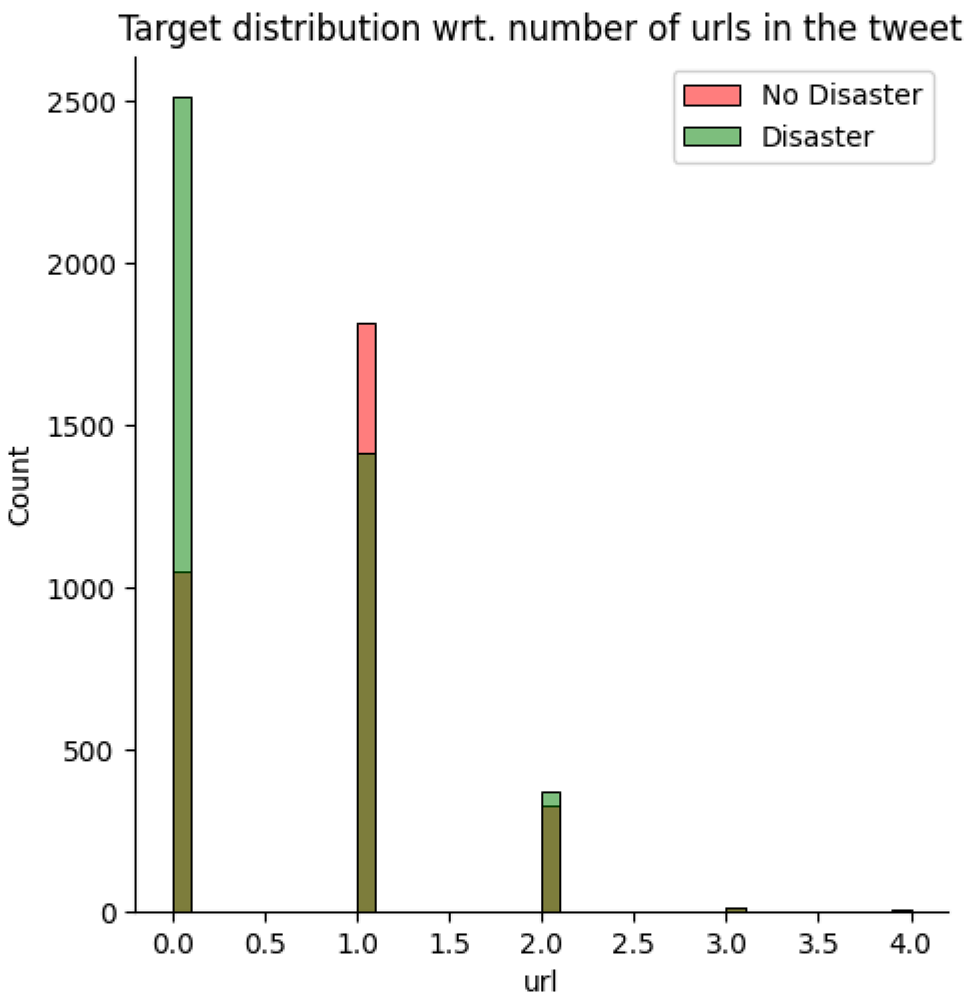
In [95]:
```
# 'text' : inbuilt attribute of dataset
# 'ctext' : tweet text after cleaning

train_df['ctext'] = train_df['text'].copy()

def clean_txt(txt):
    # normalizing the text
    res = unicodedata.normalize('NFKC', txt)
    # remove non_printable characters
    res = re.sub(r'[^\x00-\x7F]+', r'', res)
    # remove retweet chars
    res = re.sub(r'^RT[\s]+', r'', res)
    # remove stock market tiker
    res = re.sub(r'\$\w*', r'', res)
    # replalce less, greater, and chars
    res = re.sub(r'&lt;', r'<', res)
    res = re.sub(r'&gt;', r'>', res)
    res = re.sub(r'&amp;?', r'and', res)
    # remove html tags
    res = re.sub(r'<[^>]*?>', r'', res)
    # separate contected hashtags
    res = re.sub(r'#', r' #', res)
    res = re.sub(r'\s#\s', r' ', res)
    return res

train_df['ctext'] = train_df['ctext'].apply(clean_txt )
```

URL distribution visualisation

In [96]:

```
regex = r'http\S+|www\.\S+'
sub_str = r' website '
def apply_regex(txt, regex, sub_str):
    matches = re.findall(regex, txt)
    if matches:
        ctxt = re.sub(regex, sub_str, txt)
        return ctxt, len(matches)
    else:
        return txt, 0
res = train_df['ctext'].apply(apply_regex, args = [regex, sub_str],)
train_df['ctext'] = res.apply(lambda t:t[0])
train_df['url'] = res.apply(lambda t:t[1])
sns.displot(x = 'url', hue = 'target', data = train_df, palette = ['g', 'r'],
plt.legend(['No Disaster', 'Disaster'])
plt.title('Target distribution wrt. number of urls in the tweet')
plt.show()
#print(train_df)
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarni
ng: use_inf_as_na option is deprecated and will be removed in a future versio
n. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarni
ng: When grouping with a length-1 list-like, you will need to pass a length-1
tuple to get_group in a future version of pandas. Pass `(name,)` instead of `
name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarni
ng: When grouping with a length-1 list-like, you will need to pass a length-1
tuple to get_group in a future version of pandas. Pass `(name,)` instead of `
name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarni
ng: When grouping with a length-1 list-like, you will need to pass a length-1
tuple to get_group in a future version of pandas. Pass `(name,)` instead of `
name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
```



