Problem Description

Twitter has become an important communication channel in times of emergency. The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programatically monitoring Twitter (i.e. disaster relief organizations and news agencies).

Dataset:- https://www.kaggle.com/c/nlp-getting-started/overview

Problem Statement

to build a predictive model to predict if disaster is real or fake

Real world/Business Objectives and constraints

Objectives:

1. predict if disaster is real or fake.
2. increase the accuracy

Mapping the real world problem to a Machine Learning Problem

Type of Machine Learning Problem

```
1)The given problem is a classification problem
```

Performance metric

```
1) accuracy
```

In [1]:

```
! pip install transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.6/dist-packages (4.1.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.6/dist-packages (from
transformers) (3.0.12)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.6/dist-packages (from
transformers) (4.41.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from transformers)
(1.19.4)
Requirement already satisfied: dataclasses; python_version < "3.7" in
/usr/local/lib/python3.6/dist-packages (from transformers) (0.8)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from
transformers) (2.23.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.6/dist-packages (from
transformers) (20.8)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.6/dist-packages (from
transformers) (2019.12.20)
Requirement already satisfied: tokenizers==0.9.4 in /usr/local/lib/python3.6/dist-packages (from
transformers) (0.9.4)
Requirement already satisfied: sacremoses in /usr/local/lib/python3.6/dist-packages (from
transformers) (0.0.43)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from
requests->transformers) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from
requests->transformers) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from
```

```
requests->transformers) (2020.12.5)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.6/dist-packages (from requests->transformers) (1.24.3)
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.6/dist-packages (from
packaging->transformers) (2.4.7)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from sacremoses-
>transformers) (1.15.0)
Requirement already satisfied: click in /usr/local/lib/python3.6/dist-packages (from sacremoses-
>transformers) (7.1.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.6/dist-packages (from sacremoses->
transformers) (1.0.0)
```

In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

In [3]:

```
import gc
import re
import string
import operator
from collections import defaultdict

import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")

import matplotlib.pyplot as plt
import seaborn as sns
from transformers import BertTokenizer
from wordcloud import STOPWORDS

from sklearn.model_selection import StratifiedKFold, StratifiedShuffleSplit
from sklearn.metrics import precision_score, recall_score, f1_score

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.layers import Dense, Input, Dropout, GlobalAveragePooling1D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, Callback

SEED = 1337
```

In [ ]:

```
!ls "drive/MyDrive/Colab Notebooks/Amazon"
```

```
BERT_Finetuning.ipynb          test.csv
nlp-with-disaster-tweets-eda-cleaning-and-bert.ipynb  train.csv
sample_submission.csv          Untitled.ipynb
```

In [4]:

```
import pandas as pd
train = pd.read_csv('drive/MyDrive/Colab Notebooks/Amazon/train.csv')
test = pd.read_csv('drive/MyDrive/Colab Notebooks/Amazon/test.csv')

print('Training Set Shape = {}'.format(train.shape))
print('Test Set Shape = {}'.format(test.shape))
```

```
Training Set Shape = (7613, 5)
Test Set Shape = (3263, 4)
```

In [ ]:

```
train.head(10)
```

Out[ ]:

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1
5	8	NaN	NaN	#RockyFire Update => California Hwy. 20 closed...	1
6	10	NaN	NaN	#flood #disaster Heavy rain causes flash flood...	1
7	13	NaN	NaN	I'm on top of the hill and I can see a fire in...	1
8	14	NaN	NaN	There's an emergency evacuation happening now ...	1
9	15	NaN	NaN	I'm afraid that the tornado is coming to our a...	1

Exploratory Data Analysis

Keyword and Location

Missing Values

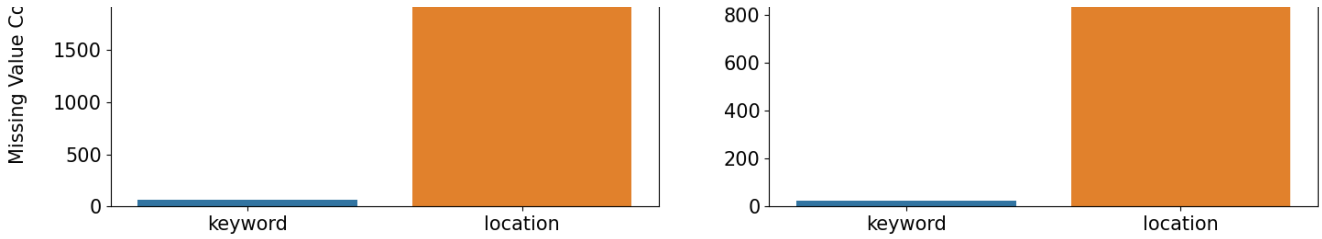
Both training and test set have same ratio of missing values in `keyword` and `location`.

- **0.8%** of `keyword` is missing in both training and test set
- **33%** of `location` is missing in both training and test set

Since missing value ratios between training and test set are too close, **they are most probably taken from the same sample**. Missing values in those features are filled with `no_keyword` and `no_location` respectively.

In [ ]:

```
missing_cols = ['keyword', 'location']

fig, axes = plt.subplots(ncols=2, figsize=(17, 4), dpi=100)

sns.barplot(x=train[missing_cols].isnull().sum().index, y=train[missing_cols].isnull().sum().values
, ax=axes[0])
sns.barplot(x=test[missing_cols].isnull().sum().index, y=test[missing_cols].isnull().sum().values,
ax=axes[1])

axes[0].set_ylabel('Missing Value Count', size=15, labelpad=20)
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('Training Set', fontsize=13)
axes[1].set_title('Test Set', fontsize=13)

plt.show()

for df in [train, test]:
    for col in ['keyword', 'location']:
        df[col] = df[col].fillna(f'no_{col}')
```



Cardinality and Target Distribution

Locations are not automatically generated, they are user inputs. That's why `location` is very dirty and there are too many unique values in it. It shouldn't be used as a feature.

Fortunately, there is signal in `keyword` because some of those words can only be used in one context. Keywords have very different tweet counts and target means. `keyword` can be used as a feature by itself or as a word added to the text. Every single keyword in training set exists in test set. If training and test set are from the same sample, it is also possible to use target encoding on `keyword`.

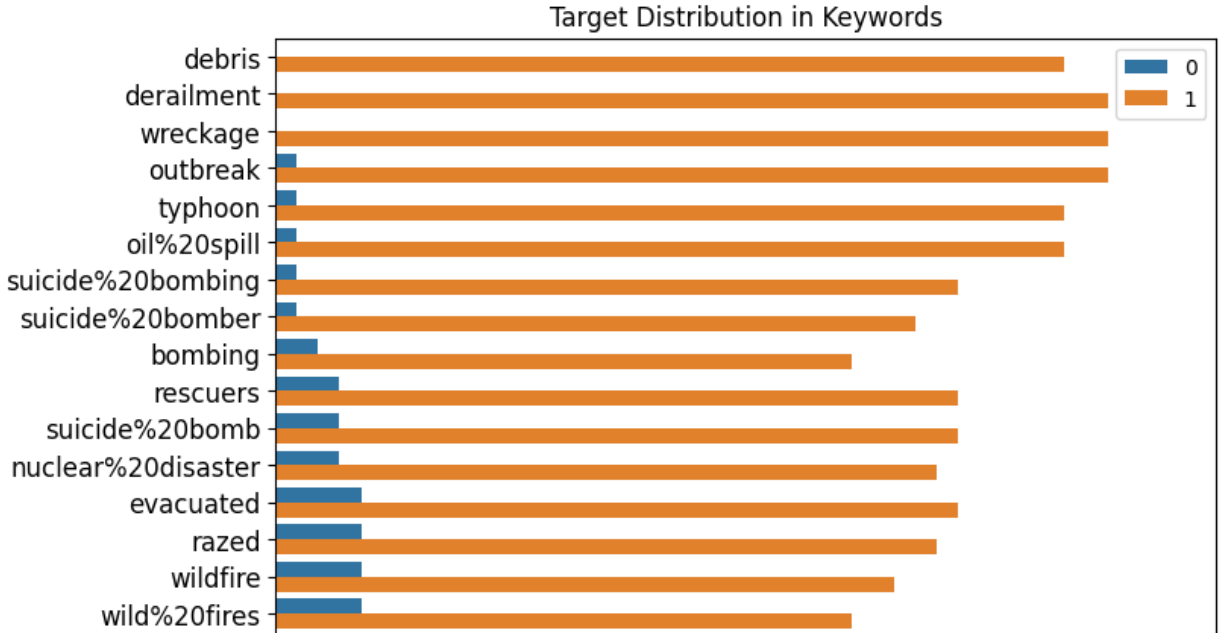
In [ ]:

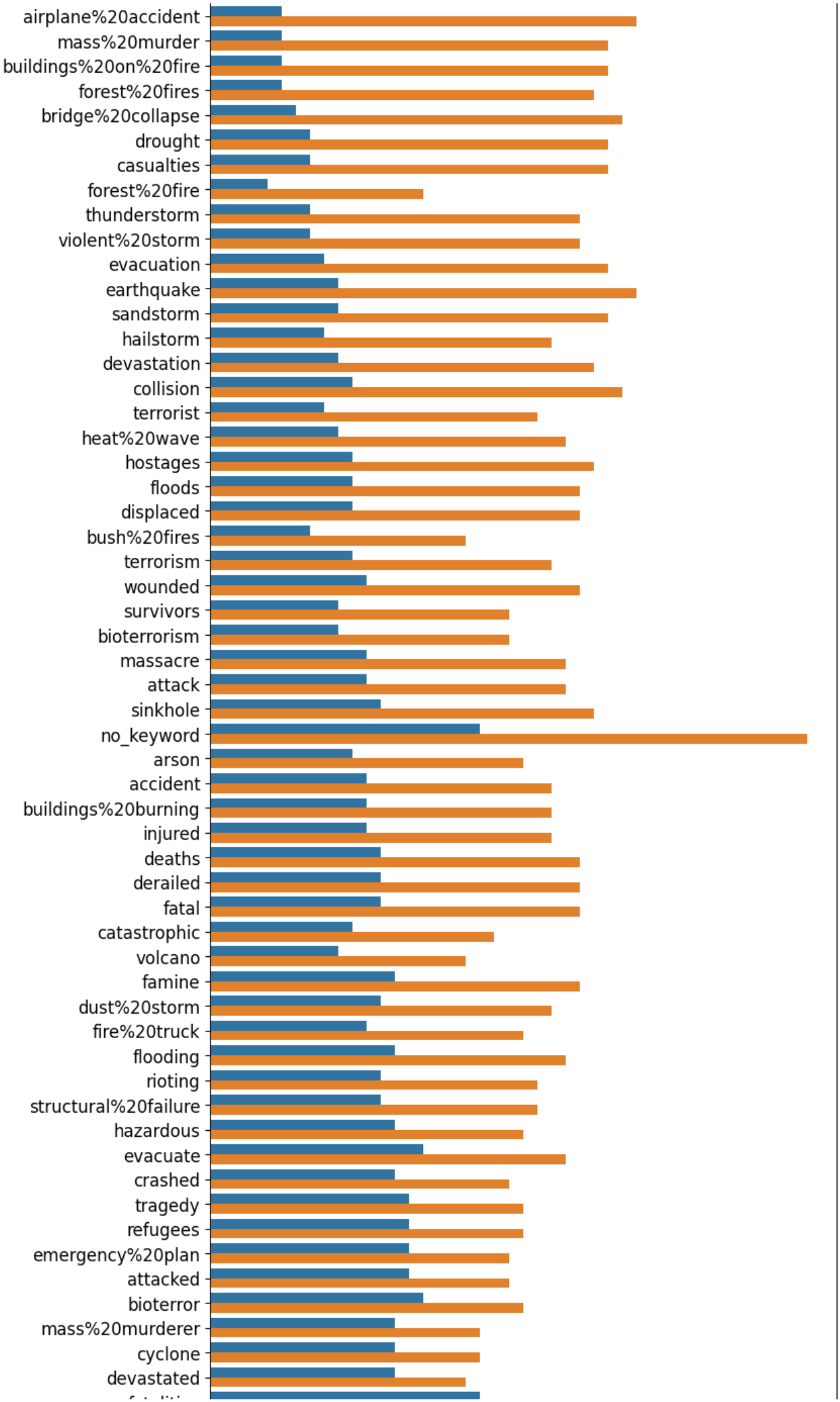
```
print(f'Number of unique values in keyword = {train["keyword"].nunique()} (Training) - {test["keyword"].nunique()} (Test)')
print(f'Number of unique values in location = {train["location"].nunique()} (Training) - {test["location"].nunique()} (Test)')
```

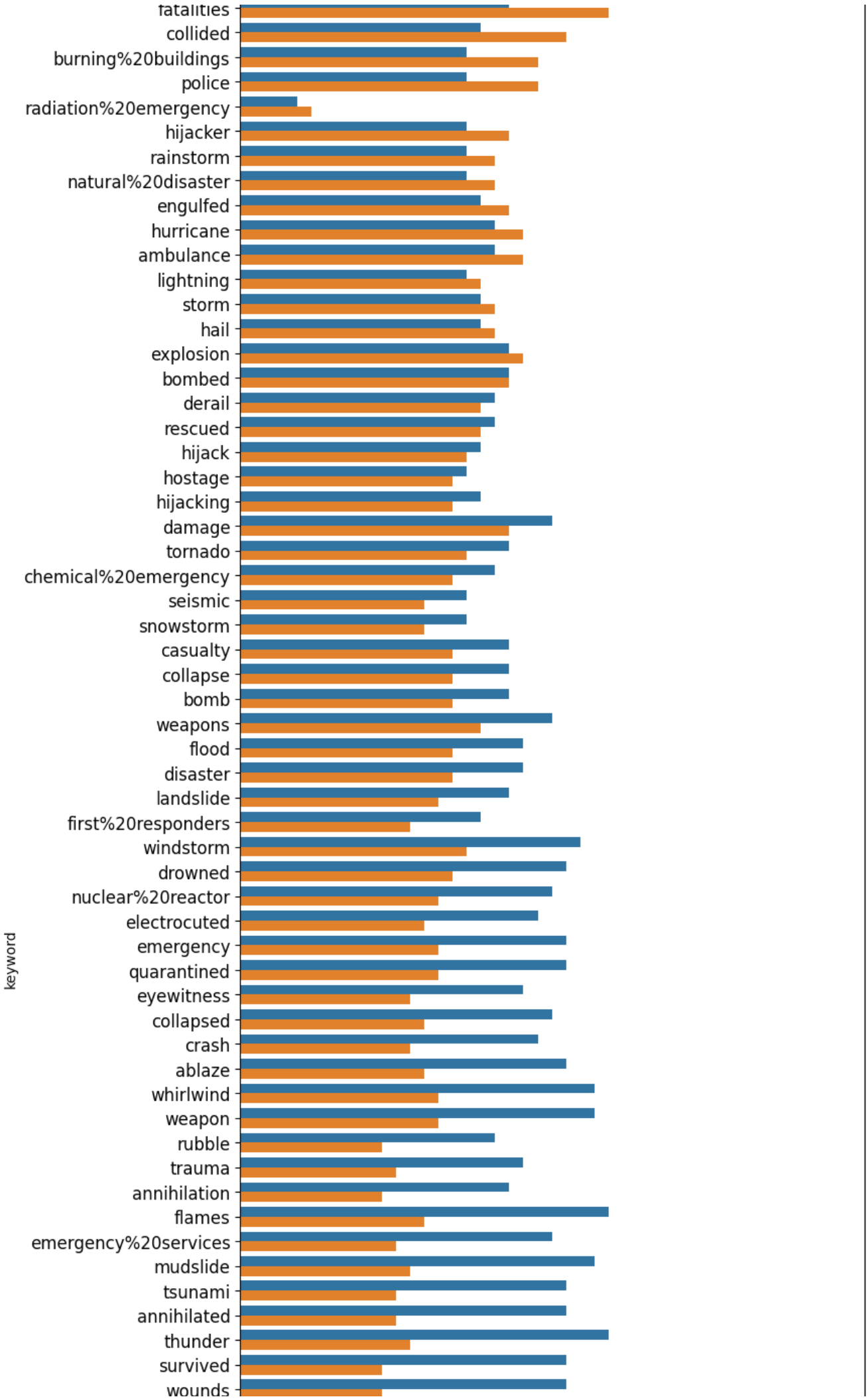
```
Number of unique values in keyword = 222 (Training) - 222 (Test)
Number of unique values in location = 3342 (Training) - 1603 (Test)
```