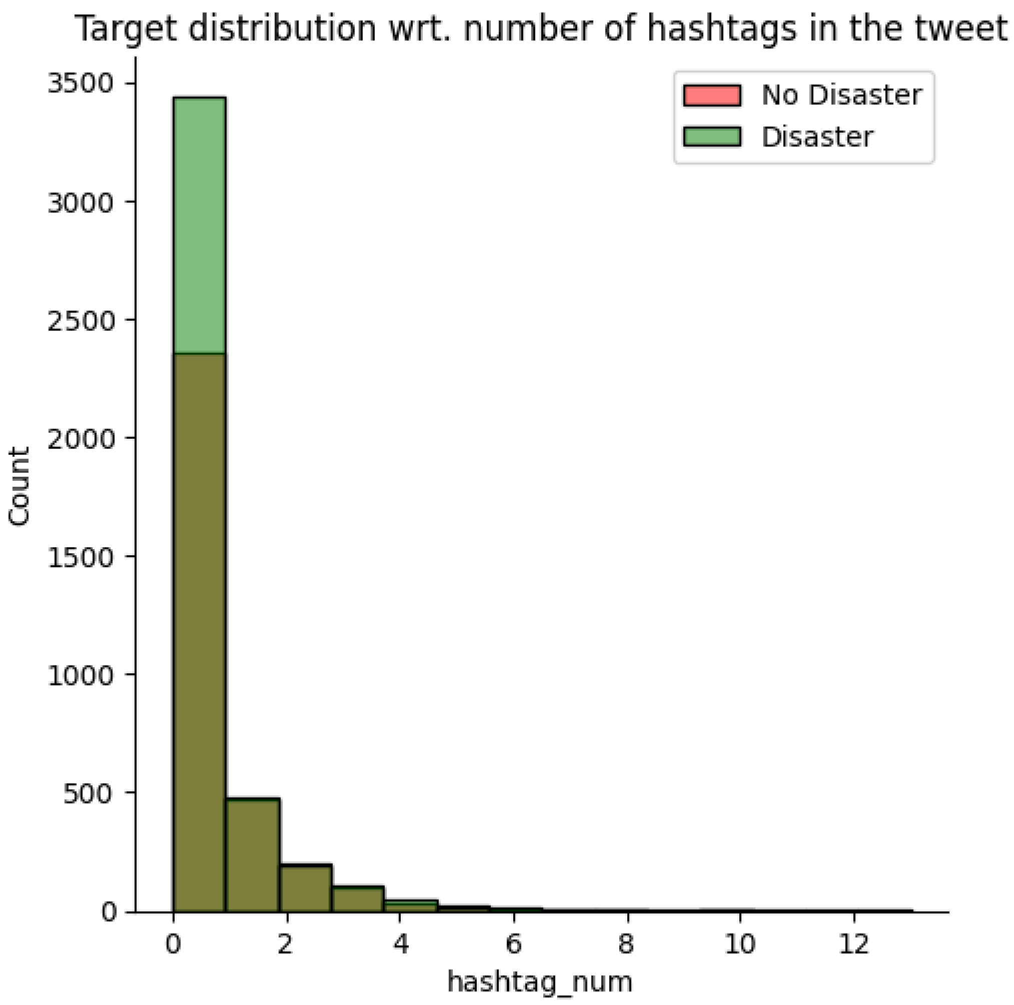
Analysis of effect of having url in the tweet

- When people share no urls in the tweet, it is more likely to be not about a disaster
- When people share one url in the tweet, it is more likely to be about a disaster

Hash tags and mentions

In [97]:
```
def validate_hashtag(hashtags):
    res = []
    for hashtag in hashtags:
        valid = re.findall(r'[a-zA-Z]+', hashtag)
        if valid:
            res.append(re.sub(r'[^\w]*', r'', hashtag).lower())
        else:
            res.append('unk')
    return res

def extract_hashtags(txt):
    regex = r'#+\s*\S+'
    hash_tag_matches = re.findall(regex, txt)
    if hash_tag_matches:
        ctxt = re.sub(r'#+\s*(?P<hash>\S+)', r'\g<hash>', txt)
        validated_hash = list(set(validate_hashtag(hash_tag_matches)))
        return ctxt, len(validated_hash), validated_hash
    else:
        return txt, 0, ['none']

res = train_df['ctext'].apply(extract_hashtags)
train_df['ctext'] = res.apply(lambda t:t[0])
train_df['hashtag_num'] = res.apply(lambda t:t[1])
train_df['hashtags'] = res.apply(lambda t:t[2])
sns.displot(x = 'hashtag_num', hue = 'target', data = train_df, palette = ['g'
plt.legend(['No Disaster', 'Disaster'])
plt.title('Target distribution wrt. number of hashtags in the tweet')
plt.show()
#print(train_df)
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarni
ng: use_inf_as_na option is deprecated and will be removed in a future versio
n. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarni
ng: When grouping with a length-1 list-like, you will need to pass a length-1
tuple to get_group in a future version of pandas. Pass `(name,)` instead of `
name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarni
ng: When grouping with a length-1 list-like, you will need to pass a length-1
tuple to get_group in a future version of pandas. Pass `(name,)` instead of `
name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1075: FutureWarni
ng: When grouping with a length-1 list-like, you will need to pass a length-1
tuple to get_group in a future version of pandas. Pass `(name,)` instead of `
name` to silence this warning.
  data_subset = grouped_data.get_group(pd_key)
```


Target distribution wrt. number of hashtags in the tweet
No Disaster
Disaster
Count
hashtag_num
3500
3000
2500
2000
1500
1000
500
0
0
2
4
6
8
10
12

How many words do exist in the dictionary

In [98]:
```
#print(train_df['ctext'].to_string())
txt = ' '.join(train_df.ctext.values)
txt = re.sub(r'[%s]' % re.escape(string.punctuation), r' ', txt)
txt = re.sub(r'\s+', r' ', txt)
tokens = set(txt.split())
word_dict = set(word_dict)
not_word_tokens = tokens - tokens.intersection(word_dict)

print(f'Tweets contain unknown words: {100 * len(not_word_tokens) / len(tokens

tokens = set([stemmer.stem(w) for w in txt.split()])
word_dict = set(word_dict)
not_word_tokens = tokens - tokens.intersection(word_dict)

print(f'After stemming! Tweets contain unknown words: {100 * len(not_word_toke

# list of non word tokens
list(not_word_tokens)[:15]
```

```
Tweets contain unknown words: 71.64%
After stemming! Tweets contain unknown words: 69.92%
```

Out[98]:
```
['ER',
 'rooftop',
 'urbanfashion',
 'apr',
 'lotz',
 'fortun',
 'shelbi',
 'hwy401',
 'joeybats19',
 'SD',
 '02',
 'sistera',
 'francisco',
 'themselv',
 'sunshin']
```

Unknown word exists due to:

- stemming errors
- name of places
- name of persons
- mis-spelling
- concatenated words
- abbreviations

DISASTER TWEETS CLEANING

In [99]:
```
!pip install beautifulsoup4
```

```
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.10/si
te-packages (4.12.2)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.10/sit
e-packages (from beautifulsoup4) (2.5)
```

In [100]:
```
!pip install ekphrasis
```

```
Requirement already satisfied: ekphrasis in /opt/conda/lib/python3.10/site-pa
ckages (0.5.4)
Requirement already satisfied: termcolor in /opt/conda/lib/python3.10/site-pa
ckages (from ekphrasis) (2.4.0)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.10/site-package
s (from ekphrasis) (4.66.1)
Requirement already satisfied: colorama in /opt/conda/lib/python3.10/site-pac
kages (from ekphrasis) (0.4.6)
Requirement already satisfied: ujson in /opt/conda/lib/python3.10/site-packag
es (from ekphrasis) (5.9.0)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.10/site-p
ackages (from ekphrasis) (3.7.5)
Requirement already satisfied: nltk in /opt/conda/lib/python3.10/site-package
s (from ekphrasis) (3.2.4)
Requirement already satisfied: ftfy in /opt/conda/lib/python3.10/site-package
s (from ekphrasis) (6.2.0)
Requirement already satisfied: numpy in /opt/conda/lib/python3.10/site-packag
es (from ekphrasis) (1.26.4)
Requirement already satisfied: wcwidth<0.3.0,>=0.2.12 in /opt/conda/lib/pytho
n3.10/site-packages (from ftfy->ekphrasis) (0.2.13)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.10/
site-packages (from matplotlib->ekphrasis) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.10/site
-packages (from matplotlib->ekphrasis) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.1
0/site-packages (from matplotlib->ekphrasis) (4.47.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.1
0/site-packages (from matplotlib->ekphrasis) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.10/s
ite-packages (from matplotlib->ekphrasis) (21.3)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.10/sit
e-packages (from matplotlib->ekphrasis) (9.5.0)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.10/
site-packages (from matplotlib->ekphrasis) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python
3.10/site-packages (from matplotlib->ekphrasis) (2.9.0.post0)
Requirement already satisfied: six in /opt/conda/lib/python3.10/site-packages
(from nltk->ekphrasis) (1.16.0)
```

In [101]:
```
!pip install spacy
```

```
Requirement already satisfied: spacy in /opt/conda/lib/python3.10/site-packag
es (3.7.3)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /opt/conda/lib/
python3.10/site-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /opt/conda/lib/
python3.10/site-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /opt/conda/lib/py
thon3.10/site-packages (from spacy) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /opt/conda/lib/python3.
10/site-packages (from spacy) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /opt/conda/lib/python
3.10/site-packages (from spacy) (3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in /opt/conda/lib/python3.
10/site-packages (from spacy) (8.2.2)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /opt/conda/lib/python
3.10/site-packages (from spacy) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /opt/conda/lib/python3.
10/site-packages (from spacy) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /opt/conda/lib/pyth
on3.10/site-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in /opt/conda/lib/python
3.10/site-packages (from spacy) (0.3.4)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /opt/conda/lib/python
3.10/site-packages (from spacy) (0.9.0)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /opt/conda/lib/pyt
hon3.10/site-packages (from spacy) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /opt/conda/lib/python3.
10/site-packages (from spacy) (4.66.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /opt/conda/lib/pyth
on3.10/site-packages (from spacy) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /opt/c
onda/lib/python3.10/site-packages (from spacy) (2.5.3)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.10/site-packa
ges (from spacy) (3.1.2)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.10/site-p
ackages (from spacy) (69.0.3)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.10/s
ite-packages (from spacy) (21.3)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /opt/conda/lib/pyth
on3.10/site-packages (from spacy) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in /opt/conda/lib/python3.10/sit
e-packages (from spacy) (1.26.4)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/lib/pyt
hon3.10/site-packages (from packaging>=20.0->spacy) (3.1.1)
Requirement already satisfied: annotated-types>=0.4.0 in /opt/conda/lib/pytho
n3.10/site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (0.6.
0)
Requirement already satisfied: pydantic-core==2.14.6 in /opt/conda/lib/python
3.10/site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (2.14.
6)
Requirement already satisfied: typing-extensions>=4.6.1 in /opt/conda/lib/pyt
hon3.10/site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (4.
9.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/pyt
hon3.10/site-packages (from requests<3.0.0,>=2.13.0->spacy) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site
-packages (from requests<3.0.0,>=2.13.0->spacy) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.1
0/site-packages (from requests<3.0.0,>=2.13.0->spacy) (1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.1
0/site-packages (from requests<3.0.0,>=2.13.0->spacy) (2024.2.2)
```

```
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /opt/conda/lib/python3.1
0/site-packages (from thinc<8.3.0,>=8.2.2->spacy) (0.7.10)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /opt/conda/lib/pyt
hon3.10/site-packages (from thinc<8.3.0,>=8.2.2->spacy) (0.1.4)
Requirement already satisfied: click<9.0.0,>=7.1.1 in /opt/conda/lib/python3.
10/site-packages (from typer<0.10.0,>=0.3.0->spacy) (8.1.7)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in /opt/conda/lib/
python3.10/site-packages (from weasel<0.4.0,>=0.1.0->spacy) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.10/s
ite-packages (from jinja2->spacy) (2.1.3)
```

Import Libs

In [102]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[102]:
```
True
```

In [103]:
```python
import requests, json
import string
import re
from itertools import chain

import numpy as np
import pandas as pd

from bs4 import BeautifulSoup

from ekphrasis.classes.preprocessor import TextPreProcessor
from ekphrasis.classes.tokenizer import SocialTokenizer
from ekphrasis.dicts.emoticons import emoticons
from ekphrasis.dicts.noslang.slangdict import slangdict

from nltk.corpus import words
from nltk.corpus import stopwords
import spacy

spacy.cli.download("en_core_web_sm")
nlp = spacy.load("en_core_web_sm")
en_words = words.words()
st_words = stopwords.words()
```

```
Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-models/releases/downloa
d/en_core_web_sm-3.7.1/en_core_web_sm-3.7.1-py3-none-any.whl (https://git
hub.com/explosion/spacy-models/releases/download/en_core_web_sm-3.7.1/en_
core_web_sm-3.7.1-py3-none-any.whl) (12.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 68.1 MB/s eta
0:00:0000:0100:01
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in /opt/conda/lib/pyth
on3.10/site-packages (from en-core-web-sm==3.7.1) (3.7.3)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /opt/conda/
lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==
3.7.1) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /opt/conda/
lib/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==
3.7.1) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /opt/conda/li
b/python3.10/site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.
7.1) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /opt/conda/lib/pyth
```

Read data

In [104]:
```
# training data : 'df' and testing data : 'test_df'

#df = pd.read_csv('/kaggle/input/nlp-getting-started/train.csv')
df = train_df
#test_df = pd.read_csv('/kaggle/input/nlp-getting-started/test.csv')
df_leak = pd.read_csv('/kaggle/input/pg-final-project-datasets/socialmedia-dis
df_leak['target'] = (df_leak['choose_one'] == 'Relevant').astype(np.int8)
df_leak['id'] = df_leak.index.astype(np.int16)
df_leak = df_leak[['target', 'id']]
test_df = test_df.merge(df_leak, on=['id'], how='left')

print(df.shape, test_df.shape)
```

```
(7503, 9) (3263, 5)
```