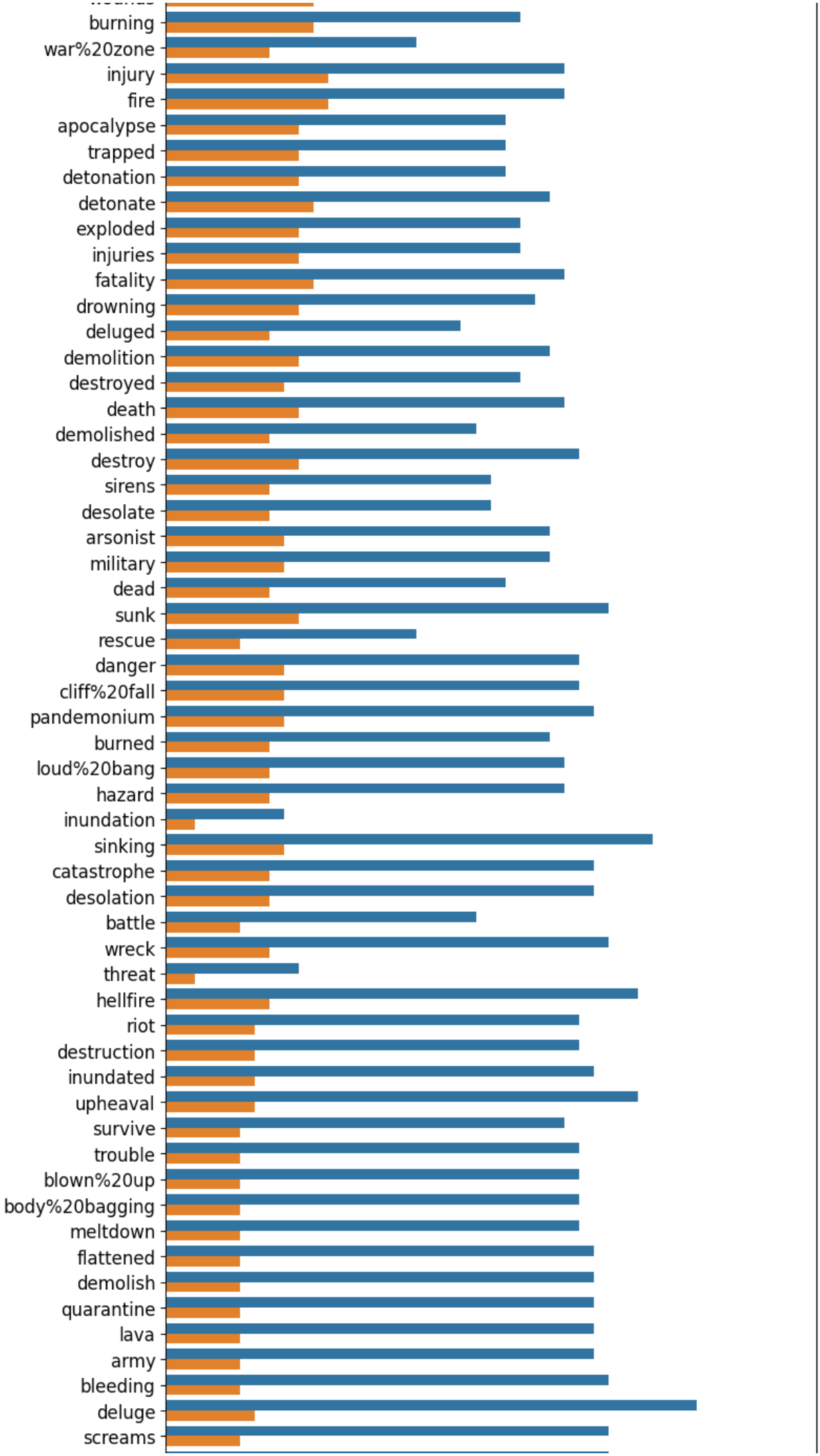
In [ ]:

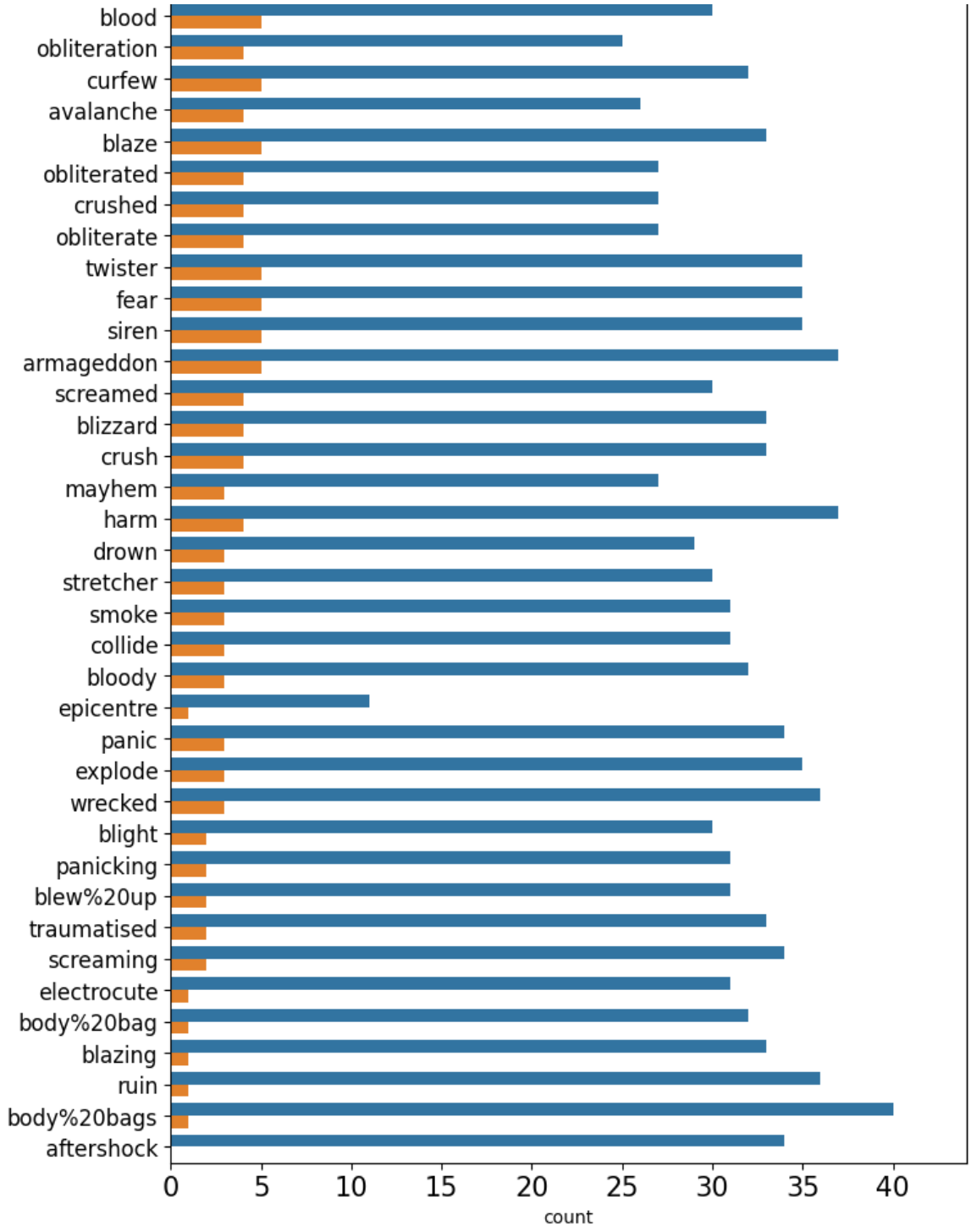
```
train['target_mean'] = train.groupby('keyword')['target'].transform('mean')

fig = plt.figure(figsize=(8, 72), dpi=100)

sns.countplot(y=train.sort_values(by='target_mean', ascending=False)['keyword'],
              hue=train.sort_values(by='target_mean', ascending=False)['target'])

plt.tick_params(axis='x', labelsize=15)
plt.tick_params(axis='y', labelsize=12)
plt.legend(loc=1)
plt.title('Target Distribution in Keywords')

plt.show()

train.drop(columns=['target_mean'], inplace=True)
```




airplane%20accident
mass%20murder
buildings%20on%20fire
forest%20fires
bridge%20collapse
drought
casualties
forest%20fire
thunderstorm
violent%20storm
evacuation
earthquake
sandstorm
hailstorm
devastation
collision
terrorist
heat%20wave
hostages
floods
displaced
bush%20fires
terrorism
wounded
survivors
bioterrorism
massacre
attack
sinkhole
no_keyword
arson
accident
buildings%20burning
injured
deaths
derailed
fatal
catastrophic
volcano
famine
dust%20storm
fire%20truck
flooding
rioting
structural%20failure
hazardous
evacuate
crashed
tragedy
refugees
emergency%20plan
attacked
bioterror
mass%20murderer
cyclone
devastated


fatalities
collided
burning%20buildings
police
radiation%20emergency
hijacker
rainstorm
natural%20disaster
engulfed
hurricane
ambulance
lightning
storm
hail
explosion
bombed
derail
rescued
hijack
hostage
hijacking
damage
tornado
chemical%20emergency
seismic
snowstorm
casualty
collapse
bomb
weapons
flood
disaster
landslide
first%20responders
windstorm
drowned
nuclear%20reactor
electrocuted
emergency
quarantined
eyewitness
collapsed
crash
ablaze
whirlwind
weapon
rubble
trauma
annihilation
flames
emergency%20services
mudslide
tsunami
annihilated
thunder
survived
wounds
keyword


burning
war%20zone
injury
fire
apocalypse
trapped
detonation
detonate
exploded
injuries
fatality
drowning
deluged
demolition
destroyed
death
demolished
destroy
sirens
desolate
arsonist
military
dead
sunk
rescue
danger
cliff%20fall
pandemonium
burned
loud%20bang
hazard
inundation
sinking
catastrophe
desolation
battle
wreck
threat
hellfire
riot
destruction
inundated
upheaval
survive
trouble
blown%20up
body%20bagging
meltdown
flattened
demolish
quarantine
lava
army
bleeding
deluge
screams



In [ ]:

```
```

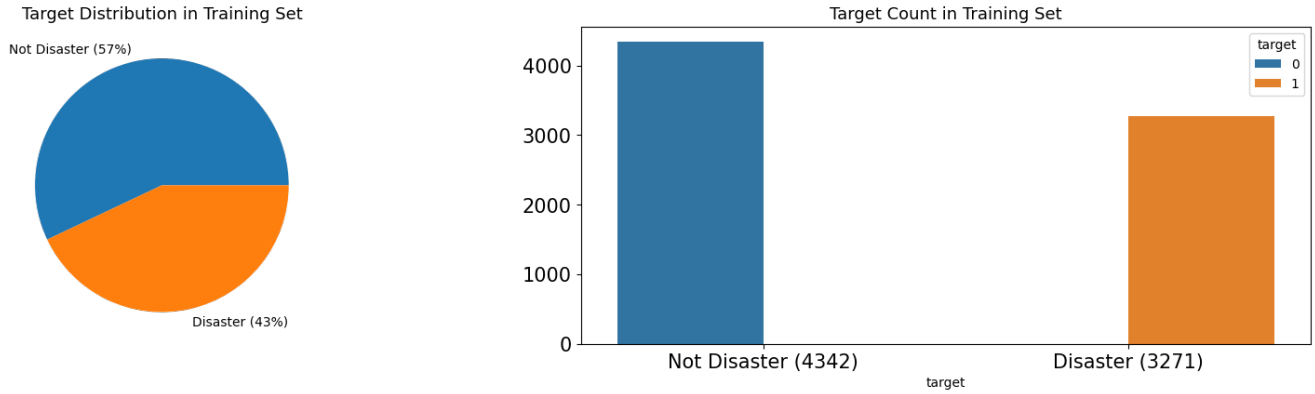
Target

Class distributions are **57%** for **0** (Not Disaster) and **43%** for **1** (Disaster). Classes are almost equally separated so they don't require any stratification by `target` in cross-validation. and accuracy metric can be used

In [ ]:

```
fig, axes = plt.subplots(ncols=2, figsize=(17, 4), dpi=100)
plt.tight_layout()

train.groupby('target').count()['id'].plot(kind='pie', ax=axes[0], labels=['Not Disaster (57%)',
'Disaster (43%)'])
sns.countplot(x=train['target'], hue=train['target'], ax=axes[1])

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()
```



In [ ]:

```
def generate_ngrams(text, n_gram=1):
    token = [token for token in text.lower().split(' ') if token != '' if token not in STOPWORDS]
    ngrams = zip(*[token[i:] for i in range(n_gram)])
    return [' '.join(ngram) for ngram in ngrams]

N = 100
DISASTER_TWEETS = train['target'] == 1
# Unigrams
disaster_unigrams = defaultdict(int)
nondisaster_unigrams = defaultdict(int)

for tweet in train[DISASTER_TWEETS]['text']:
    for word in generate_ngrams(tweet):
        disaster_unigrams[word] += 1

for tweet in train[~DISASTER_TWEETS]['text']:
    for word in generate_ngrams(tweet):
        nondisaster_unigrams[word] += 1

df_disaster_unigrams = pd.DataFrame(sorted(disaster_unigrams.items(), key=lambda x: x[1])[::-1])
df_nondisaster_unigrams = pd.DataFrame(sorted(nondisaster_unigrams.items(), key=lambda x: x[1])[::-
1])

# Bigrams
disaster_bigrams = defaultdict(int)
nondisaster_bigrams = defaultdict(int)

for tweet in train[DISASTER_TWEETS]['text']:
    for word in generate_ngrams(tweet, n_gram=2):
        disaster_bigrams[word] += 1

for tweet in train[~DISASTER_TWEETS]['text']:
    for word in generate_ngrams(tweet, n_gram=2):
        nondisaster_bigrams[word] += 1

df_disaster_bigrams = pd.DataFrame(sorted(disaster_bigrams.items(), key=lambda x: x[1])[::-1])
df_nondisaster_bigrams = pd.DataFrame(sorted(nondisaster_bigrams.items(), key=lambda x: x[1])[::-1]
)

# Trigrams
disaster_trigrams = defaultdict(int)
nondisaster_trigrams = defaultdict(int)

for tweet in train[DISASTER_TWEETS]['text']:
    for word in generate_ngrams(tweet, n_gram=3):
```

```
        disaster_trigrams[word] += 1

for tweet in train[~DISASTER_TWEETS]['text']:
    for word in generate_ngrams(tweet, n_gram=3):
        nondisaster_trigrams[word] += 1

df_disaster_trigrams = pd.DataFrame(sorted(disaster_trigrams.items(), key=lambda x: x[1])[::-1])
df_nondisaster_trigrams = pd.DataFrame(sorted(nondisaster_trigrams.items(), key=lambda x: x[1])[::-
1])
```

Unigrams

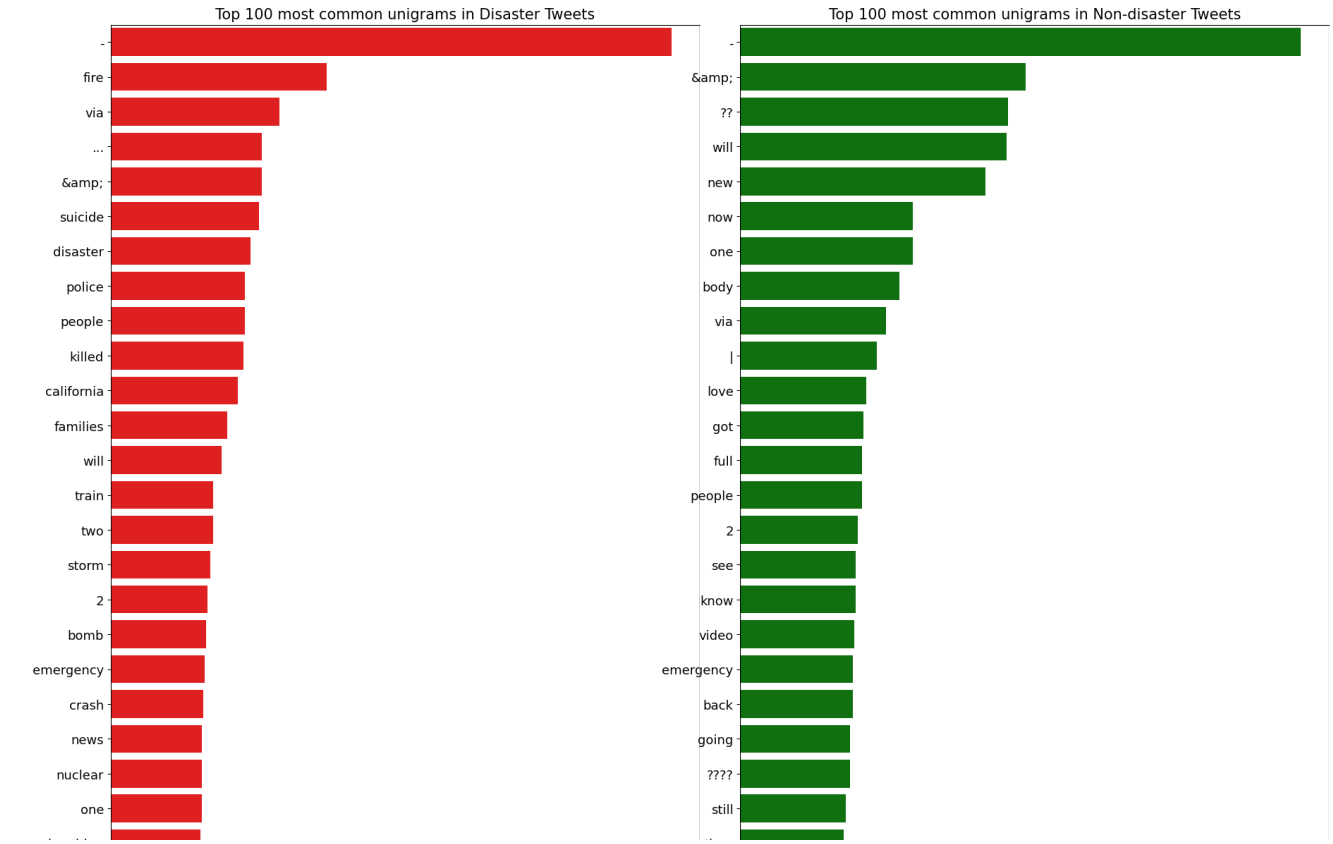
Most common unigrams exist in **both classes** are mostly punctuations, stop words or numbers. It is better to clean them before modelling since they don't give much information about `target`.

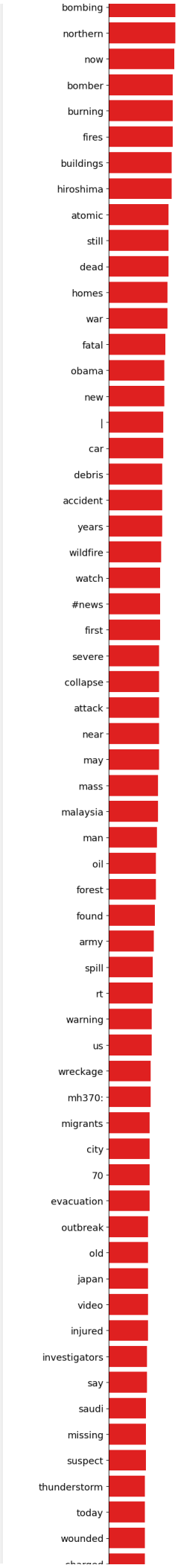
Most common unigrams in **disaster** tweets are already giving information about disasters. It is very hard to use some of those words in other contexts.