Util Functions

In [105]:
```
def get_unk_words(txts):
    txt = ' '.join(txts)
    doc = nlp(txt)
    tokens = set([tok.lemma_ for tok in doc])

    word_dict = set([tok.lower() for tok in en_words])
    not_word_tokens = tokens - tokens.intersection(word_dict)
    print('Vocabs size', len(tokens))
    print('Unknown vocabs size', len(not_word_tokens))
    print(f'Tweets contain unknown words: {100 * len(not_word_tokens) / len(tc
    return not_word_tokens
```

Chat Abbreviations

In [106]:

```
pairs =  list(slangdict.items())
for i in range(10):
    print(pairs[i])

uncased_slang_dict = {}
for key, value in slangdict.items():
    value = re.sub(r'it means', '', value)
    value = re.sub(r'\*\*\*', 'uck', value)
    value = re.sub(r'\*\*', 'it', value)
    value = re.sub(r'it refers to', '', value)
    uncased_slang_dict[key.lower()] = value.lower()

print()
print('Slang Words Count:', len(uncased_slang_dict))
print()

replacement_patterns = [
    (r'won\'t', 'will not'),
    (r'can\'t', 'cannot'),
    (r'i\'m', 'i am'),
    (r'I\'m', 'I am'),
    (r'ain\'t', 'is not'),
    (r'(\w+)\'ll', '\g<1> will'),
    (r'(\w+)n\'t', '\g<1> not'),
    (r'(\w+)\'ve', '\g<1> have'),
    (r'(\w+)\'s', '\g<1> is'),
    (r'(\w+)\'re', '\g<1> are'),
    (r'(\w+)\'d', '\g<1> would'),
]

abbr = {
    "$" : " dollar ",
    "€" : " euro ",
    "4ao" : "for adults only",
    "a.m" : "before midday",
    "a3" : "anytime anywhere anyplace",
    "aamof" : "as a matter of fact",
    "acct" : "account",
    "adih" : "another day in hell",
    "afaic" : "as far as i am concerned",
    "afaict" : "as far as i can tell",
    "afaik" : "as far as i know",
    "afair" : "as far as i remember",
    "afk" : "away from keyboard",
    "app" : "application",
    "approx" : "approximately",
    "apps" : "applications",
    "asap" : "as soon as possible",
    "asl" : "age, sex, location",
    "atk" : "at the keyboard",
    "ave." : "avenue",
    "aymm" : "are you my mother",
    "ayor" : "at your own risk",
    "b&b" : "bed and breakfast",
    "b+b" : "bed and breakfast",
    "b.c" : "before christ",
    "b2b" : "business to business",
    "b2c" : "business to customer",
    "b4" : "before",
    "b4n" : "bye for now",
    "b@u" : "back at you",
    "bae" : "before anyone else",
    "bak" : "back at keyboard",
    "bbbg" : "bye bye be good",
    "bbc" : "british broadcasting corporation",
    "bbias" : "be back in a second",
    "bbl" : "be back later",
    "bbs" : "be back soon",
    "be4" : "before",
    "bfn" : "bye for now",
    "blvd" : "boulevard",
    "bout" : "about",
    "brb" : "be right back",
    "bros" : "brothers",
    "brt" : "be right there",
    "bsaaw" : "big smile and a wink",
    "btw" : "by the way",
    "bwl" : "bursting with laughter",
    "c/o" : "care of",
    "cet" : "central european time",
    "cf" : "compare",
    "cia" : "central intelligence agency",
    "csl" : "can not stop laughing",
    "cu" : "see you",
    "cul8r" : "see you later",
    "cv" : "curriculum vitae",
    "cwot" : "complete waste of time",
    "cya" : "see you",
    "cyt" : "see you tomorrow",
    "dae" : "does anyone else",
    "dbmib" : "do not bother me i am busy",
    "diy" : "do it yourself",
    "dm" : "direct message",
    "dwh" : "during work hours",
    "e123" : "easy as one two three",
    "eet" : "eastern european time",
    "eg" : "example",
    "embm" : "early morning business meeting",
    "encl" : "enclosed",
    "encl." : "enclosed",
    "etc" : "and so on",
    "faq" : "frequently asked questions",
    "fawc" : "for anyone who cares",
    "fb" : "facebook",
    "fc" : "fingers crossed",
    "fig" : "figure",
    "fimh" : "forever in my heart",
    "ft." : "feet",
    "ft" : "featuring",
    "ftl" : "for the loss",
    "ftw" : "for the win",
    "fwiw" : "for what it is worth",
    "fyi" : "for your information",
    "g9" : "genius",
    "gahoy" : "get a hold of yourself",
    "gal" : "get a life",
    "gcse" : "general certificate of secondary education",
    "gfn" : "gone for now",
    "gg" : "good game",
    "gl" : "good luck",
    "glhf" : "good luck have fun",
    "gmt" : "greenwich mean time",
    "gmta" : "great minds think alike",
    "gn" : "good night",
    "g.o.a.t" : "greatest of all time",
    "goat" : "greatest of all time",
    "goi" : "get over it",
    "gps" : "global positioning system",
    "gr8" : "great",
    "gratz" : "congratulations",
    "gyal" : "girl",
    "h&c" : "hot and cold",
    "hp" : "horsepower",
    "hr" : "hour",
    "hrh" : "his royal highness",
    "ht" : "height",
    "ibrb" : "i will be right back",
    "ic" : "i see",
    "icq" : "i seek you",
    "icymi" : "in case you missed it",
    "idc" : "i do not care",
    "idgadf" : "i do not give a damn fuck",
    "idgaf" : "i do not give a fuck",
    "idk" : "i do not know",
    "ie" : "that is",
    "i.e" : "that is",
    "ifyp" : "i feel your pain",
    "IG" : "instagram",
    "iirc" : "if i remember correctly",
    "ilu" : "i love you",
    "ily" : "i love you",
    "imho" : "in my humble opinion",
    "imo" : "in my opinion",
    "imu" : "i miss you",
    "iow" : "in other words",
    "irl" : "in real life",
    "j4f" : "just for fun",
    "jic" : "just in case",
    "jk" : "just kidding",
    "jsyk" : "just so you know",
    "l8r" : "later",
    "lb" : "pound",
    "lbs" : "pounds",
    "ldr" : "long distance relationship",
    "lmao" : "laugh my ass off",
    "lmfao" : "laugh my fucking ass off",
    "lol" : "laughing out loud",
    "ltd" : "limited",
    "ltns" : "long time no see",
    "m8" : "mate",
    "mf" : "motherfucker",
    "mfs" : "motherfuckers",
    "mfw" : "my face when",
    "mofo" : "motherfucker",
    "mph" : "miles per hour",
    "mr" : "mister",
    "mrw" : "my reaction when",
    "ms" : "miss",
    "mte" : "my thoughts exactly",
    "nagi" : "not a good idea",
    "nbc" : "national broadcasting company",
    "nbd" : "not big deal",
    "nfs" : "not for sale",
    "ngl" : "not going to lie",
    "nhs" : "national health service",
    "nrn" : "no reply necessary",
    "nsfl" : "not safe for life",
    "nsfw" : "not safe for work",
    "nth" : "nice to have",
    "nvr" : "never",
    "nyc" : "new york city",
    "oc" : "original content",
    "og" : "original",
    "ohp" : "overhead projector",
    "oic" : "oh i see",
    "omdb" : "over my dead body",
    "omg" : "oh my god",
    "omw" : "on my way",
    "p.a" : "per annum",
    "p.m" : "after midday",
    "pm" : "prime minister",
    "poc" : "people of color",
    "pov" : "point of view",
    "pp" : "pages",
    "ppl" : "people",
    "prw" : "parents are watching",
    "ps" : "postscript",
    "pt" : "point",
    "ptb" : "please text back",
    "pto" : "please turn over",
    "qpsa" : "what happens", #"que pasa",
    "ratchet" : "rude",
    "rbtl" : "read between the lines",
    "rlrt" : "real life retweet",
    "rofl" : "rolling on the floor laughing",
    "roflol" : "rolling on the floor laughing out loud",
    "rotflmao" : "rolling on the floor laughing my ass off",
    "rt" : "retweet",
    "ruok" : "are you ok",
    "sfw" : "safe for work",
    "sk8" : "skate",
    "smh" : "shake my head",
    "sq" : "square",
    "srsly" : "seriously",
    "ssdd" : "same stuff different day",
    "tbh" : "to be honest",
    "tbs" : "tablespooful",
    "tbsp" : "tablespooful",
    "tfw" : "that feeling when",
    "thks" : "thank you",
    "tho" : "though",
    "thx" : "thank you",
    "tia" : "thanks in advance",
    "til" : "today i learned",
    "tl;dr" : "too long i did not read",
    "tldr" : "too long i did not read",
    "tmb" : "tweet me back",
    "tntl" : "trying not to laugh",
    "ttyl" : "talk to you later",
    "u" : "you",
    "u2" : "you too",
    "u4e" : "yours for ever",
    "utc" : "coordinated universal time",
    "w/" : "with",
    "w/o" : "without",
    "w8" : "wait",
    "wassup" : "what is up",
    "wb" : "welcome back",
    "wtf" : "what the fuck",
    "wtg" : "way to go",
    "wtpa" : "where the party at",
    "wuf" : "where are you from",
    "wuzup" : "what is up",
    "wywh" : "wish you were here",
    "yd" : "yard",
    "ygtr" : "you got that right",
    "ynk" : "you never know",
    "zzz" : "sleeping bored and tired",
    "yr": "year",
    "u.s":"usa",
}

def replace_slang(txt, slang):
    ctxt = re.sub(r'\s+', ' ', txt)
    res = []
    for tok in ctxt.split():
        if tok.lower() in slang:
            res.append(slang[tok.lower()])
        else:
            res.append(tok)
    res = ' '.join(res)
    return res.strip()

sent = 'I want to go aamof home'
print(sent)
print(replace_slang(sent, abbr))
```

```
('*4u', 'Kiss for you')
('*67', 'unknown')
('*eg*', 'evil grin')
('07734', 'hello')
('0day', 'software illegally obtained before it was released')
('0noe', 'Oh No')
('0vr', 'over')
('10q', 'thank you')
('10tacle', 'tentacle')
('10x', 'thanks')

Slang Words Count: 5429

I want to go aamof home
I want to go as a matter of fact home
```

Define text preprocessor

- extract emojis
- replace numbers/date/money
- extract hashtags
- correct enlongated/repeated character

In [107]:
```
text_processor = TextPreProcessor(
    # terms that will be normalized
    normalize = ['rest_emoticons', 'rtl_face', 'cashtag','url',
                 'email', 'percent', 'money', 'phone', 'user',
                 'time', 'date', 'number', 'eastern_emoticons'],
    # terms that will be annotated
    annotate = set(["elongated", "repeated"]),
    fix_html = True,  # fix HTML tokens
    segmenter="twitter",
    corrector = "twitter",
    unpack_hashtags = True,  # perform word segmentation on hashtags
    unpack_contractions = True,  # Unpack contractions (can't -> can not)
    spell_correct_elong = False,  # spell correction for elongated words
    tokenizer=SocialTokenizer(lowercase=False).tokenize,
    dicts=[emoticons]
)
```

```
Reading twitter - 1grams ...
Reading twitter - 2grams ...
Reading twitter - 1grams ...
```

Text Cleaning

In [108]:
```
def preprocess(txt):
    # remove non-ascii characters
    res = txt.encode('ascii', 'ignore').decode()
    # replace slang token if the token is not an english word
    res = replace_slang(res, uncased_slang_dict)
    # replace shorten pattern i.e I'll--> I will
    for patt, rep in replacement_patterns:
        res = re.sub(patt, rep, res)
    # Extract emojis and hashtags and segment the txt
    res = ' '.join(text_processor.pre_process_doc(res)).strip()
    for patt in [r"<elongated>", r"<repeated>"]:
        res = re.sub(patt, '', res)

    # another try to replace the slangs after segmentation
    res = replace_slang(res, uncased_slang_dict)

    # remove punctuaions
    res = re.sub(r'[%s]' % re.escape(''.join(string.punctuation)), r' ',res)
    # lower case
    res = res.lower()
    # remove consecutive duplicated tokens
    res = re.sub(r'\b(\w+)(?:\W+\1\b)+', r'\1', res)
    #remove extra spaces
    res = re.sub(r'\s+', ' ', res)
    return res.strip()
```

In [109]:
```
df['ctext'] = df['text'].apply(preprocess)
test_df['ctext'] = test_df['text'].apply(preprocess)

# have a look at clean dataset

display(df.iloc[35:40])
display(test_df['ctext'].iloc[35:40])
```

	text	id	location	keyword	target	ctext	url	hashtag_num	hasht
35	#BBSNews latest 4 #Palestine &amp; #Israel - ...	3337	USA	demolished	1	be back soon news latest number palestine isra...	1	3	[is bbsn pales
36	#BBShelli seems pretty sure she's the one that...	9971	Louavul, KY	tsunami	0	bye shelli seems pretty sure she is the one th...	0	2	[bbsl b
37	#BHRAMABULL Watch Run The Jewels Use Facts to ...	8262	tri state	rioting	1	bhramabull watch run the jewels use facts to d...	1	1	[bhrama
38	#BREAKING 10th death confirmed in Legionnaires...	7594	Pro-American and Anti-#Occupy	outbreak	1	breaking 1 0 th death confirmed in legionnaire...	1	1	[break
39	#BREAKING411 4 police officers arrested for ab...	7748	New York, NY	police	1	breaking 411 number police officers arrested f...	1	1	[breaking

```
35    user if you pretend to feel a certain way the ...
36    for legal and medical referral service user ca...
37    there is a construction guy working on the dis...
38    user i feel like i am going to do it on accide...
39    on the m 42 northbound between junctions j3 an...
Name: ctext, dtype: object
```