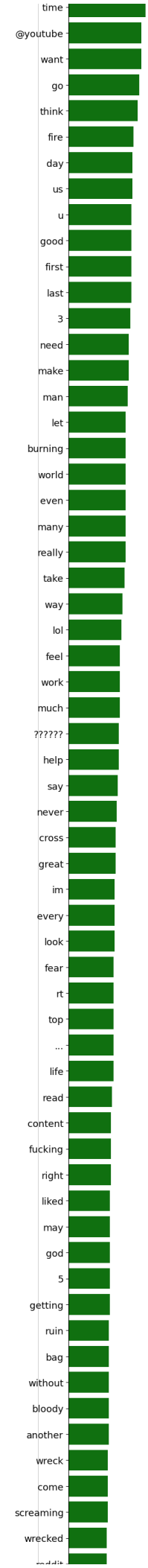
Most common unigrams in **non-disaster** tweets are verbs. This makes sense because most of those sentences have informal active structure since they are coming from individual users.

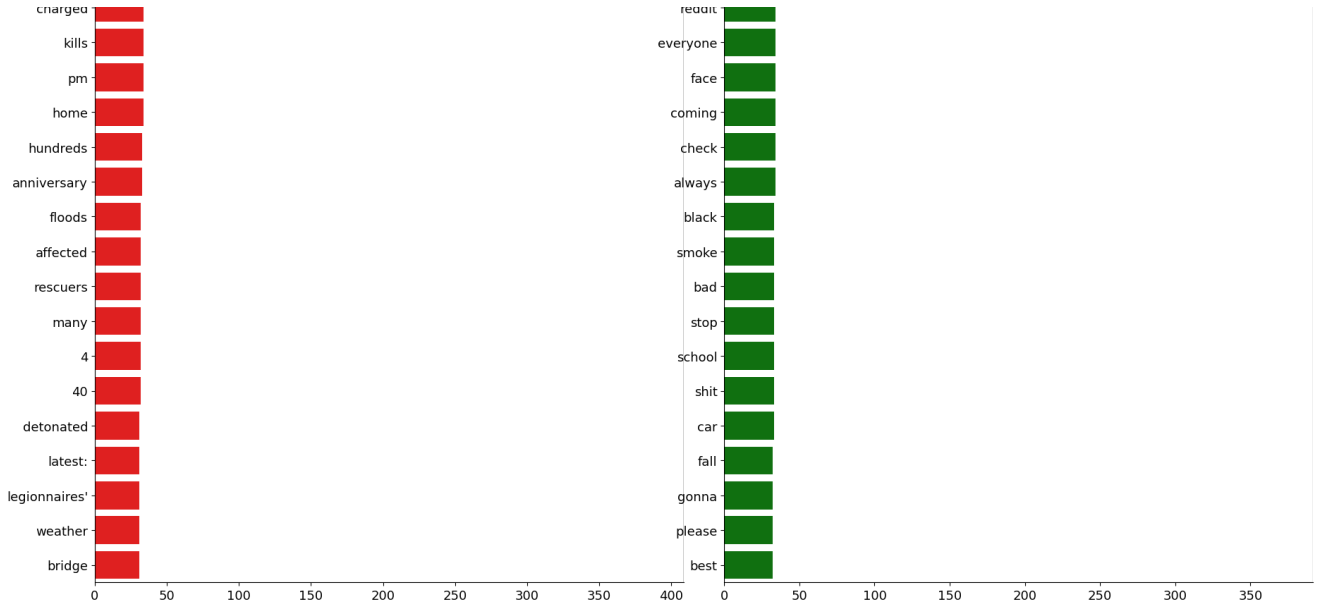
In [ ]:

```
fig, axes = plt.subplots(ncols=2, figsize=(18, 50), dpi=100)
plt.tight_layout()

sns.barplot(y=df_disaster_unigrams[0].values[:N], x=df_disaster_unigrams[1].values[:N], ax=axes[0],
color='red')
sns.barplot(y=df_nondisaster_unigrams[0].values[:N], x=df_nondisaster_unigrams[1].values[:N], ax=ax
es[1], color='green')

for i in range(2):
    axes[i].spines['right'].set_visible(False)
    axes[i].set_xlabel('')
    axes[i].set_ylabel('')
    axes[i].tick_params(axis='x', labelsize=13)
    axes[i].tick_params(axis='y', labelsize=13)

axes[0].set_title(f'Top {N} most common unigrams in Disaster Tweets', fontsize=15)
axes[1].set_title(f'Top {N} most common unigrams in Non-disaster Tweets', fontsize=15)

plt.show()
```




bombing
northern
now
bomber
burning
fires
buildings
hiroshima
atomic
still
dead
homes
war
fatal
obama
new
|
car
debris
accident
years
wildfire
watch
#news
first
severe
collapse
attack
near
may
mass
malaysia
man
oil
forest
found
army
spill
rt
warning
us
wreckage
mh370:
migrants
city
70
evacuation
outbreak
old
japan
video
injured
investigators
say
saudi
missing
suspect
thunderstorm
today
wounded


time
@youtube
want
go
think
fire
day
us
u
good
first
last
3
need
make
man
let
burning
world
even
many
really
take
way
lol
feel
work
much
??????
help
say
never
cross
great
im
every
look
fear
rt
top
...
life
read
content
fucking
right
liked
may
god
5
getting
ruin
bag
without
bloody
another
wreck
come
screaming
wrecked



Bigrams

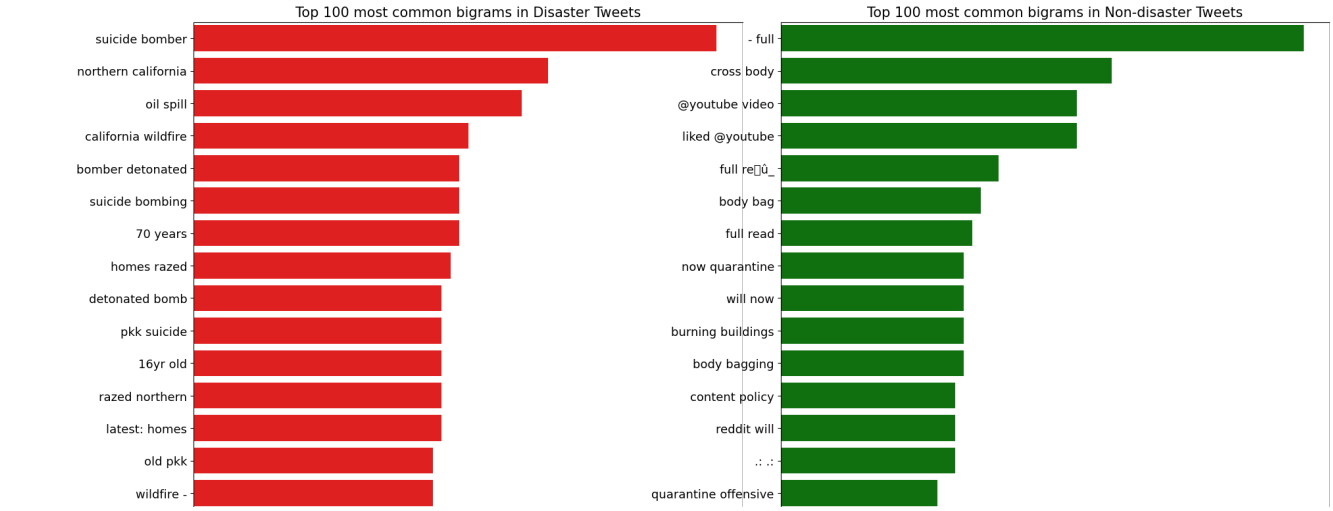
There are no common bigrams exist in **both classes** because the context is clearer.

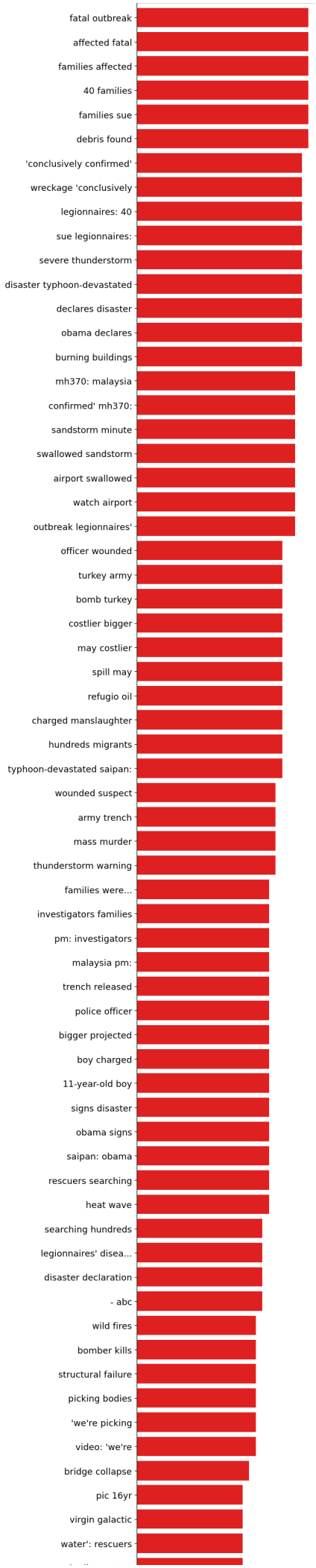
Most common bigrams in **disaster** tweets are giving more information about the disasters than unigrams, but punctuations have to be stripped from words.

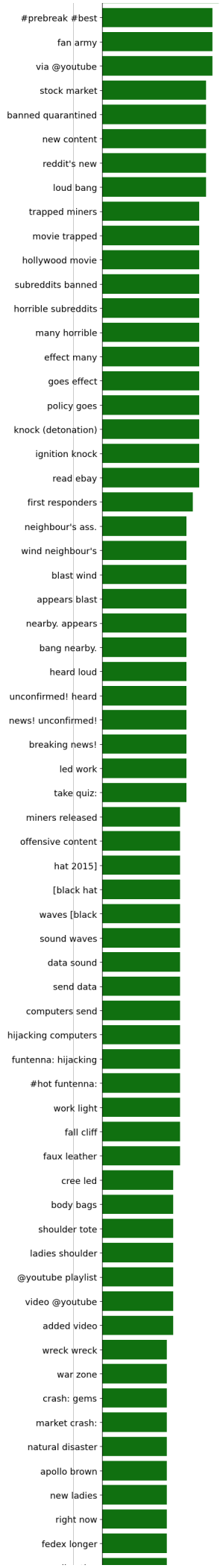
Most common bigrams in **non-disaster** tweets are mostly about reddit or youtube, and they contain lots of punctuations. Those punctuations have to be cleaned out of words as well.

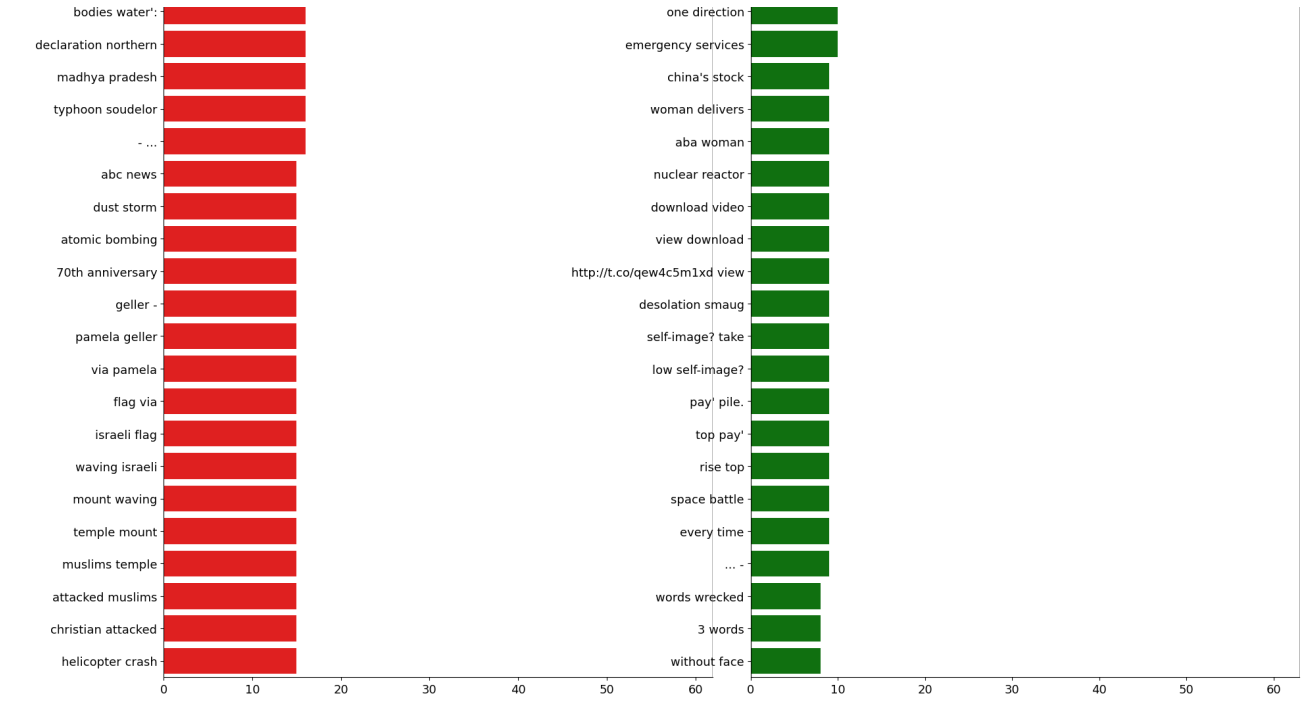
In [ ]:

```
fig, axes = plt.subplots(ncols=2, figsize=(18, 50), dpi=100)
plt.tight_layout()

sns.barplot(y=df_disaster_bigrams[0].values[:N], x=df_disaster_bigrams[1].values[:N], ax=axes[0], c
olor='red')
sns.barplot(y=df_nondisaster_bigrams[0].values[:N], x=df_nondisaster_bigrams[1].values[:N], ax=axes
[1], color='green')

for i in range(2):
    axes[i].spines['right'].set_visible(False)
    axes[i].set_xlabel('')
    axes[i].set_ylabel('')
    axes[i].tick_params(axis='x', labelsize=13)
    axes[i].tick_params(axis='y', labelsize=13)

axes[0].set_title(f'Top {N} most common bigrams in Disaster Tweets', fontsize=15)
axes[1].set_title(f'Top {N} most common bigrams in Non-disaster Tweets', fontsize=15)

plt.show()
```




fatal outbreak
affected fatal
families affected
40 families
families sue
debris found
'conclusively confirmed'
wreckage 'conclusively
legionnaires: 40
sue legionnaires:
severe thunderstorm
disaster typhoon-devastated
declares disaster
obama declares
burning buildings
mh370: malaysia
confirmed' mh370:
sandstorm minute
swallowed sandstorm
airport swallowed
watch airport
outbreak legionnaires'
officer wounded
turkey army
bomb turkey
costlier bigger
may costlier
spill may
refugio oil
charged manslaughter
hundreds migrants
typhoon-devastated saipan:
wounded suspect
army trench
mass murder
thunderstorm warning
families were...
investigators families
pm: investigators
malaysia pm:
trench released
police officer
bigger projected
boy charged
11-year-old boy
signs disaster
obama signs
saipan: obama
rescuers searching
heat wave
searching hundreds
legionnaires' disea...
disaster declaration
- abc
wild fires
bomber kills
structural failure
picking bodies
'we're picking
video: 'we're
bridge collapse
pic 16yr
virgin galactic
water': rescuers


#prebreak #best
fan army
via @youtube
stock market
banned quarantined
new content
reddit's new
loud bang
trapped miners
movie trapped
hollywood movie
subreddits banned
horrible subreddits
many horrible
effect many
goes effect
policy goes
knock (detonation)
ignition knock
read ebay
first responders
neighbour's ass.
wind neighbour's
blast wind
appears blast
nearby. appears
bang nearby.
heard loud
unconfirmed! heard
news! unconfirmed!
breaking news!
led work
take quiz:
miners released
offensive content
hat 2015]
[black hat
waves [black
sound waves
data sound
send data
computers send
hijacking computers
funtenna: hijacking
#hot funtenna:
work light
fall cliff
faux leather
cree led
body bags
shoulder tote
ladies shoulder
@youtube playlist
video @youtube
added video
wreck wreck
war zone
crash: gems
market crash:
natural disaster
apollo brown
new ladies
right now
fedex longer



Trigrams

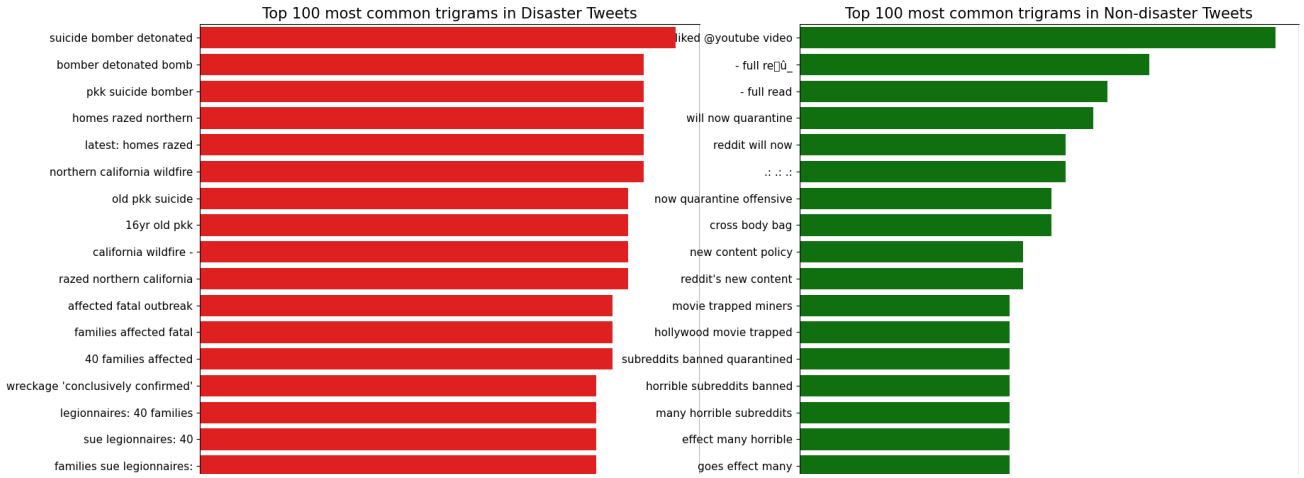
There are no common trigrams exist in **both classes** because the context is clearer.