Pre-Processing Results

In [110]:
```
for txt, ctxt in df[['text', 'ctext']].values[0:10]:
    print('Plain tweet text: ',txt)
    print()
    print('Cleaned tweet text: ', ctxt)
    print()

print('Count of unknown words in train dataset')
not_word_tokens = get_unk_words(df['ctext'])
print()
print('Count of unknown words in test dataset')
not_word_tokens = get_unk_words(test_df['ctext'])
```

```
Plain tweet text:  ! Residents Return To Destroyed Homes As Washington Wildfi
re Burns on http://t.co/UcI8stQUg1 (http://t.co/UcI8stQUg1)

Cleaned tweet text:  residents return to destroyed homes as washington wildfi
re burns on url

Plain tweet text:  # handbags Genuine Mulberry Antony Cross Body Messenger Ba
g Dark Oak Soft Buffalo Leather:  å£279.00End Date: W... http://t.co/FTM4RKl8
mN (http://t.co/FTM4RKl8mN)

Cleaned tweet text:  handbags genuine mulberry antony cross body messenger ba
g dark oak soft buffalo leather number 0 end date w url

Plain tweet text:  #360WiseNews : China's Stock Market Crash: Are There Gems
In The Rubble? http://t.co/9Naw3QOQOL (http://t.co/9Naw3QOQOL)

Cleaned tweet text:  360 wise news china is stock market crash are there gems
in the rubble url

Plain tweet text:  #360WiseNews : China's Stock Market Crash: Are There Gems
In The Rubble? http://t.co/aOd2ftBMGU (http://t.co/aOd2ftBMGU)

Cleaned tweet text:  360 wise news china is stock market crash are there gems
in the rubble url

Plain tweet text:  #360WiseNews : China's Stock Market Crash: Are There Gems
In The Rubble? http://t.co/eaTFro3d5x (http://t.co/eaTFro3d5x)

Cleaned tweet text:  360 wise news china is stock market crash are there gems
in the rubble url

Plain tweet text:  #360WiseNews : China's Stock Market Crash: Are There Gems
In The Rubble? http://t.co/gQskwqZuUl (http://t.co/gQskwqZuUl)

Cleaned tweet text:  360 wise news china is stock market crash are there gems
in the rubble url

Plain tweet text:  #3: TITAN WarriorCord 100 Feet - Authentic Military 550 Pa
racord - MIL-C-5040-H Type III 7 Strand 5/16' di... http://t.co/EEjRMKtJ0R (h
ttp://t.co/EEjRMKtJ0R)

Cleaned tweet text:  number titan warriorcord number feet authentic military
number paracord mil c number h type i number strand number di url

Plain tweet text:  #3Novices : Renison mine sees seismic event http://t.co/2i
4EOGGO5j (http://t.co/2i4EOGGO5j) A small earthquake at Tasmania's Renison ti
n project has created a tem�Û_

Cleaned tweet text:  3 novices renison mine sees seismic event url a small ea
rthquake at tasmania is renison tin project has created a tem

Plain tweet text:  #4: The Hobbit: The Desolation of Smaug (Bilingual) htt
p://t.co/G5dO2X6226 (http://t.co/G5dO2X6226)

Cleaned tweet text:  number the hobbit the desolation of smaug bilingual url

Plain tweet text:  #?? #?? #??? #??? MH370: Aircraft debris found on La Reuni
on is from missing Malaysia Airlines ...  http://t.co/MRVXBZywd4 (http://t.c
o/MRVXBZywd4)

Cleaned tweet text:  mh370 aircraft debris found on la reunion is from missin
```

```
g malaysia airlines url

Count of unknown words in train dataset
Vocabs size 11303
Unknown vocabs size 4135
Tweets contain unknown words: 36.58%

Count of unknown words in test dataset
Vocabs size 7134
Unknown vocabs size 2213
Tweets contain unknown words: 31.02%
```

Unknown Words

In [111]:
```
for tok in list(not_word_tokens)[0:20]:
    print(tok)
```

```
holmes
euro
shania
jd
amerika
trad
naija
midfielder
fil
radcliff
efak
au
tahoe
ketep
rnb
yazidis
darker
rooney
rousey
monitoring
```

Remove duplicated texts

In [112]:
```
df = df.fillna('unk')
test_df = test_df.fillna('unk')

df = df.groupby(by = ['ctext']).agg({
    'id': 'first',
    'location': lambda x:x.value_counts().index[0],
    'keyword':lambda x:x.value_counts().index[0],
    'target': lambda x:x.value_counts().index[0],
    'text': lambda x:x.value_counts().index[0],
}).reset_index()

print(df.shape)
```

```
(6861, 6)
```

Clean column keywords

- remove strange characters
- replace words with their lemma

In [113]:
```
def preprocess_keywords(txt):
    res = txt.lower()
    res = re.sub(r'[^a-zA-Z]', r' ', res)
    res = re.sub(r'\s+', r' ', res)
    doc = nlp(res)
    res = ' '.join([token.lemma_ for token in doc])
    res = re.sub(r'\s+', r' ', res)
    return res.strip()
```

In [114]:
```
keyword = set(df['keyword'].values)
keyword = {
    key: preprocess_keywords(key).lower() for key in keyword
}

df['ckeyword'] = df['keyword'].apply(lambda txt: keyword[txt])
test_df['ckeyword'] = test_df['keyword'].apply(lambda txt: keyword[txt])
```

In [115]:
```
df[df['keyword']!='unk'][['keyword', 'ckeyword']]  # print valid keywords whic

ids = [328,443,513,2619,3640,3900,4342,5781,6552,6554,6570,6701,6702,6729,6861
df.loc[df['id'].isin(ids), 'target'] = 0

df.to_csv('df.csv', index = False)
test_df.to_csv('test_df.csv', index = False)
```

Model Training

1. LSTM MODEL

In [116]:
```
import tensorflow as tf
from tensorflow.keras import layers

X = df['ctext'].to_numpy()
y = df['target'].to_numpy()

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,
                                                 random_state=42,
                                                 test_size=0.2)
```

In [117]:
```
import tensorflow as tf
from tensorflow.keras import layers

MAX_TOKENS = 20_000
EMBEDDING_SIZE = 64

# Assuming X contains your input data
text_vectorizer = layers.TextVectorization(max_tokens=MAX_TOKENS)
text_vectorizer.adapt(X)

inputs = layers.Input(shape=(1,), dtype=tf.string)
x = text_vectorizer(inputs)
x = layers.Embedding(MAX_TOKENS, EMBEDDING_SIZE)(x)
x = layers.Bidirectional(layers.LSTM(64, activation='relu', return_sequences=T
x = layers.LSTM(32, activation='relu')(x)
outputs = layers.Dense(1, activation='sigmoid')(x)

model = tf.keras.Model(inputs=inputs, outputs=outputs)
```

In [118]:
```
import tensorflow as tf
model = tf.keras.models.Model(inputs, outputs)

model.summary()
model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics = ['accuracy'])
history = model.fit(X_train,y_train,
          validation_data = (X_test,y_test),
          epochs = 10)

print('Average Accuracy of LSTM Model :')
model.evaluate(X_test,y_test)
```

```
Model: "functional_11"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ input_layer_2 (InputLayer)      │ (None, 1)              │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ text_vectorization_2            │ (None, None)           │             0 │
│ (TextVectorization)             │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ embedding_2 (Embedding)         │ (None, None, 64)       │     1,280,000 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ bidirectional_2 (Bidirectional) │ (None, None, 128)      │        66,048 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm_5 (LSTM)                   │ (None, 32)             │        20,608 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 1)              │            33 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 1,366,689 (5.21 MB)

 Trainable params: 1,366,689 (5.21 MB)

 Non-trainable params: 0 (0.00 B)
```

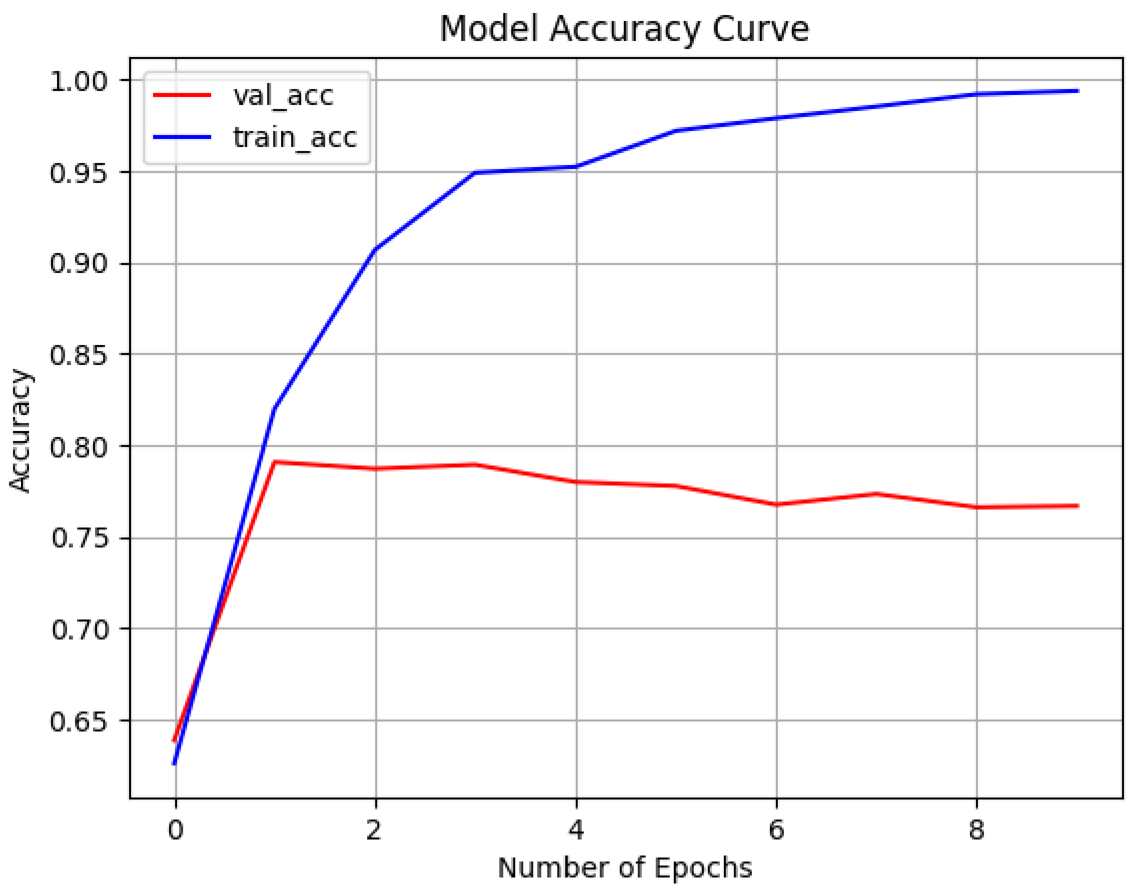
```
Epoch 1/10
172/172 ━━━━━━━━━━━━━━━━━━━━ 21s 99ms/step - accuracy: 0.5974 - loss: 0.6769
- val_accuracy: 0.6387 - val_loss: 0.6037
Epoch 2/10
172/172 ━━━━━━━━━━━━━━━━━━━━ 16s 94ms/step - accuracy: 0.7779 - loss: 0.4477
- val_accuracy: 0.7910 - val_loss: 0.4508
Epoch 3/10
172/172 ━━━━━━━━━━━━━━━━━━━━ 16s 95ms/step - accuracy: 0.9090 - loss: 0.2310
- val_accuracy: 0.7873 - val_loss: 0.5217
Epoch 4/10
172/172 ━━━━━━━━━━━━━━━━━━━━ 16s 94ms/step - accuracy: 0.9561 - loss: 0.1285
- val_accuracy: 0.7895 - val_loss: 0.9239
Epoch 5/10
172/172 ━━━━━━━━━━━━━━━━━━━━ 16s 96ms/step - accuracy: 0.9610 - loss: 0.1303
- val_accuracy: 0.7800 - val_loss: 0.8285
Epoch 6/10
172/172 ━━━━━━━━━━━━━━━━━━━━ 16s 93ms/step - accuracy: 0.9739 - loss: 0.0805
- val_accuracy: 0.7779 - val_loss: 1.2439
Epoch 7/10
172/172 ━━━━━━━━━━━━━━━━━━━━ 16s 95ms/step - accuracy: 0.9796 - loss: 0.0845
- val_accuracy: 0.7677 - val_loss: 18.2424
Epoch 8/10
172/172 ━━━━━━━━━━━━━━━━━━━━ 16s 94ms/step - accuracy: 0.9853 - loss: 0.1086
- val_accuracy: 0.7735 - val_loss: 2.2065
Epoch 9/10
172/172 ━━━━━━━━━━━━━━━━━━━━ 16s 95ms/step - accuracy: 0.9940 - loss: 0.0221
- val_accuracy: 0.7662 - val_loss: 1.9865
Epoch 10/10
172/172 ━━━━━━━━━━━━━━━━━━━━ 16s 93ms/step - accuracy: 0.9967 - loss: 0.0131
- val_accuracy: 0.7669 - val_loss: 2.1125
Average Accuracy of LSTM Model :
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.7867 - loss: 1.8330
```

```
Out[118]: [2.1124584674835205, 0.7669337391853333]
```