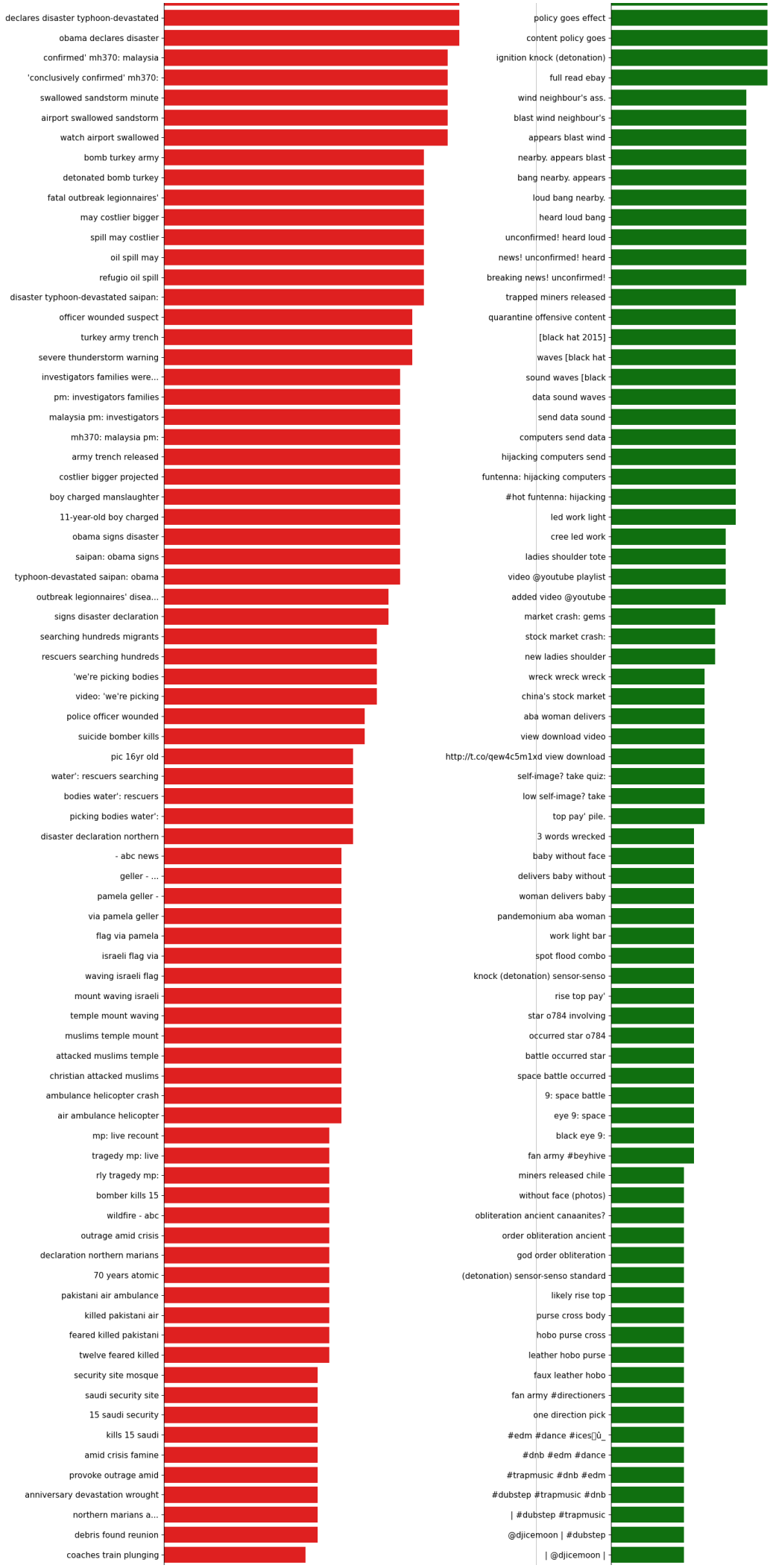
Most common trigrams in **disaster** tweets are very similar to bigrams. They give lots of information about disasters, but they may not provide any additional information along with bigrams.

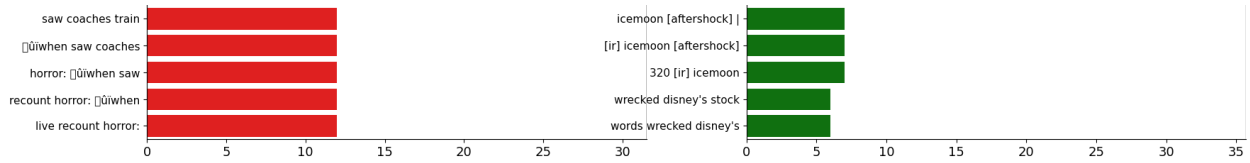
Most common trigrams in **non-disaster** tweets are also very similar to bigrams, and they contain even more punctuations.

In [ ]:

```
fig, axes = plt.subplots(ncols=2, figsize=(20, 50), dpi=100)

sns.barplot(y=df_disaster_trigrams[0].values[:N], x=df_disaster_trigrams[1].values[:N], ax=axes[0],
color='red')
sns.barplot(y=df_nondisaster_trigrams[0].values[:N], x=df_nondisaster_trigrams[1].values[:N], ax=ax
es[1], color='green')

for i in range(2):
    axes[i].spines['right'].set_visible(False)
    axes[i].set_xlabel('')
    axes[i].set_ylabel('')
    axes[i].tick_params(axis='x', labelsize=13)
    axes[i].tick_params(axis='y', labelsize=11)

axes[0].set_title(f'Top {N} most common trigrams in Disaster Tweets', fontsize=15)
axes[1].set_title(f'Top {N} most common trigrams in Non-disaster Tweets', fontsize=15)

plt.show()
```




declares disaster typhoon-devastated
obama declares disaster
confirmed' mh370: malaysia
'conclusively confirmed' mh370:
swallowed sandstorm minute
airport swallowed sandstorm
watch airport swallowed
bomb turkey army
detonated bomb turkey
fatal outbreak legionnaires'
may costlier bigger
spill may costlier
oil spill may
refugio oil spill
disaster typhoon-devastated saipan:
officer wounded suspect
turkey army trench
severe thunderstorm warning
investigators families were...
pm: investigators families
malaysia pm: investigators
mh370: malaysia pm:
army trench released
costlier bigger projected
boy charged manslaughter
11-year-old boy charged
obama signs disaster
saipan: obama signs
typhoon-devastated saipan: obama
outbreak legionnaires' disea...
signs disaster declaration
searching hundreds migrants
rescuers searching hundreds
'we're picking bodies
video: 'we're picking
police officer wounded
suicide bomber kills
pic 16yr old
water': rescuers searching
bodies water': rescuers
picking bodies water':
disaster declaration northern
- abc news
geller - ...
pamela geller -
via pamela geller
flag via pamela
israeli flag via
waving israeli flag
mount waving israeli
temple mount waving
muslims temple mount
attacked muslims temple
christian attacked muslims
ambulance helicopter crash
air ambulance helicopter
mp: live recount
tragedy mp: live
rly tragedy mp:
bomber kills 15
wildfire - abc
outrage amid crisis
declaration northern marians
70 years atomic
pakistani air ambulance
killed pakistani air
feared killed pakistani
twelve feared killed
security site mosque
saudi security site
15 saudi security
kills 15 saudi
amid crisis famine
provoke outrage amid
anniversary devastation wrought
northern marians a...
debris found reunion
coaches train plunging
policy goes effect
content policy goes
ignition knock (detonation)
full read ebay
wind neighbour's ass.
blast wind neighbour's
appears blast wind
nearby. appears blast
bang nearby. appears
loud bang nearby.
heard loud bang
unconfirmed! heard loud
news! unconfirmed! heard
breaking news! unconfirmed!
trapped miners released
quarantine offensive content
[black hat 2015]
waves [black hat
sound waves [black
data sound waves
send data sound
computers send data
hijacking computers send
funtenna: hijacking computers
#hot funtenna: hijacking
led work light
cree led work
ladies shoulder tote
video @youtube playlist
added video @youtube
market crash: gems
stock market crash:
new ladies shoulder
wreck wreck wreck
china's stock market
aba woman delivers
view download video
http://t.co/qew4c5m1xd view download
self-image? take quiz:
low self-image? take
top pay' pile.
3 words wrecked
baby without face
delivers baby without
woman delivers baby
pandemonium aba woman
work light bar
spot flood combo
knock (detonation) sensor-senso
rise top pay'
star o784 involving
occurred star o784
battle occurred star
space battle occurred
9: space battle
eye 9: space
black eye 9:
fan army #beyhive
miners released chile
without face (photos)
obliteration ancient canaanites?
order obliteration ancient
god order obliteration
(detonation) sensor-senso standard
likely rise top
purse cross body
hobo purse cross
leather hobo purse
faux leather hobo
fan army #directioners
one direction pick
#edm #dance #ices□û_
#dnb #edm #dance
#trapmusic #dnb #edm
#dubstep #trapmusic #dnb
| #dubstep #trapmusic
@djicemoon | #dubstep
| @djicemoon |



In [ ]:

```
```

Mislabeled Samples

There are **18** unique tweets in training set which are labeled differently in their duplicates. Those tweets are probably labeled by different people and they interpreted the meaning differently because some of them are not very clear. Tweets with two unique `target` values are relabeled since they can affect the training score.

In [5]:

```
df_mislabeled = train.groupby(['text']).nunique().sort_values(by='target', ascending=False)
df_mislabeled = df_mislabeled[df_mislabeled['target'] > 1]['target']
df_mislabeled.index.tolist()
```

Out[5]:

```
['like for the music video I want some real action shit like burning buildings and police chases n
ot some weak ben winston shit',
 'Hellfire! We don\x89Ûªt even want to think about it or mention it so let\x89Ûªs not do anything
that leads to it #islam!',
 "The Prophet (peace be upon him) said 'Save yourself from Hellfire even if it is by giving half a
date in charity.'",
 'In #islam saving a person is equal in reward to saving all humans! Islam is the opposite of terr
orism!',
 'To fight bioterrorism sir.',
 'Who is bringing the tornadoes and floods. Who is bringing the climate change. God is after Ameri
ca He is plaguing her\n \n#FARRAKHAN #QUOTE',
 '#foodscare #offers2go #NestleIndia slips into loss after #Magginoodle #ban unsafe and hazardous
for #humanconsumption',
 '#Allah describes piling up #wealth thinking it would last #forever as the description of the peo
ple of #Hellfire in Surah Humaza. #Reflect',
 'He came to a land which was engulfed in tribal war and turned it into a land of peace i.e. Madin
ah. #ProphetMuhammad #islam',
 'RT NotExplained: The only known image of infamous hijacker D.B. Cooper. http://t.co/JlzK2HdeTG',
 'Hellfire is surrounded by desires so be careful and don\x89Ûªt let your desires control you!
#Afterlife',
 'CLEARED:incident with injury:I-495  inner loop Exit 31 - MD 97/Georgia Ave Silver Spring',
 "Mmmmmm I'm burning.... I'm burning buildings I'm building.... Ooooooohhhh oooh ooh...",
 'wowo--=== 12000 Nigerian refugees repatriated from Cameroon',
 '.POTUS #StrategicPatience is a strategy for #Genocide; refugees; IDP Internally displaced
people; horror; etc. https://t.co/rqWuoy1fm4',
 'Caution: breathing may be hazardous to your health.',
 'I Pledge Allegiance To The P.O.P.E. And The Burning Buildings of Epic City. ??????',
 'that horrible sinking feeling when you\x89Ûªve been at home on your phone for a while and you re
alise its been on 3G this whole time']
```

In [6]:

```
train['target_relabeled'] = train['target'].copy()

train.loc[train['text'] == 'like for the music video I want some real action shit like burning bui
ldings and police chases not some weak ben winston shit', 'target_relabeled'] = 0
train.loc[train['text'] == 'Hellfire is surrounded by desires so be careful and don Ûªt let your d
esires control you! #Afterlife', 'target_relabeled'] = 0
train.loc[train['text'] == 'To fight bioterrorism sir.', 'target_relabeled'] = 0
train.loc[train['text'] == '.POTUS #StrategicPatience is a strategy for #Genocide; refugees; IDP I
nternally displaced people; horror; etc. https://t.co/rqWuoy1fm4', 'target_relabeled'] = 1
train.loc[train['text'] == 'CLEARED:incident with injury:I-495  inner loop Exit 31 - MD 97/Georgia
Ave Silver Spring', 'target_relabeled'] = 1
train.loc[train['text'] == '#foodscare #offers2go #NestleIndia slips into loss after #Magginoodle
#ban unsafe and hazardous for #humanconsumption', 'target_relabeled'] = 0
train.loc[train['text'] == 'In #islam saving a person is equal in reward to saving all humans! Isl
am is the opposite of terrorism!', 'target_relabeled'] = 0
```

```
train.loc[train['text'] == 'Who is bringing the tornadoes and floods. Who is bringing the climate
change. God is after America He is plaguing her\n \n#FARRAKHAN #QUOTE', 'target_relabeled'] = 1
train.loc[train['text'] == 'RT NotExplained: The only known image of infamous hijacker D.B.
Cooper. http://t.co/JlzK2HdeTG', 'target_relabeled'] = 1
train.loc[train['text'] == "Mmmmmm I'm burning.... I'm burning buildings I'm building....
Ooooooohhhh oooh ooh...", 'target_relabeled'] = 0
train.loc[train['text'] == "wowo--=== 12000 Nigerian refugees repatriated from Cameroon",
'target_relabeled'] = 0
train.loc[train['text'] == "He came to a land which was engulfed in tribal war and turned it into
a land of peace i.e. Madinah. #ProphetMuhammad #islam", 'target_relabeled'] = 0
train.loc[train['text'] == "Hellfire! We don Ûªt even want to think about it or mention it so let Û
ªs not do anything that leads to it #islam!", 'target_relabeled'] = 0
train.loc[train['text'] == "The Prophet (peace be upon him) said 'Save yourself from Hellfire even
if it is by giving half a date in charity.'", 'target_relabeled'] = 0
train.loc[train['text'] == "Caution: breathing may be hazardous to your health.",
'target_relabeled'] = 1
train.loc[train['text'] == "I Pledge Allegiance To The P.O.P.E. And The Burning Buildings of Epic
City. ??????", 'target_relabeled'] = 0
train.loc[train['text'] == "#Allah describes piling up #wealth thinking it would last #forever as
the description of the people of #Hellfire in Surah Humaza. #Reflect", 'target_relabeled'] = 0
train.loc[train['text'] == "that horrible sinking feeling when you Ûªve been at home on your phone
for a while and you realise its been on 3G this whole time", 'target_relabeled'] = 0
```

Glove embedding (word2vec with 300 dimensional)

In [ ]:

```
import gensim.downloader as api

w2v_model = api.load('word2vec-google-news-300')
```

```
[==================================================] 100.0% 1662.8/1662.8MB downloaded
```

In [ ]:

```
print(w2v_model.most_similar('delhi'))
print(w2v_model.most_similar('worst'))
```

```
[('Delhi', 0.681606650352478), ('mumbai', 0.6771246194839478), ('chennai', 0.6614564657211304), ('
kerala', 0.6286536455154419), ('Chandigarh', 0.6241948008537292), ('Ranchi', 0.623480498790741), (
'pune', 0.6212411522865295), ('Lucknow', 0.6141113638877869), ('kolkata', 0.6114561557769775), ('b
ihar', 0.606876015663147)]
[('Worst', 0.6146092414855957), ('weakest', 0.6143776774406433), ('scariest', 0.5957258343696594),
('ugliest', 0.5931180715560913), ('best', 0.5835110545158386), ('bleakest', 0.5718506574630737), (
'strongest', 0.567145586013794), ('nastiest', 0.5644308924674988), ('lousiest',
0.5631451606750488), ('toughest', 0.5624395608901978)]
```

Before preprocessing

we can see only 44 and 50% of vocabulary and 70-71% of text is covered by word2vec because of all the noise and punctuations int text and store the words which were not in vocabulary in "*_glove_oov" for furthur analysis

In [ ]:

```
def build_vocab(X):

    tweets = X.apply(lambda s: s.split()).values
    vocab = {}

    for tweet in tweets:
        for word in tweet:
            try:
                vocab[word] += 1
            except KeyError:
                vocab[word] = 1
    return vocab
```

```
def check_embeddings_coverage(X, embeddings):

    vocab = build_vocab(X)

    covered = {}
    oov = {}
    n_covered = 0
    n_oov = 0

    for word in vocab:
        try:
            covered[word] = embeddings[word]
            n_covered += vocab[word]
        except:
            oov[word] = vocab[word]
            n_oov += vocab[word]

    vocab_coverage = len(covered) / len(vocab)
    text_coverage = (n_covered / (n_covered + n_oov))

    sorted_oov = sorted(oov.items(), key=lambda x: x[1])[::-1]
    return sorted_oov, vocab_coverage, text_coverage,covered

train_glove_oov, train_glove_vocab_coverage, train_glove_text_coverage,train_covered =
check_embeddings_coverage(train['text'], w2v_model)
test_glove_oov, test_glove_vocab_coverage, test_glove_text_coverage,test_covered=
check_embeddings_coverage(test['text'], w2v_model)
print('GloVe Embeddings cover {:.2%} of vocabulary and {:.2%} of text in Training
Set'.format(train_glove_vocab_coverage, train_glove_text_coverage))
print('GloVe Embeddings cover {:.2%} of vocabulary and {:.2%} of text in Test
Set'.format(test_glove_vocab_coverage, test_glove_text_coverage))
```

```
GloVe Embeddings cover 44.36% of vocabulary and 71.50% of text in Training Set
GloVe Embeddings cover 50.27% of vocabulary and 70.94% of text in Test Set
```

preprocessing

In [7]:

```
def decontracted(tweet):
    #after seeing some words were not converted to vectors since they were not in vocabulary we re
ctify the words

    #spelling mistakes
    tweet = re.sub(r"traumatised", "traumatized", tweet)
    tweet = re.sub(r"realise", "realize", tweet)
    tweet = re.sub(r"prebreak", "pre break", tweet)
    tweet = re.sub(r"Mediterran", "Mediterranean", tweet)

    #words combined
    tweet = re.sub(r"subreddits", "sub reddits", tweet)
    tweet = re.sub(r"emmerdale", "Emmerdale", tweet)
    tweet = re.sub(r"nowplaying", "now playing", tweet)
    tweet = re.sub(r"neighbour", "Neighbour", tweet)
    tweet = re.sub(r"colour", "Colour", tweet)
    tweet = re.sub(r"animalrescue", "animal rescue", tweet)
    tweet = re.sub(r"SCREAMED", "Screamed", tweet)
    tweet = re.sub(r"ArianaGrande", "Ariana Grande", tweet)
    tweet = re.sub(r"ProphetMuhammad", "Prophet Muhammad", tweet)

    #accronyms
    tweet = re.sub(r"yyc", "Calgary International Airport", tweet)
    tweet = re.sub(r"Trfc", "transactional Remote Function Call", tweet)
    tweet = re.sub(r"GBBO", "Great British Bake Off", tweet)
    tweet = re.sub(r"Sismo", "Seismograph", tweet)

    tweet = re.sub(r"y'all", "you all", tweet)
    tweet = re.sub(r"let's", "let us", tweet)

     #general
    tweet = re.sub(r"n\'t", " not", tweet)
    tweet = re.sub(r"\'re", " are", tweet)
    tweet = re.sub(r"\'s", " is", tweet)
```

```
    tweet = re.sub(r"\'d", " would", tweet)
    tweet = re.sub(r"\'ll", " will", tweet)
    tweet = re.sub(r"\'t", " not", tweet)
    tweet = re.sub(r"\'ve", " have", tweet)
    tweet = re.sub(r"\'m", " am", tweet)
    return tweet
```

In [8]:

```
import spacy
import re
nlp = spacy.load('en')
def preprocessing(text):
    text = decontracted(text)
  #html
    text = re.sub('<.*?>','',text)
  #urls
    text = re.sub('http[s]?://\S+|www\.\S+','',text)
    token=[]
    result=''
    text = re.sub('[^A-Za-z]',' ',text)


    text = nlp(text)
    for t in text:
        if not t.is_stop and len(t)>2:
        token.append(t)
        result = ' '.join([str(i) for i in token])

    return result.strip()
```

after preprocess the vocabulary coverage increased to 80% and 90% of text is covered

In [ ]:

```
train.text = train.text.apply(lambda x : preprocessing(x))
test.text = test.text.apply(lambda x : preprocessing(x))

train_glove_oov, train_glove_vocab_coverage, train_glove_text_coverage = check_embeddings_coverage
(train['text'], w2v_model)
test_glove_oov, test_glove_vocab_coverage, test_glove_text_coverage =
check_embeddings_coverage(test['text'], w2v_model)
print('GloVe Embeddings cover {:.2%} of vocabulary and {:.2%} of text in Training
Set'.format(train_glove_vocab_coverage, train_glove_text_coverage))
print('GloVe Embeddings cover {:.2%} of vocabulary and {:.2%} of text in Test
Set'.format(test_glove_vocab_coverage, test_glove_text_coverage))
```

```
GloVe Embeddings cover 79.35% of vocabulary and 92.05% of text in Training Set
GloVe Embeddings cover 82.42% of vocabulary and 91.53% of text in Test Set
```

In [ ]:

```
train_glove_oov
```

Out[ ]:

```
[('traumatised', 32),
 ('prebreak', 30),
 ('Soudelor', 22),
 ('disea', 19),
 ('GBBO', 19),
 ('bestnaijamade', 18),
 ('subreddits', 17),
 ('Funtenna', 17),
 ('neighbour', 16),
 ('emmerdale', 14),
 ('NowPlaying', 14),
 ('ArianaGrande', 11),
 ('nowplaying', 11),
 ('yyc', 10),
 ('Sismo', 10),
 ('colour', 9),
 ('Trfc', 9),
```

('Directioners', 9),
('SCREAMED', 8),
('Mediterran', 8),
('abstorm', 8),
('IranDeal', 8),
('Beyhive', 8),
('realise', 7),
('animalrescue', 7),
('Linkury', 7),
('fnet', 7),
('mhtw', 7),
('MTVHottest', 7),
('TrapMusic', 7),
('djicemoon', 7),
('ICEMOON', 7),
('Bestnaijamade', 6),
('okwx', 6),
('ProphetMuhammad', 6),
('MikeParrActor', 6),
('FOXTROT', 6),
('Dorret', 6),
('tcot', 6),
('usatoday', 6),
('KerrickTrial', 5),
('NASAHurricane', 5),
('PantherAttack', 5),
('YoungHeroesID', 5),
('socialnews', 5),
('StrategicPatience', 5),
('nothe', 5),
('USAgov', 5),
('theatre', 5),
('ViralSpell', 5),
('worldnews', 5),
('IDFire', 4),
('Karymsky', 4),
('isuicide', 4),
('Rexyy', 4),
('Selfies', 4),
('DETECTADO', 4),
('WiseNews', 4),
('carryi', 4),
('whao', 4),
('worldnetdaily', 4),
('OBLITERATED', 4),
('OBLITERATE', 4),
('lulgzimbestpicts', 4),
('RAmag', 4),
('DiamondKesawn', 4),
('RaynbowAffair', 4),
('Zouma', 4),
('OTRAMETLIFE', 4),
('WorldNews', 4),
('epicentre', 4),
('EarthQuake', 4),
('abcnews', 4),
('SummerFate', 4),
('wmata', 4),
('unsuckdcmetro', 4),
('tubestrike', 4),
('centre', 4),
('AbbsWinston', 4),
('OPPRESSIONS', 4),
('SLANGLUCCI', 4),
('FETTILOOTCH', 4),
('KurtSchlichter', 4),
('beforeitsnews', 4),
('LoneWolffur', 4),
('lgl', 4),
('BLKs', 4),
('aRmageddon', 4),
('PBBan', 4),
('WITHER', 4),
('VoteJKT', 4),
('nsfw', 4),
('RockyFire', 4),
('Throwingknifes', 3),

('dannyonpc', 3),
('thankU', 3),
('GodsLove', 3),
('usNWSgov', 3),
('BUDDYS', 3),
('bookboost', 3),
('ibooklove', 3),
('AoMS', 3),
('BlowMandyUp', 3),
('iTunesMusic', 3),
('realmandyrain', 3),
('NickCannon', 3),
('onlinecommunities', 3),
('amudslide', 3),
('FaroeIslands', 3),
('matako', 3),
('guillermo', 3),
('Bluedio', 3),
('LOOPING', 3),
('Humaza', 3),
('Olap', 3),
('humanconsumption', 3),
('Magginoodle', 3),
('NestleIndia', 3),
('foodscare', 3),
('yycstorm', 3),
('arwx', 3),
('FARRAKHAN', 3),
('MNPDNashville', 3),
('selfie', 3),
('grey', 3),
('TfLBusAlerts', 3),
('GamerGate', 3),
('IHHen', 3),
('Connecto', 3),
('ApolloBrown', 3),
('spinningbot', 3),
('TinyJecht', 3),
('demonstratio', 3),
('DESOLATE', 3),
('ModiMinistry', 3),
('YahooNews', 3),
('realDonaldTrump', 3),
('DavidVonderhaar', 3),
('saddlebrooke', 3),
('gerenciatodos', 3),
('accionempresa', 3),
('CecilTheLion', 3),
('ArtistsUnited', 3),
('ClaytonBryant', 3),
('SOUDELOR', 3),
('NAVBL', 3),
('TAXIWAYS', 3),
('OPER', 3),
('WorstSummerJob', 3),
('Enrt', 3),
('HarryBeCareful', 3),
('tyre', 3),
('sharethis', 3),
('weathernetwork', 3),
('instagram', 3),
('mfs', 3),
('snapchat', 3),
('IDis', 3),
('WHTs', 3),
('BIOTERRORISM', 3),
('scoopit', 3),
('GOPDebate', 3),
('RickPerry', 3),
('NewsInTweets', 3),
('UltimaLucha', 3),
('TrubGME', 3),
('ANNIHILATED', 3),
('afterShock', 3),
('JonathanFerrell', 2),
('ushed', 2),
('wocowae', 2),

('WindStorm', 2),
('TheTawniest', 2),
('ahrary', 2),
('cheyenne', 2),
('rapidcity', 2),
('sixpenceee', 2),
('Diaporama', 2),
('Aogashima', 2),
('Raung', 2),
('NASASolarSystem', 2),
('stormchase', 2),
('OutBid', 2),
('lavenderpoetrycafe', 2),
('EudryLantiqua', 2),
('LivingSafely', 2),
('OriginalFunko', 2),
('MetroFmTalk', 2),
('Autoinsurance', 2),
('narendramodi', 2),
('Suruc', 2),
('rightwaystan', 2),
('Rightways', 2),
('PLlolz', 2),
('treeporn', 2),
('SouthDowns', 2),
('cancelled', 2),
('FEVWarrior', 2),
('gidi', 2),
('WEYREY', 2),
('YoNews', 2),
('camilacabello', 2),
('ishell', 2),
('SANDSTORM', 2),
('BLutz', 2),
('bbcnews', 2),
('UndergroundRailraod', 2),
('YAHIstorical', 2),
('CindyNoonan', 2),
('wowo', 2),
('ZippedNews', 2),
('freespeech', 2),
('amageddon', 2),
('PANICKING', 2),
('montetjwitter', 2),
('Tgirl', 2),
('BANKSTOWN', 2),
('OBLITERATION', 2),
('GlobalWarming', 2),
('MicheleBachman', 2),
('JonVoyage', 2),
('BakeOffFriends', 2),
('MUDSLIDE', 2),
('TeamHendrick', 2),
('YMcglaun', 2),
('ushanka', 2),
('NoSurrender', 2),
('PlayThursdays', 2),
('alexbelloli', 2),
('itsjustinstuart', 2),
('gunsense', 2),
('DebateQuestionsWeWantToHear', 2),
('RoyalCarribean', 2),
('anyname', 2),
('samanthaturne', 2),
('shawie', 2),
('MartinMJ', 2),
('SuryaRay', 2),
('pattonoswalt', 2),
('MeteoEarth', 2),
('minhazmerchant', 2),
('auspol', 2),
('TLVFaces', 2),
('pmarca', 2),
('Softenza', 2),
('NotExplained', 2),
('greatbritishbakeoff', 2),
('pdx', 2)

('pdx', 2),
('COPYCATS', 2),
('Multidimensi', 2),
('LondonFire', 2),
('EBOLA', 2),
('jamaicaplain', 2),
('KOTAWeather', 2),
('Japton', 2),
('RouteComplex', 2),
('INSubcontinent', 2),
('FLATTENED', 2),
('LINERLESS', 2),
('LuchaUnderground', 2),
('NJTurnpike', 2),
('Politifiact', 2),
('ZSS', 2),
('vaBengal', 2),
('vgbootcamp', 2),
('FaTality', 2),
('GMMBC', 2),
('versethe', 2),
('Shizune', 2),
('LiveOnK', 2),
('NewsGoldCoast', 2),
('TubeStrike', 2),
('EFAK', 2),
('MissionHills', 2),
('batfanuk', 2),
('ProtectDenaliWolves', 2),
('NANKANA', 2),
('weallheartonedirection', 2),
('danisnotonfire', 2),
('niall', 2),
('isuperhero', 2),
('gawx', 2),
('scwx', 2),
('gofundme', 2),
('pmharper', 2),
('LatestNews', 2),
('realised', 2),
('Worldnews', 2),
('IvanBerroa', 2),
('LosDelSonido', 2),
('ARNLEY', 2),
('MOTORCRAFT', 2),
('bancodeseries', 2),
('Marquei', 2),
('thisiswhywecanthavenicethings', 2),
('zaynmalik', 2),
('smaug', 2),
('Photoset', 2),
('comingsoon', 2),
('redeemeth', 2),
('CorleoneDaBoss', 2),
('IndianNews', 2),
('timkaine', 2),
('IdentityTheft', 2),
('AllLivesMatter', 2),
('liveOnline', 2),
('mishacollins', 2),
('programme', 2),
('BillNeelyNBC', 2),
('BeClearOnCancer', 2),
('Kowing', 2),
('isecond', 2),
('ScreamQueens', 2),
('AskCharley', 2),
('goku', 2),
('cawx', 2),
('BlizzHeroes', 2),
('BradleyBrad', 2),
('HannaPH', 2),
('wordpressdotcom', 2),
('Jhaustin', 2),
('nikeplus', 2),
('SportWatch', 2),
('samel', 2),
('meinlcymbals', 2)

('MelHeyMbals', 2),
('Techesback', 2),
('Walerga', 2),
('Ptbo', 2),
('SmantiBatam', 2),
('Ruebs', 2),
('savebees', 2),
('JamesMelville', 2),
('newyorker', 2),
('hiroshima', 2),
('gbbo', 2),
('peterjukes', 2),
('megynkelly', 2),
('GreenHarvard', 2),
('StandwithPP', 2),
('cnewslive', 2),
('JamaicaObserver', 2),
('atk', 2),
('hermancranston', 2),
('myfitnesspal', 2),
('LoseIt', 2),
('TweetLikeItsSeptember', 2),
('RockBottomRadFM', 2),
('fewmoretweets', 2),
('BlackLivesMatter', 2),
('cjoyner', 2),
('minimehh', 2),
('themagickidraps', 2),
('Ooooooohhhh', 2),
('ENGvAUS', 2),
('ameenshaikh', 2),
('ProSyn', 2),
('NBCNews', 2),
('Daesh', 2),
('TeamStream', 2),
('wouldemocracy', 2),
('ScottWalker', 2),
('Micom', 2),
('MeekMill', 2),
('Ashayo', 2),
('anellatulip', 2),
('iClown', 2),
('magisters', 2),
('favourite', 2),
('DMPL', 2),
('listenlive', 2),
('CDCgov', 2),
('FoxNew', 2),
('CBSBigBrother', 2),
('DIDNT', 2),
('jdabe', 2),
('JulieDiCaro', 2),
('NickCoCoFree', 2),
('envw', 2),
('diamorfiend', 2),
('theadvocatemag', 2),
('CountyNews', 2),
('VideoVeranoMTV', 2),
('SOSFAM', 2),
('RohnertParkDPS', 2),
('LOTZ', 2),
('RTRRT', 2),
('lith', 2),
('TrueLove', 2),
('ouvindo', 2),
('THISIZBWRIGHT', 2),
('DoubleCups', 2),
('Popularmmos', 2),
('WildHorses', 2),
('TomcatArts', 2),
('FantasticFour', 2),
('Trancy', 2),
('fouseyTUBE', 2),
('travelling', 2),
('yugvani', 2),
('scuf', 2),
('DeLo', 2),
('HORNDALE', 2),

('HORNDALE', 2),
('PINER', 2),
('nothere', 2),
('mufc', 2),
('Ssw', 1),
('BathAndNorthEastSomerset', 1),
('tnwx', 1),
('Ayyo', 1),
('explodingkittens', 1),
('gameofkittens', 1),
('engineshed', 1),
('cameronhacker', 1),
('widda', 1),
('wario', 1),
('Kirafrog', 1),
('bArra', 1),
('Caitsroberts', 1),
('topnews', 1),
('marynmck', 1),
('stevenrulles', 1),
('ohhmyjoshh', 1),
('WGG', 1),
('residualincome', 1),
('thatswhatfriendsarefor', 1),
('oliviaapalmerr', 1),
('istare', 1),
('RossMartin', 1),
('GrantGordy', 1),
('YahooNewsDigest', 1),
('MalaysiaAirlines', 1),
('najibrazak', 1),
('ajabrown', 1),
('janeenorman', 1),
('raineishida', 1),
('ombudsmanship', 1),
('LARIOUS', 1),
('TitorTau', 1),
('AmazonDeals', 1),
('MissCharleyWebb', 1),
('shoalstraffic', 1),
('GeorgeFoster', 1),
('girlthatsrio', 1),
('newave', 1),
('artectura', 1),
('IceQueenFroslas', 1),
('PokemonCards', 1),
('restrospect', 1),
('Memenaar', 1),
('destiel', 1),
('Captainn', 1),
('Squeaver', 1),
('Sorrower', 1),
('soldi', 1),
('DauntedPsyche', 1),
('NicolaClements', 1),
('IcyMagistrate', 1),
('neverforget', 1),
('alaskan', 1),
('BritishBakeOff', 1),
('BishopFred', 1),
('celebrety', 1),
('Zarry', 1),
('mattmosley', 1),
('TWCNewsCLT', 1),
('DianneG', 1),
('KashmirConflict', 1),
('FreeKashmir', 1),
('EndConflict', 1),
('EndOccupation', 1),
('UNENDING', 1),
('UNHEALED', 1),
('CharlesDagnall', 1),
('Senzu', 1),
('Latestnews', 1),
('wwp', 1),
('AsterPuppet', 1),
('claimin', 1),
('MGTAB', 1),