Visualisation

In [119]:
```
# import matplotlib.pyplot as plt
plt.plot(history.history['val_accuracy'],'r',label='val_acc')
plt.plot(history.history['accuracy'],'b',label='train_acc')
plt.title("Model Accuracy Curve")
plt.ylabel("Accuracy")
plt.xlabel("Number of Epochs")
plt.grid()
plt.legend()
```

Out[119]:
```
<matplotlib.legend.Legend at 0x78233700f730>
```



Model Testing

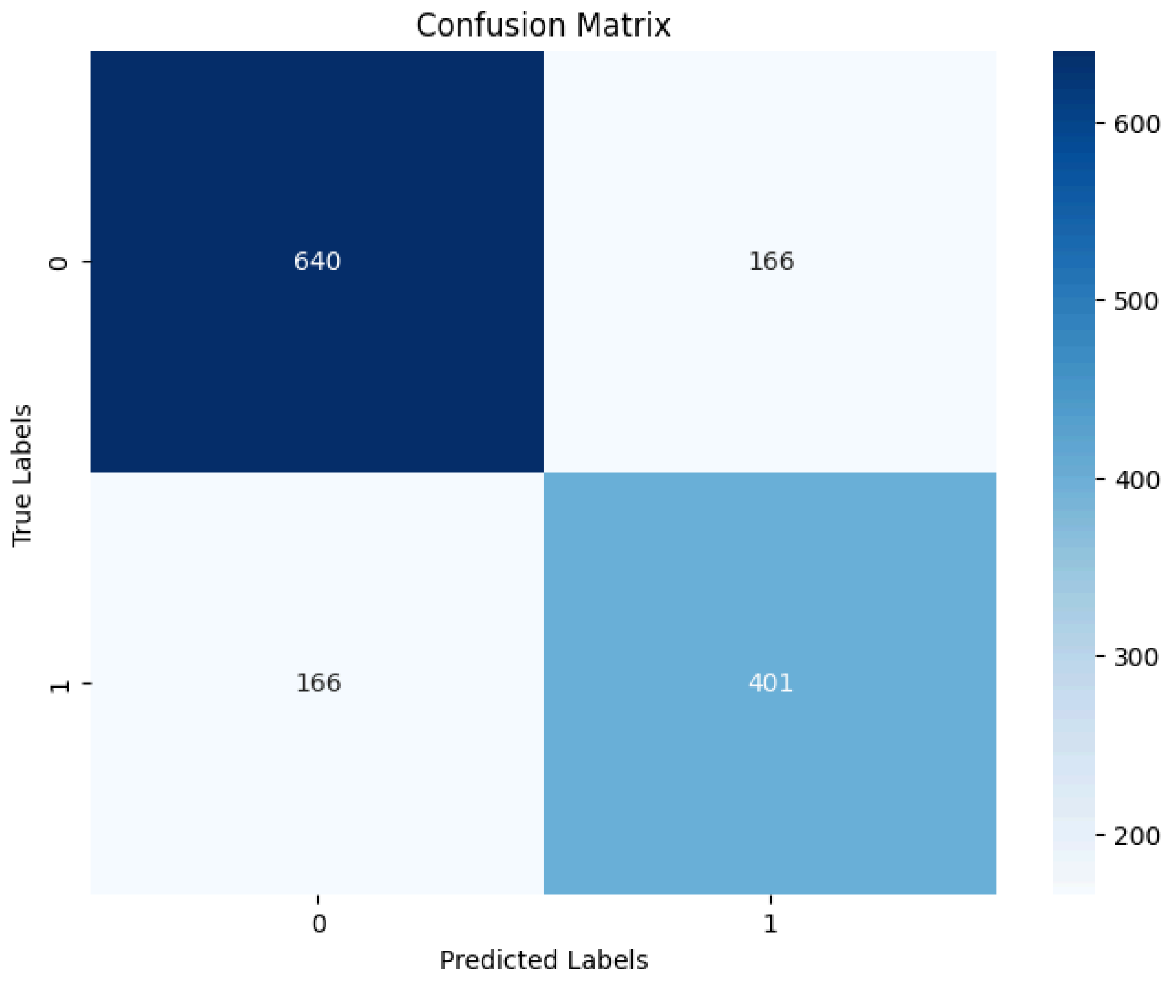
In [120]:
```
#Model prediction on TC

x_test = test_df['ctext'].to_numpy()

#display(test_df.iloc[:25])
#print(x_test)

y_actual = test_df['target'].to_numpy()
print("Actual target values : ")
print(y_actual[:25])
print()
preds = model.predict(test_df['ctext'])
preds = tf.squeeze(tf.round(preds))

submission = pd.read_csv('/kaggle/input/pg-final-project-datasets/sample_submi

submission['target'] = preds.numpy().astype(int)

#submission.head()
print('Predicted Values by the model : ')
display(submission.iloc[:25])
```

```
Actual target values :
[1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0]

102/102 ━━━━━━━━━━━━━━━━━━━━ 4s 34ms/step
Predicted Values by the model :
```

	id	target
0	0	1
1	2	1
2	3	1
3	9	1
4	11	1
5	12	1
6	21	0
7	22	0
8	27	0
9	29	0
10	30	0
11	35	0
12	42	0
13	43	0
14	45	0
15	46	1
16	47	0
17	51	1
18	58	0
19	60	0
20	69	0
21	70	0
22	72	0
23	75	1
24	84	0

In [125]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_
import seaborn as sns
import matplotlib.pyplot as plt
# Calculate evaluation metrics
test_accuracy = accuracy_score(y_test, y_pred)
print("Test Accuracy:", test_accuracy)

precision = precision_score(y_test, y_pred)
print("Precision:", precision)

recall = recall_score(y_test, y_pred)
print("Recall:", recall)

f1 = f1_score(y_test, y_pred)
print("F1-score:", f1)

# Generate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

```
Test Accuracy: 0.7581937363437727
Precision: 0.7072310405643739
Recall: 0.7072310405643739
F1-score: 0.7072310405643738
```


Confusion Matrix
True Labels
Predicted Labels
0
1
0
1
640
166
166
401
600
500
400
300
200

```
In [ ]:
```

```
In [ ]:
```