('MGTAB', 1),
('MFRWauthor', 1),
('KindleCountdown', 1),
('NoMoreHandouts', 1),
('rangerkaitimay', 1),
('goodreads', 1),
('datingtips', 1),
('ijournal', 1),
('WINDSTORM', 1),
('blakeshelton', 1),
('BeSafe', 1),
('charlesadler', 1),
('chriscesq', 1),
('twia', 1),
('txlege', 1),
('WZBT', 1),
('Gbonyin', 1),
('WSVR', 1),
('niiiice', 1),
('TheBachelorette', 1),
('ZiUW', 1),
('lastingness', 1),
('GalvNews', 1),
('katunews', 1),
('CalWildfires', 1),
('easternoregon', 1),
('LancasterOnline', 1),
('UTFire', 1),
('FireNews', 1),
('smem', 1),
('ListenLive', 1),
('CAFire', 1),
('azwx', 1),
('FingerRockFire', 1),
('doessnt', 1),
('WBCShirl', 1),
('ProfitToThePeople', 1),
('EnzasBargains', 1),
('randerson', 1),
('forageSECRET', 1),
('Farmr', 1),
('MORELS', 1),
('thedayct', 1),
('SenFeinstein', 1),
('ispecialist', 1),
('PrisonPlanet', 1),
('cnni', 1),
('Jennife', 1),
('FOXDebateQuestions', 1),
('SanOnofre', 1),
('riooooos', 1),
('Articals', 1),
('picthis', 1),
('cantwaittoplayinminneapolis', 1),
('whedonesque', 1),
('Summerhallery', 1),
('edfringe', 1),
('ckosova', 1),
('TheEmoBrago', 1),
('FroFroFro', 1),
('RemyMarcel', 1),
('ktfounder', 1),
('kttape', 1),
('alexhammerstone', 1),
('Flatback', 1),
('LucyMayOfficial', 1),
('byuwnbeki', 1),
('SYJEXO', 1),
('Suho', 1),
('Baekhyun', 1),
('VixMeldrew', 1),
('DrMartyFox', 1),
('jaiden', 1),
('Glosblue', 1),
('FinsUp', 1),
('AnimalAdvocate', 1),
('skynet', 1),

('TOKTeacher', 1),
('hurn', 1),
('Erasuterism', 1),
('eyecuts', 1),
('DorisMatsui', 1),
('OjouBot', 1),
('GameRant', 1),
('WOOOOOOO', 1),
('USWarCrimes', 1),
('ineedexposure', 1),
('Gunsense', 1),
('HeidiA', 1),
('NOOB', 1),
('BLACKCATS', 1),
('Hendrixonfire', 1),
('WaynesterAtl', 1),
('danagould', 1),
('lurkin', 1),
('servicin', 1),
('catalogue', 1),
('Snazzychipz', 1),
('SalmanMyDarling', 1),
('splattershot', 1),
('splatoon', 1),
('ThatRussianMan', 1),
('Splatling', 1),
('Splatdown', 1),
('Slosher', 1),
('choppas', 1),
('CrayKain', 1),
('InsaneLimits', 1),
('astrologian', 1),
('Yeyeulala', 1),
('jiahahahha', 1),
('kwwwkwwwk', 1),
('muttatek', 1),
('RosemaryTravale', 1),
('DwarfOnJetpack', 1),
('junsuisengen', 1),
('RogueWatson', 1),
('GrowingupBlack', 1),
('RobertONeill', 1),
('kasiakosek', 1),
('clairecmc', 1),
('GovJayNixon', 1),
('WorldWatchesFerguson', 1),
('onshit', 1),
('seasonfrom', 1),
('LittleWomenLA', 1),
('Jasmines', 1),
('scegnews', 1),
('PageXXI', 1),
('CityofMemphis', 1),
('LATechWatch', 1),
('CSismica', 1),
('purduee', 1),
('lexi', 1),
('alextucker', 1),
('songhey', 1),
('Muazimus', 1),
('MrMikeEaton', 1),
('Maailiss', 1),
('Korzhonov', 1),
('uprootin', 1),
('iateyourfood', 1),
('TeaFrystlik', 1),
('Skarletan', 1),
('Scourgue', 1),
('BANTROPHYHUNTING', 1),
('BADChoices', 1),
('notaking', 1),
('orpol', 1),
('orcot', 1),
('areporting', 1),
('bRZjc', 1),
('postexistence', 1),
('TEJc', 1),

('JhmNYe', 1),
('ENTERSHIKARI', 1),
('chestertweetsuk', 1),
('lyeh', 1),
('urbanisation', 1),
('Yygb', 1),
('nbanews', 1),
('GeneralNews', 1),
('soudelor', 1),
('DevPeace', 1),
('AceNewsDesk', 1),
('declarat', 1),
('jrlallo', 1),
('liguistic', 1),
('TheJonesesVoice', 1),
('TwisterLovesShania', 1),
('ShaniaTwain', 1),
('TheTwisterOkc', 1),
('TheBuffShow', 1),
('davidlabrava', 1),
('EmilioRivera', 1),
('mrsbinker', 1),
('rodarmer', 1),
('LLEGASTE', 1),
('ElianaRaquel', 1),
('carolinagutierr', 1),
('briannafrost', 1),
('toosoon', 1),
('ellenfromnowon', 1),
('PathFinders', 1),
('NoEmotion', 1),
('Tittie', 1),
('TheTwister', 1),
('ItrAWcWB', 1),
('JustJon', 1),
('sarahmcpants', 1),
('EDcXO', 1),
('Hardside', 1),
('oMw', 1),
('OKTXDUO', 1),
('thetwister', 1),
('todayI', 1),
('Dalroy', 1),
('alil', 1),
('GreenLacey', 1),
('Campanha', 1),
('youuu', 1),
('Kamunt', 1),
('eyesss', 1),
('slone', 1),
('nopeach', 1),
('Arceen', 1),
('Borgeous', 1),
('DVBBS', 1),
('blockchain', 1),
('bitcoing', 1),
('Crptotech', 1),
('BBShelli', 1),
('freefromwolves', 1),
('yancey', 1),
('helene', 1),
('nopower', 1),
('PrinceofFencing', 1),
('nkulw', 1),
('charlieputh', 1),
('KerryKatona', 1),
('charleyisqueen', 1),
('niallhariss', 1),
('caaaaaall', 1),
('WildWestSixGun', 1),
('JulieKragt', 1),
('detroitnews', 1),
('RYANS', 1),
('Fartanxiety', 1),
('blanktgt', 1),
('rslm', 1),
('staubs', 1),

```
('JCOMANSE', 1),
('annajhm', 1),
('comcastcares', 1),
('wbu', 1),
('JusstdoitGirl', 1),
('smoak', 1),
('BadAstronomer', 1),
('canagal', 1),
('theBargain', 1),
('PreOrder', 1),
('lucysforsale', 1),
('learni', 1),
('VarageSale', 1),
('brookesddl', 1),
('ArgentaElite', 1),
('laurathorne', 1),
('megancoopy', 1),
('labour', 1),
('cwheate', 1),
('KushWush', 1),
('VickyBrush', 1),
('CiaraMcKendry', 1),
('LIVA', 1),
('keits', 1),
('EVEREVER', 1),
('wrongperson', 1),
('snapchatselfie', 1),
('disneyIrh', 1),
('wouldead', 1),
('malabamiandsons', 1),
('wrongdejavu', 1),
('PerkPearl', 1),
('Yessum', 1),
('MPRnews', 1),
('ianokavo', 1),
('PyramidHead', 1),
('niamhosullivanx', 1),
('AnnmarieRonan', 1),
('JamieGriff', 1),
('Ruddyyyyyy', 1),
('oneborn', 1),
('PioneerPress', 1),
('dateswhitecoats', 1),
('mustachemurse', 1),
('FOAMcc', 1),
('FOAMed', 1),
('paeds', 1),
('usg', 1),
('pozarmy', 1),
('RandallPinkston', 1),
('keithboykin', 1),
('ARobotLegion', 1),
('Intragenerational', 1),
('hempoil', 1),
('Fricken', 1),
('saalon', 1),
('thetimepast', 1),
('AshGhebranious', 1),
('gravitychat', 1),
('PTSDchat', 1),
('ptsdchat', 1),
('crazyindapeg', 1),
('TLoZ', 1),
('AGDQ', 1),
('MorganLawGrp', 1),
('DrPhil', 1),
('RaabChar', 1),
('NissanNews', 1),
('giang', 1),
('TaraSwart', 1),
('lilithsaintcrow', 1),
('HugoMatz', 1),
('BoyInAHorsemask', 1),
('newsdict', 1),
('NARRY', 1),
('zarry', 1),
('lilourry', 1),
```

```
('dramaa', 1),
('Hollyw', 1),
('BlakeSchmidt', 1),
('BattleRoyaleMod', 1),
('onihimedesu', 1),
('prettyboyshyflizzy', 1),
('DICKHEADS', 1),
('almusafirah', 1),
('CSAresu', 1),
('ariana', 1),
('hirochii', 1),
('selenaluna', 1),
('DLin', 1),
('sriramk', 1),
('tanehisicoates', 1),
('caued', 1),
('TOIIndiaNews', 1),
('thisisperidot', 1),
('HomeworldGym', 1),
('Maaaaan', 1),
('timesofindia', 1),
('TopStories', 1),
('wyattb', 1),
('Toocodtodd', 1),
('Ayshun', 1),
('isub', 1),
('Rebelmage', 1),
('kunalkapoor', 1),
('SakhalinTribune', 1),
('evebrigid', 1),
('ticklemeshawn', 1),
('thebookclub', 1),
('TornadoGiveaway', 1),
('BrrookkllyynnR', 1),
('soonergrunt', 1),
('lightningstrike', 1),
('VivaArgentina', 1),
('OKwx', 1),
('FForecast', 1),
('SPSGSP', 1),
('iNWS', 1),
('RachelRofe', 1),
('xylodemon', 1),
('beenghosting', 1),
('museawayfic', 1),
('LORR', 1),
('NIggas', 1),
('HaydnExists', 1),
('SWINGMAN', 1),
('essenceOfMe', 1),
('leonalewis', 1),
('Aaaaaand', 1),
('KevR', 1),
('NemesisK', 1),
('AsimTanvir', 1),
('udest', 1),
('weebly', 1),
('thewebbeffect', 1),
('populardemand', 1),
('WATERMELOANN', 1),
('SoothMySlumber', 1),
('GoKitGo', 1),
('JoeStrong', 1),
('LightUmUpBeast', 1),
('nlwx', 1),
('DevinJoslyn', 1),
('KristyLeeMusic', 1),
('BLOODBOUND', 1),
('PlayingNow', 1),
('Illusoria', 1),
('dmassa', 1),
('Islamaphobe', 1),
('BHUSA', 1),
('moblin', 1),
('KingGerudo', 1),
('newscomauHQ', 1),
('ThreatIntel', 1),
```

```
('DFIR', 1),
('ThreatConnect', 1),
('metrofmtalk', 1),
('keepingtheviginaclean', 1),
('ALIPAPER', 1),
('washard', 1),
('shedid', 1),
('lovemywife', 1),
('UdhampurAgain', 1),
('acmilan', 1),
('ImmortalTech', 1),
('TaoistInsight', 1),
('SwiftyCommissh', 1),
('IDEK', 1),
('ShipsXAnchors', 1),
('KasabWe', 1),
('rahulkanwal', 1),
('warra', 1),
('bjpsamvad', 1),
('AcheDin', 1),
('convivted', 1),
('OfficialMqm', 1),
('AdamNibloe', 1),
('DireTube', 1),
('Dawabsha', 1),
('LDNR', 1),
('huyovoeTripolye', 1),
('RobPulseNews', 1),
('JLester', 1),
('ARizzo', 1),
('LawfulSurvivor', 1),
('FedPorn', 1),
('Arovolturi', 1),
('LungCancer', 1),
('CrackedGem', 1),
('MithiTennis', 1),
('kotowsa', 1),
('coppednews', 1),
('nuclearweapons', 1),
('humanityI', 1),
('BBCWomansHour', 1),
('davidduchovny', 1),
('MadMakNY', 1),
('duchovbutt', 1),
('THRArchives', 1),
('Angelheartnight', 1),
('PixelJanosz', 1),
('MyLifeStory', 1),
('Chronicillness', 1),
('Tsutomi', 1),
('makinmemories', 1),
('notwins', 1),
('whatsapp', 1),
('thalapathi', 1),
('vimvith', 1),
('KarnakaranK', 1),
('rjkrraj', 1),
('ChubyChux', 1),
('daubt', 1),
('TheSmallClark', 1),
('scofield', 1),
('taaylordarr', 1),
('thoutaylorbrown', 1),
('holomua', 1),
('TheDailyShow', 1),
('driverlesscars', 1),
('adriennetomah', 1),
('ArvindKejriwal', 1),
('NitishKumar', 1),
('DDNewsLive', 1),
('wew', 1),
('hikagezero', 1),
('mochichiiiii', 1),
('zombiefunrun', 1),
('teamsurvivors', 1),
('Oktaviana', 1),
('ThomasSMonson', 1),
```

```
 ('ELITIST', 1),
 ('dorrian', 1),
 ('tootrue', 1),
 ('thruuu', 1),
 ('AYHHHHHDJJFJRJJRDJJEKS', 1),
 ('lucypalladino', 1),
 ('silverstar', 1),
 ('UnrealTouch', 1),
 ('bonhomme', 1),
 ('mihirssharma', 1),
 ('ShekharGupta', 1),
 ('ResoluteShield', 1),
 ('lonelyness', 1),
 ('ArrestpastorNganga', 1),
 ('BenAffleck', 1),
 ('UntamedDirewolf', 1),
 ('SaintRobinho', 1),
 ('CodeMeW', 1),
 ('ResoluteVanity', 1),
 ('Blaaaaaaa', 1),
 ('YAMASHIRO', 1),
 ('aphyr', 1),
 ('backty', 1),
 ('Safsufa', 1),
 ('NBCPolitics', 1),
 ('Shareaholic', 1),
 ('muzzies', 1),
 ('MiddleEastEye', 1),
 ('RayquazaErk', 1),
 ('KPPolice', 1),
 ('KPdied', 1),
 ('Saadthe', 1),
 ('JewhadiTM', 1),
 ('trickxie', 1),
 ('Shimmyfab', 1),
 ('sonofbobBOB', 1),
 ('GROGParty', 1),
 ('daisycuttertz', 1),
 ('noahj', 1),
 ('bbclaurak', 1),
 ('deai', 1),
 ('CanadaSuicide', 1),
 ('saudiarabia', 1),
 ('josephjett', 1),
 ('GRupdates', 1),
 ('Hashd', 1),
 ('ALWSL', 1),
 ('MsOreo', 1),
 ('AlfaPedia', 1),
 ('rembr', 1),
 ('FoxNewsInsider', 1),
 ('whvholst', 1),
 ('stopevictions', 1),
 ('CatoInstitute', 1),
 ('sabcnewsroom', 1),
 ('jeremyduns', 1),
 ('NafeezAhmed', 1),
 ('SCynic', 1),
 ('sirtophamhat', 1),
 ('SirTitan', 1),
 ...]
```

In [ ]:

```
test_glove_oov
```

Out[ ]:

```
[('traumatised', 16),
 ('prebreak', 11),
 ('neighbour', 10),
 ('colour', 10),
 ('Funtenna', 9),
 ('emmerdale', 9),
 ('SCREAMED', 7),
 ('subreddits', 7),
```

('Soudelor', 7),
('bestnaijamade', 6),
('disea', 6),
('MTVHottest', 6),
('Olap', 5),
('yyc', 5),
('MikeParrActor', 5),
('SummerFate', 5),
('Directioners', 5),
('Mediterran', 4),
('PlayThursdays', 4),
('Linkury', 4),
('abstorm', 4),
('fnet', 4),
('mhtw', 4),
('ViralSpell', 4),
('LivingSafely', 3),
('ekdar', 3),
('WiseNews', 3),
('plsss', 3),
('reaad', 3),
('GBBO', 3),
('TontitownGrape', 3),
('yycstorm', 3),
('Zouma', 3),
('okwx', 3),
('realmandyrain', 3),
('GraysonDolan', 3),
('StrategicPatience', 3),
('ARNLEY', 3),
('ApolloBrown', 3),
('liveOnline', 3),
('saddlebrooke', 3),
('AbbsWinston', 3),
('ShaunKing', 3),
('SOUDELOR', 3),
('istrong', 3),
('MeekMill', 3),
('IDis', 3),
('lgl', 3),
('USAgov', 3),
('WHTs', 3),
('BLKs', 3),
('WITHER', 3),
('TrapMusic', 3),
('djicemoon', 3),
('ICEMOON', 3),
('pdx', 3),
('nowplaying', 3),
('boonew', 2),
('NCJFCJ', 2),
('hempoil', 2),
('TeambrianMundial', 2),
('TornadoGiveaway', 2),
('timesofindia', 2),
('Dannic', 2),
('Bestnaijamade', 2),
('GRupdates', 2),
('SouthDowns', 2),
('MAGNE', 2),
('LOCOMOTIVES', 2),
('sumn', 2),
('AMPLIFIER', 2),
('WHELEN', 2),
('Selfies', 2),
('Sismo', 2),
('braininjury', 2),
('camilacabello', 2),
('BHRAMABULL', 2),
('BlowMandyUp', 2),
('iTunesMusic', 2),
('NickCannon', 2),
('auspol', 2),
('PeterDutton', 2),
('CecilTheLion', 2),
('zerohedge', 2),
('worldnetdaily', 2),

('OBLITERATE', 2),
('fukushima', 2),
('lulgzimbestpicts', 2),
('RAmag', 2),
('DiamondKesawn', 2),
('RaynbowAffair', 2),
('anyname', 2),
('matako', 2),
('PantherAttack', 2),
('YoungHeroesID', 2),
('cheyenne', 2),
('rapidcity', 2),
('StandwithPP', 2),
('PlannedParenthood', 2),
('calgaryweather', 2),
('selfie', 2),
('mkx', 2),
('noobde', 2),
('versethe', 2),
('IranDeal', 2),
('tubestrike', 2),
('weallheartonedirection', 2),
('thankU', 2),
('GodsLove', 2),
('nothe', 2),
('FAMEMOP', 2),
('BILLDANZEMOP', 2),
('RondaRousey', 2),
('wusa', 2),
('wmata', 2),
('quivk', 2),
('PumpkinMari', 2),
('OpTic', 2),
('DefundPP', 2),
('isecond', 2),
('favourite', 2),
('ScorpionCBS', 2),
('JoeNBC', 2),
('nikeplus', 2),
('Unkn', 2),
('Trfc', 2),
('LiteraryCakes', 2),
('edsheeran', 2),
('PPact', 2),
('SmantiBatam', 2),
('SenTedCruz', 2),
('GOPDebate', 2),
('Vpzedd', 2),
('KurtSchlichter', 2),
('AdamRubinESPN', 2),
('TrueHeroes', 2),
('denisleary', 2),
('followme', 2),
('ComplexMag', 2),
('XLEAK', 2),
('BioTerrorism', 2),
('usatoday', 2),
('TheAdvocateMag', 2),
('SOSFAM', 2),
('Beyhive', 2),
('Prepper', 2),
('AskConnor', 2),
('thehill', 2),
('nsfw', 2),
('CityofCalgary', 1),
('XrWn', 1),
('FASTENERS', 1),
('Rajman', 1),
('MsOreo', 1),
('EbolaOutbreak', 1),
('stighefootball', 1),
('remus', 1),
('jsyk', 1),
('thrillhho', 1),
('yakubOObs', 1),
('dtom', 1),
('ndtv', 1),

('janisctv', 1),
('Herologist', 1),
('whatstheimportantvideo', 1),
('ClaudioMeloni', 1),
('xxxmrbootleg', 1),
('facialabuse', 1),
('DukeSkywalker', 1),
('carsonmwr', 1),
('FtCarsonPAO', 1),
('offdishduty', 1),
('RunningRoom', 1),
('WorldRunners', 1),
('YahooNews', 1),
('forgivenwife', 1),
('kerricktrial', 1),
('GoogleFacts', 1),
('bkanioros', 1),
('travelled', 1),
('helloimivan', 1),
('lordjewcup', 1),
('suspe', 1),
('Miloko', 1),
('rhodeisland', 1),
('PQPHistoryWeekend', 1),
('Presqu', 1),
('WINDSTORM', 1),
('easternoregon', 1),
('LakeCounty', 1),
('smHRN', 1),
('IDFire', 1),
('Alynacarol', 1),
('chelan', 1),
('WAwildfire', 1),
('catalinas', 1),
('fingerrockfire', 1),
('taviiikenney', 1),
('chattanooga', 1),
('therealmattleaf', 1),
('JacobHoggard', 1),
('BeingAuthor', 1),
('BYNR', 1),
('MrRat', 1),
('myralynn', 1),
('NetNewsLedger', 1),
('lanahillman', 1),
('HarperANetflixShow', 1),
('newnewnew', 1),
('getitbeforeitsgone', 1),
('freshoutofthebox', 1),
('YOUKU', 1),
('amwriting', 1),
('WhirlWind', 1),
('Louisan', 1),
('BarstoolBigCat', 1),
('kunais', 1),
('Bokoharm', 1),
('upgrad', 1),
('SIDEARM', 1),
('marstu', 1),
('deray', 1),
('DANNYonPC', 1),
('JPens', 1),
('usthe', 1),
('Nowlike', 1),
('sindy', 1),
('KnaveTBE', 1),
('boyxboy', 1),
('vickysuewrites', 1),
('fucboi', 1),
('Slosher', 1),
('splattling', 1),
('slosher', 1),
('callofmini', 1),
('TerrellT', 1),
('TeamDuval', 1),
('onshit', 1),
('seasonfrom', 1)

('seasonfrom', 1),
('dandre', 1),
('thelovatoagent', 1),
('epicente', 1),
('ketep', 1),
('KETEP', 1),
('idgaf', 1),
('eevans', 1),
('Jesssssssee', 1),
('sicklife', 1),
('CatholicMomVA', 1),
('grey', 1),
('shellno', 1),
('yycweather', 1),
('Axegressor', 1),
('kjc', 1),
('calgarysun', 1),
('suddendly', 1),
('approachng', 1),
('evng', 1),
('Sumthng', 1),
('hippieXox', 1),
('RoxYunkFalls', 1),
('EllenPompeo', 1),
('shondarhimes', 1),
('ABCNetwork', 1),
('QWkD', 1),
('Permutable', 1),
('livescience', 1),
('pVDRDLJvc', 1),
('MGSV', 1),
('ArvindKejriwal', 1),
('fJiZzTxK', 1),
('SushmaSwaraj', 1),
('WorldNews', 1),
('rumblevideo', 1),
('Makurazaki', 1),
('KIPMOOREMUSIC', 1),
('bamagaga', 1),
('kady', 1),
('recentlu', 1),
('lizjillies', 1),
('BBMeg', 1),
('dougiedabandit', 1),
('RhyareeGaming', 1),
('Zelse', 1),
('Maliceqt', 1),
('LPlays', 1),
('STLCards', 1),
('Walktrough', 1),
('DArchambau', 1),
('TerriF', 1),
('Illusionimagess', 1),
('TheDIYHacks', 1),
('AshNiggas', 1),
('kian', 1),
('technews', 1),
('kmurphalurph', 1),
('corcoran', 1),
('BestSeller', 1),
('TCMParty', 1),
('SummerUnderTheStars', 1),
('ShallWeDance', 1),
('TitusOReily', 1),
('ibAUTUMN', 1),
('HRWright', 1),
('TomCenci', 1),
('DanDoherty', 1),
('SemiMooch', 1),
('lovestheworld', 1),
('sunakawa', 1),
('WestEndShiv', 1),
('Natalieala na', 1),
('lisajdavidson', 1),
('Williamso', 1),
('michellemccann', 1),
('JamesBond', 1),
('DIDN', 1),

('DIDN', 1),
('TremendousTroye', 1),
('judithabarrow', 1),
('IPOK', 1),
('IFAK', 1),
('sageuk', 1),
('gnr', 1),
('marijuananews', 1),
('AngstAttack', 1),
('onbeingwithKristaTippett', 1),
('Susancares', 1),
('Beingtweets', 1),
('MountSinaiNYC', 1),
('newauthors', 1),
('Clevel', 1),
('sickmund', 1),
('JSETT', 1),
('JetsetExtra', 1),
('WOZNI', 1),
('holibobs', 1),
('Pandamoanimum', 1),
('Mottas', 1),
('topstories', 1),
('slrqxcrls', 1),
('SheilaGunnReid', 1),
('LauraE', 1),
('jiwa', 1),
('remedyyyy', 1),
('Idgaf', 1),
('MindfulYoga', 1),
('enjoltair', 1),
('thebookclub', 1),
('libertyville', 1),
('sarahnandola', 1),
('MarquisDeSpade', 1),
('OliLafont', 1),
('alwx', 1),
('HeadlinesApp', 1),
('cancelled', 1),
('myswc', 1),
('ithats', 1),
('icouldsitinthismomentforever', 1),
('FatLoss', 1),
('Dievas', 1),
('Praam', 1),
('DarshanRavalDZ', 1),
('szuniverse', 1),
('MetroFmTalk', 1),
('buzzfeed', 1),
('sethiankit', 1),
('RegaJha', 1),
('sympathising', 1),
('Presstitutes', 1),
('anoder', 1),
('SculrismFinis', 1),
('takig', 1),
('Bstrd', 1),
('bldy', 1),
('MetrofmTalk', 1),
('wololo', 1),
('Grolsh', 1),
('Tltltltltltttlt', 1),
('sympathyONLF', 1),
('BOKO', 1),
('terrorismturn', 1),
('radicalisation', 1),
('BBCNewsAsia', 1),
('BehindTheScenes', 1),
('GeorgeTakei', 1),
('TheNerdsofColor', 1),
('penelopephoto', 1),
('SyedIHusain', 1),
('WomensWeeklyMag', 1),
('thejournal', 1),
('yogal', 1),
('Incubustour', 1),
('SurvivorsGuidetoEarth', 1),
('thematthewgeeks', 1),

('themattnewcooke', 1),
('incubusband', 1),
('croakeyblog', 1),
('juliepower', 1),
('justgetawayx', 1),
('Citymapper', 1),
('Babypicturethis', 1),
('Fotoset', 1),
('BombEffects', 1),
('lanparty', 1),
('idkidk', 1),
('wildestsdream', 1),
('mperegrym', 1),
('RookieBlue', 1),
('Rookiebluetv', 1),
('energyqfor', 1),
('thezlong', 1),
('TheWalkingDead', 1),
('amyschumer', 1),
('Idek', 1),
('ASOLIDRIGHTHOOK', 1),
('Birdyword', 1),
('jeezus', 1),
('ZAYN', 1),
('stelena', 1),
('crewlist', 1),
('Kumasian', 1),
('VengefulKnave', 1),
('TPGazette', 1),
('GabbyOgbechie', 1),
('Qaryatayn', 1),
('parallelpond', 1),
('trensabby', 1),
('wherepond', 1),
('Shahasda', 1),
('nomar', 1),
('Craigyb', 1),
('alvin', 1),
('mehdirhasan', 1),
('isuicide', 1),
('PieroCastellano', 1),
('KILLEDSOMEBODY', 1),
('GODJESUS', 1),
('WASTHE', 1),
('samfbiddle', 1),
('Zirngast', 1),
('MyyTwoCentss', 1),
('JRehling', 1),
('detona', 1),
('ispecial', 1),
('ZeeNews', 1),
('qtSZg', 1),
('GazelleBundchen', 1),
('BBCLive', 1),
('TonyAbbottMHR', 1),
('paulmyerscough', 1),
('Rightways', 1),
('georgegallagher', 1),
('JimmieJohnson', 1),
('KEGm', 1),
('aiden', 1),
('Rexyy', 1),
('TheJasonTaylorR', 1),
('niggaeven', 1),
('pctool', 1),
('SDotJR', 1),
('periodstories', 1),
('timeshighered', 1),
('Greymane', 1),
('DovahSwag', 1),
('DoingHashtagsRight', 1),
('KyleSalive', 1),
('ThrowbackThursday', 1),
('asda', 1),
('momentumcamps', 1),
('SnowBackSunday', 1),
('WhistlerBlckcmb', 1),
('Ben2f', 1),

('DonDr', 1),
('URTY', 1),
('treeporn', 1),
('LakeEffect', 1),
('kellyannwx', 1),
('irishirr', 1),
('snowstormI', 1),
('erincandy', 1),
('BigBang', 1),
('RTphotographyUK', 1),
('writerslife', 1),
('woodelijah', 1),
('NaturalBirth', 1),
('zouis', 1),
('UnusualWords', 1),
('misocapnist', 1),
('wizkhalifa', 1),
('spookyerica', 1),
('fckng', 1),
('jassy', 1),
('PEDO', 1),
('escuchando', 1),
('hago', 1),
('BobbyCakes', 1),
('HusnaaVhora', 1),
('nko', 1),
('Crhedrys', 1),
('Kronykal', 1),
('defreitas', 1),
('manyvids', 1),
('vscodc', 1),
('dctography', 1),
('acreativedc', 1),
('VSCOcam', 1),
('TheBEACHDC', 1),
('buildingmuseum', 1),
('realise', 1),
('edjschenk', 1),
('andreajmarkley', 1),
('NewsBeatSocial', 1),
('Statoilasa', 1),
('oilandgas', 1),
('DETECTADO', 1),
('WorldOil', 1),
('deetelecare', 1),
('AxWave', 1),
('ysl', 1),
('Seismicsoftware', 1),
('Thrusta', 1),
('elephino', 1),
('OMGGGG', 1),
('Hozier', 1),
('melodores', 1),
('CitiOpen', 1),
('ATPWorldTour', 1),
('ShojoShit', 1),
('HANAYO', 1),
('sugayiffer', 1),
('simplyysacred', 1),
('McKenzieBlackw', 1),
('AVOCADO', 1),
('WHERES', 1),
('MeganRestivo', 1),
('LordMinion', 1),
('GabrielleAplin', 1),
('burna', 1),
('yaaasss', 1),
('HSBFUCKJSJ', 1),
('ArianaGrande', 1),
('gotdamn', 1),
('AKFNEJF', 1),
('TroyLWiggins', 1),
('CokeBoys', 1),
('TopherBreezy', 1),
('Verrry', 1),
('ganwilson', 1),
('frailnerves', 1),
('YUSKAY', 1),

('XUSKAK', 1),
('SBUJDJSJS', 1),
('OOORRRR', 1),
('beachboyIrh', 1),
('OMLETTE', 1),
('DIDNT', 1),
('OTRAMETLIFE', 1),
('YESSSS', 1),
('theboysftvines', 1),
('AUNTS', 1),
('joshshatsays', 1),
('Wahlburgers', 1),
('redwedding', 1),
('WHYYYY', 1),
('AmazingRaceCanada', 1),
('jessemontani', 1),
('montani', 1),
('wouldarude', 1),
('centralupload', 1),
('LuchaUnderground', 1),
('UltimaLucha', 1),
('FORREAL', 1),
('fUCK', 1),
('peens', 1),
('Jfc', 1),
('OrianaArzola', 1),
('gerenciatodos', 1),
('accionempresa', 1),
('LyricalEyes', 1),
('jayElectronica', 1),
('melodyJKT', 1),
('PemantauJKT', 1),
('badcredit', 1),
('UndergroundRailraod', 1),
('YAHIstorical', 1),
('CindyNoonan', 1),
('runjewels', 1),
('DebateQuestionsWeWantToHear', 1),
('CleanPowerPlan', 1),
('WakeUpAmerica', 1),
('brownblaze', 1),
('foxandfriends', 1),
('dahboo', 1),
('abbydphillip', 1),
('Allahuakbar', 1),
('amiestager', 1),
('kog', 1),
('shootn', 1),
('bleased', 1),
('Ulzzang', 1),
('ModiMinistry', 1),
('JoshLeeKwai', 1),
('MtgoDoc', 1),
('jfwong', 1),
('OwenHarris', 1),
('AlexandBondarev', 1),
('CazorlaPlay', 1),
('CHold', 1),
('theonion', 1),
('carryi', 1),
('nigeriantribune', 1),
('KOJIMA', 1),
('HIDEO', 1),
('FusionFestival', 1),
('RockyFire', 1),
('FirstAid', 1),
('zix', 1),
('MvDint', 1),
('NoAgenda', 1),
('LesMcKeownUK', 1),
('aaronkearneyaus', 1),
('wcvh', 1),
('animalrescue', 1),
('NickWilson', 1),
('WolvenBeauty', 1),
('Meowing', 1),
('zacktiller', 1),

```
('bcook', 1),
('WhiteGenocide', 1),
('dirtylying', 1),
('whao', 1),
('SyrianRefugees', 1),
('changetheworld', 1),
('wowo', 1),
('honeystaysuper', 1),
('RazedOnIt', 1),
('RazeD', 1),
('monterrey', 1),
('SavanahResnik', 1),
('WerdEmUp', 1),
('petrichour', 1),
('BRINING', 1),
('RAINSTORM', 1),
('amandagiroux', 1),
('ChaseIngram', 1),
('famoushorse', 1),
('mcgtech', 1),
('hjt', 1),
('Changelessly', 1),
('kirillzubovsky', 1),
('astolfialessio', 1),
('advancedwarfare', 1),
('callofduty', 1),
('atk', 1),
('letsFootball', 1),
('chickmt', 1),
('longshipdriver', 1),
('chanelwestcoast', 1),
('tcbinaflash', 1),
('MbbLive', 1),
('LateNiteMix', 1),
('PhilCollinsFeed', 1),
('MuckRock', 1),
('organisation', 1),
('Sitc', 1),
('milefromhome', 1),
('RudyHavenstein', 1),
('estus', 1),
('GuerrillaDawg', 1),
('ScarFacedCully', 1),
('snapharmony', 1),
('BizInsider', 1),
('bexrayandvav', 1),
('underrrtow', 1),
('meca', 1),
('candydulfer', 1),
('puzzledtriangle', 1),
('SHTFPlan', 1),
('SixSecondCov', 1),
('Kayhow', 1),
('Ogbor', 1),
('IBLMapleLeafs', 1),
('misslyndaleigh', 1),
('SEBEE', 1),
('BANKSTOWN', 1),
('disclos', 1),
('globetrottingwino', 1),
('Hangarback', 1),
('RENDERINGS', 1),
('OldFolkExplainStuff', 1),
('KDonhoops', 1),
('bcwilliams', 1),
('bradohearne', 1),
('BlacklivesMatter', 1),
('mindkiller', 1),
('InsaneLimits', 1),
('XMwTE', 1),
('schematization', 1),
('Nuks', 1),
('realised', 1),
('Makrina', 1),
('RomanGaul', 1),
('LeonC', 1),
('dinnerwithjulie', 1),
```

```
('Roquey', 1),
('AllOutAsh', 1),
('KoscielnyFC', 1),
('Eunhae', 1),
('Chullie', 1),
('Hyukkie', 1),
('ohnoeunhae', 1),
('HEYOOO', 1),
('ThunderCastTV', 1),
('Wadeycakes', 1),
('youcantsitwithus', 1),
('beez', 1),
('bhill', 1),
('itsTiimothy', 1),
('AmericanJewry', 1),
('lionofjudah', 1),
('mirajane', 1),
('TheIranDeal', 1),
('PauloSergioMDC', 1),
('MilesWithTate', 1),
('JimmyFallon', 1),
('DrawLiomDraw', 1),
('BattyAfterDawn', 1),
('TrinityFox', 1),
('PAYER', 1),
('ONOFRE', 1),
('AlbertBrooks', 1),
('juhasipila', 1),
('ollirehn', 1),
('JukkaOksaharju', 1),
('nuclearrcSA', 1),
('Auspol', 1),
('ThorCon', 1),
('NuclearPower', 1),
('MaximEristavi', 1),
('Lacci', 1),
('fubiz', 1),
('WhiteTerrorism', 1),
('NoNukes', 1),
('truthfrequencyradio', 1),
('HARDA', 1),
('jamucsb', 1),
('jonk', 1),
('ErasureIsNotEquality', 1),
('Karnythia', 1),
('ProBonoNews', 1),
('JakartaPost', 1),
('ghaccount', 1),
('patrickandsam', 1),
('toopainful', 1),
('bottleowhite', 1),
('melindahaunton', 1),
('curryspcworld', 1),
('ineedcake', 1),
('greatbritishbakeoff', 1),
('picniclester', 1),
('gbbo', 1),
('blackforestgateau', 1),
('tyre', 1),
('BBCOne', 1),
('BritishBakeOff', 1),
('BakeOffFriends', 1),
('AlexxPage', 1),
('jonathanserrie', 1),
('GorillaThumbz', 1),
('SHEATH', 1),
('BAYONET', 1),
('PHROBIS', 1),
('gatewaypundit', 1),
('SocialJerkBlog', 1),
('weakley', 1),
('OILER', 1),
('Militarydotcom', 1),
('Btm', 1),
('SBJohn', 1),
('ChelseaVPeretti', 1),
('GRAMZD', 1),
```

```
('irongiant', 1),
('kumailn', 1),
('RonFunches', 1),
('TimCook', 1),
('ameltdown', 1),
('rymathieson', 1),
('civilwar', 1),
('sebastianstanisaliveandwell', 1),
('Madsummer', 1),
('julescheff', 1),
('subpop', 1),
('METZtheband', 1),
('featcha', 1),
('NowYouKnow', 1),
('amayhem', 1),
('livemusic', 1),
('concertphotography', 1),
('MamanyaDana', 1),
('IAmMrCash', 1),
('ynovak', 1),
('TheMGWVboss', 1),
('FOLLOWnGAIN', 1),
('RETWEET', 1),
('RobertoSAGuedes', 1),
('TheDreamingGoat', 1),
('Bdays', 1),
('ESWEnrage', 1),
('Terraria', 1),
('TomLandry', 1),
('showgirldayoff', 1),
('TimesNewsdesk', 1),
('Yougslavia', 1),
('kermit', 1),
('xMissXanthippex', 1),
('stefsy', 1),
('QuantumDataInformatics', 1),
('Muumbo', 1),
('FromTheDesk', 1),
('TheaterTrial', 1),
('CatoInstitute', 1),
('gojam', 1),
('AParra', 1),
('VPSay', 1),
('EmekaGift', 1),
('RiceeChrispies', 1),
('CamPepperr', 1),
('llr', 1),
('MishaWeller', 1),
('LetsBe', 1),
('Cynicalreality', 1),
('FaroeIslands', 1),
('FredOlsenCruise', 1),
('Conservatexian', 1),
('Rawrheem', 1),
('newscomauHQ', 1),
('musim', 1),
('tanstaafl', 1),
('RedScareBot', 1),
('absol', 1),
('AMAAS', 1),
('JohnMtaita', 1),
('ijreview', 1),
('OfGod', 1),
('tarmineta', 1),
('shrnnclbautista', 1),
('kiilud', 1),
('TksgS', 1),
('iBliz', 1),
('reba', 1),
('Followtheblonde', 1),
('LtDalius', 1),
('kWXGgt', 1),
('multitudinal', 1),
('MFi', 1),
('iCASEIT', 1),
('rebeccafav', 1),
('JacksonLaurie', 1),
```

```
('laneyslife', 1),
('jace', 1),
('jamisonevann', 1),
('gracemccreave', 1),
('DOMINOS', 1),
('NRmawa', 1),
('Niastewart', 1),
('Souuul', 1),
('HandmadeJewelry', 1),
('etsymntt', 1),
('AlienatePlays', 1),
('kiia', 1),
('Mirnawan', 1),
('bestdayeva', 1),
('Lavamobiles', 1),
('KoFapp', 1),
('kherr', 1),
('Instiinctz', 1),
('jGIsAvq', 1),
('CecilTownship', 1),
('lides', 1),
('QCt', 1),
('RakestrawJeff', 1),
('MartinMJ', 1),
('seeiey', 1),
('chihaya', 1),
('chinmaykrvd', 1),
('kdudakia', 1),
('DKHQgv', 1),
('TommyGShow', 1),
('Mhl', 1),
('womanxking', 1),
('MARSAC', 1),
('EPAO', 1),
('INUNDATED', 1),
('alexeeles', 1),
('Rossifumibhai', 1),
('NotSorry', 1),
('UseYourWords', 1),
('WordoftheDay', 1),
('Dictionarycom', 1),
('TypeEd', 1),
('TheBrooklynLife', 1),
('jokethey', 1),
('Splats', 1),
('eliistender', 1),
('keypsters', 1),
('nflweek', 1),
('uiseful', 1),
('Tyuler', 1),
('JusticeDotOrg', 1),
('amild', 1),
('autoaccidents', 1),
('SteveGursten', 1),
('MichiganAutoLaw', 1),
('birdgang', 1),
('ringsau', 1),
('sunstar', 1),
('Mva', 1),
('JaeSmoothCYB', 1),
('AZCardinals', 1),
('nflnetwork', 1),
('NYDNSports', 1),
('RVacchianoNYDN', 1),
('yeg', 1),
('EdmontonEsks', 1),
('wral', 1),
('wouldud', 1),
('brelsford', 1),
('ChristianStec', 1),
('LoganMeadows', 1),
('hegot', 1),
('SkinsOn', 1),
('Karjo', 1),
('sothathappened', 1),
('NewsBreaker', 1),
('BPGVII', 1),
```

```
('LCOutOfDoors', 1),
('leeranaldo', 1),
('LOOPING', 1),
('ReinhardBonnke', 1),
('slix', 1),
('habor', 1),
('thegoon', 1),
('amay', 1),
('AngelRiveraLib', 1),
('WatchmanIS', 1),
('Deosl', 1),
('NationFirst', 1),
('Iamtssudhir', 1),
('IndiaToday', 1),
('HLPS', 1),
('GUTRKPS', 1),
('HOSTAGESTHROSW', 1),
('INSPCTKPS', 1),
('BILNO', 1),
('MILTARY', 1),
('SNCTIONS', 1),
('RBpK', 1),
('Worthfull', 1),
('bluecurls', 1),
('BidTime', 1),
('theCHIVE', 1),
('TeamNyle', 1),
('damballa', 1),
('crunchysensible', 1),
('chacos', 1),
('RandomActsOfRomance', 1),
('TJGreaterSJ', 1),
('TJProvincial', 1),
('DailyGleaner', 1),
('aunft', 1),
('ElasticaInc', 1),
('nonononono', 1),
('ReasonVsFear', 1),
('timas', 1),
('NuNus', 1),
('MomentsAtHill', 1),
('eatshit', 1),
('csgpelmash', 1),
('RealDB', 1),
('liveleakfun', 1),
('SahelNews', 1),
('YKJL', 1),
('JagexSupport', 1),
('newsbayarea', 1),
('CentreTransfer', 1),
('ellmore', 1),
('tickens', 1),
('donbas', 1),
('daehyvn', 1),
('malyasiaairlines', 1),
('Jps', 1),
('Amase', 1),
('Himika', 1),
('QelricDK', 1),
('Humaza', 1),
('investigatin', 1),
('AaronGoodwin', 1),
('NickGroff', 1),
('Vickygeex', 1),
('emmap', 1),
('skie', 1),
('spikepoint', 1),
('WalpurgisNightZ', 1),
('facilitiesmanagement', 1),
...]
```

after checking we find that some words were not covered in vocabulary because of spelling mistakes, words were combined together, accronyms etc so included some of them manually, the perecentage increased by small amount, other words can be handled to increase the percentage coverage

This is done so that instead of manually going throug all the tweets to hndle the words which is impossible if the data is very large and time consuming we can just checking which words are not covered based on glove and only handle those

In [ ]:

```
train.text = train.text.apply(lambda x : preprocessing(x))
test.text = test.text.apply(lambda x : preprocessing(x))

train_glove_oov, train_glove_vocab_coverage, train_glove_text_coverage,train_embd =
check_embeddings_coverage(train['text'], w2v_model)
test_glove_oov, test_glove_vocab_coverage, test_glove_text_coverage,test_embd =
check_embeddings_coverage(test['text'], w2v_model)
print('GloVe Embeddings cover {:.2%} of vocabulary and {:.2%} of text in Training
Set'.format(train_glove_vocab_coverage, train_glove_text_coverage))
print('GloVe Embeddings cover {:.2%} of vocabulary and {:.2%} of text in Test
Set'.format(test_glove_vocab_coverage, test_glove_text_coverage))
```

```
GloVe Embeddings cover 79.42% of vocabulary and 92.39% of text in Training Set
GloVe Embeddings cover 82.56% of vocabulary and 91.86% of text in Test Set
```

Glove embedding + LSTM

we use glove embedding to embedd for LSTM

In [ ]:

```
embedding_dict = {}
embedding_dict.update(train_embd)
embedding_dict.update(test_embd)
```

In [ ]:

```
len(embedding_dict)
```

Out[ ]:

```
18806
```

In [10]:

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding,LSTM,Dense,SpatialDropout1D, Dropout
from keras.initializers import Constant
from sklearn.model_selection import train_test_split
from keras.optimizers import Adam
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import numpy as np
```

In [ ]:

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[ ]:

```
True
```

In [ ]:

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out[ ]:

```
True
```

In [ ]:

```
df1 = pd.concat([train, test])
stop = set(stopwords.words("english"))

def create_corpus(df):
    corpus = []
    for tweet in df["text"]:
        words = [word.lower() for word in word_tokenize(tweet) if \
        ((word.isalpha() == 1) & (word not in stop))]
        corpus.append(words)

    return corpus

corpus = create_corpus(df1)
```

In [ ]:

```
MAX_LEN = 50
tokenizer_obj = Tokenizer()
tokenizer_obj.fit_on_texts(corpus)

sequences = tokenizer_obj.texts_to_sequences(corpus)

tweet_pad = pad_sequences(sequences,
                          maxlen = MAX_LEN,
                          truncating = 'post',
                          padding = 'post')
```

In [ ]:

```
word_index = tokenizer_obj.word_index
print('number of unique words: ', len(word_index))
```

```
number of unique words:  19630
```

In [ ]:

```
num_words = len(word_index) + 1
embedding_matrix = np.zeros((num_words,300))


for word, i in word_index.items():
    if i > num_words:
        continue

    embedding_vector = embedding_dict.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [ ]:

```
from keras import regularizers

model = Sequential()

glove_embedding = Embedding(num_words, 300, embeddings_initializer = Constant(embedding_matrix),
                    input_length = MAX_LEN,
                    trainable = False)

model.add(glove_embedding)
model.add(SpatialDropout1D(0.2))
```

```
model.add(LSTM(128, dropout = 0.2, recurrent_dropout = 0.2))
model.add(Dense(128, activation = 'relu', kernel_regularizer=regularizers.l1_l2(l1=1e-5, l2=1e-4)))
model.add(Dense(256, activation = 'relu', kernel_regularizer=regularizers.l1_l2(l1=1e-5, l2=1e-4)))
model.add(Dropout(0.2))
model.add(Dense(1, activation = 'sigmoid'))

optimizer = Adam(learning_rate=1e-5)

model.compile(loss = 'binary_crossentropy', optimizer = optimizer, metrics = ["accuracy"])
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernel since it doesn't meet the cuDNN kernel cri
teria. It will use generic GPU kernel as fallback when running on GPU
```

In [ ]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 50, 300)           5889300
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 50, 300)           0
_________________________________________________________________
lstm (LSTM)                  (None, 128)               219648
_________________________________________________________________
dense (Dense)                (None, 128)               16512
_________________________________________________________________
dense_1 (Dense)              (None, 256)               33024
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 6,158,741
Trainable params: 269,441
Non-trainable params: 5,889,300
_________________________________________________________________
```

In [ ]:

```
train_data = tweet_pad[:train.shape[0]]
test_data = tweet_pad[train.shape[0]:]
```

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(train_data, train["target_relabeled"].values, t
est_size = 0.15)
```

In [ ]:

```
hist = model.fit(X_train, y_train, batch_size = 64, epochs = 30, validation_data = (X_test, y_test)
)
```

```
Epoch 1/30
102/102 [==============================] - 21s 171ms/step - loss: 0.7553 - accuracy: 0.5570 - val_
loss: 0.7537 - val_accuracy: 0.5727
Epoch 2/30
102/102 [==============================] - 18s 172ms/step - loss: 0.7531 - accuracy: 0.5743 - val_
loss: 0.7515 - val_accuracy: 0.5727
Epoch 3/30
102/102 [==============================] - 17s 167ms/step - loss: 0.7511 - accuracy: 0.5706 - val_
loss: 0.7490 - val_accuracy: 0.5727
Epoch 4/30
102/102 [==============================] - 17s 168ms/step - loss: 0.7487 - accuracy: 0.5613 - val_
loss: 0.7429 - val_accuracy: 0.5727
Epoch 5/30
102/102 [==============================] - 17s 170ms/step - loss: 0.7296 - accuracy: 0.5657 - val_
loss: 0.6281 - val_accuracy: 0.7277
Epoch 6/30
```

```
102/102 [==============================] - 17s 169ms/step - loss: 0.6384 - accuracy: 0.7427 - val_
loss: 0.5819 - val_accuracy: 0.7863
Epoch 7/30
102/102 [==============================] - 17s 170ms/step - loss: 0.5974 - accuracy: 0.7686 - val_
loss: 0.5440 - val_accuracy: 0.7872
Epoch 8/30
102/102 [==============================] - 17s 170ms/step - loss: 0.5661 - accuracy: 0.7682 - val_
loss: 0.5267 - val_accuracy: 0.7925
Epoch 9/30
102/102 [==============================] - 17s 168ms/step - loss: 0.5474 - accuracy: 0.7698 - val_
loss: 0.5197 - val_accuracy: 0.7907
Epoch 10/30
102/102 [==============================] - 18s 173ms/step - loss: 0.5516 - accuracy: 0.7703 - val_
loss: 0.5139 - val_accuracy: 0.7916
Epoch 11/30
102/102 [==============================] - 17s 167ms/step - loss: 0.5361 - accuracy: 0.7800 - val_
loss: 0.5116 - val_accuracy: 0.7951
Epoch 12/30
102/102 [==============================] - 17s 167ms/step - loss: 0.5279 - accuracy: 0.7856 - val_
loss: 0.5067 - val_accuracy: 0.7907
Epoch 13/30
102/102 [==============================] - 17s 167ms/step - loss: 0.5264 - accuracy: 0.7913 - val_
loss: 0.5048 - val_accuracy: 0.8004
Epoch 14/30
102/102 [==============================] - 17s 169ms/step - loss: 0.5280 - accuracy: 0.7895 - val_
loss: 0.5028 - val_accuracy: 0.8004
Epoch 15/30
102/102 [==============================] - 17s 171ms/step - loss: 0.5326 - accuracy: 0.7828 - val_
loss: 0.5003 - val_accuracy: 0.8004
Epoch 16/30
102/102 [==============================] - 17s 169ms/step - loss: 0.5256 - accuracy: 0.7837 - val_
loss: 0.4983 - val_accuracy: 0.8039
Epoch 17/30
102/102 [==============================] - 17s 169ms/step - loss: 0.5122 - accuracy: 0.7934 - val_
loss: 0.4965 - val_accuracy: 0.8039
Epoch 18/30
102/102 [==============================] - 17s 168ms/step - loss: 0.5192 - accuracy: 0.7968 - val_
loss: 0.4944 - val_accuracy: 0.8047
Epoch 19/30
102/102 [==============================] - 17s 169ms/step - loss: 0.5190 - accuracy: 0.7932 - val_
loss: 0.4929 - val_accuracy: 0.8100
Epoch 20/30
102/102 [==============================] - 17s 169ms/step - loss: 0.5138 - accuracy: 0.7916 - val_
loss: 0.4955 - val_accuracy: 0.8074
Epoch 21/30
102/102 [==============================] - 17s 169ms/step - loss: 0.5198 - accuracy: 0.7868 - val_
loss: 0.4911 - val_accuracy: 0.8091
Epoch 22/30
102/102 [==============================] - 17s 167ms/step - loss: 0.5025 - accuracy: 0.8012 - val_
loss: 0.4904 - val_accuracy: 0.8109
Epoch 23/30
102/102 [==============================] - 17s 168ms/step - loss: 0.5111 - accuracy: 0.7945 - val_
loss: 0.4896 - val_accuracy: 0.8117
Epoch 24/30
102/102 [==============================] - 18s 171ms/step - loss: 0.4992 - accuracy: 0.7928 - val_
loss: 0.4879 - val_accuracy: 0.8109
Epoch 25/30
102/102 [==============================] - 17s 170ms/step - loss: 0.5169 - accuracy: 0.7853 - val_
loss: 0.4866 - val_accuracy: 0.8109
Epoch 26/30
102/102 [==============================] - 17s 170ms/step - loss: 0.5069 - accuracy: 0.7940 - val_
loss: 0.4866 - val_accuracy: 0.8117
Epoch 27/30
102/102 [==============================] - 18s 172ms/step - loss: 0.4968 - accuracy: 0.7989 - val_
loss: 0.4854 - val_accuracy: 0.8109
Epoch 28/30
102/102 [==============================] - 17s 171ms/step - loss: 0.5050 - accuracy: 0.7921 - val_
loss: 0.4858 - val_accuracy: 0.8100
Epoch 29/30
102/102 [==============================] - 17s 170ms/step - loss: 0.4990 - accuracy: 0.8029 - val_
loss: 0.4843 - val_accuracy: 0.8109
Epoch 30/30
102/102 [==============================] - 17s 165ms/step - loss: 0.4889 - accuracy: 0.8046 - val_
loss: 0.4844 - val_accuracy: 0.8100
```

In [ ]:

```
test["target"]=model.predict(test_data).round().astype(int)
```

In [ ]:

```
submission = test[["id", "target"]]
submission.to_csv("drive/MyDrive/Colab Notebooks/Amazon/submission1.csv",index=False)
```

BERT

In [11]:

```
from transformers import TFAutoModel, AutoTokenizer
```

In [12]:

```
def encode_tweets(tokenizer, tweets, max_len):
    nb_tweets = len(tweets)
    tokens = np.ones((nb_tweets,max_len),dtype='int32')
    masks = np.zeros((nb_tweets,max_len),dtype='int32')
    segs = np.zeros((nb_tweets,max_len),dtype='int32')

    for k in range(nb_tweets):
        # INPUT_IDS
        tweet = tweets[k]
        enc = tokenizer.encode(tweet)
        if len(enc) < max_len-2:
            tokens[k,:len(enc)+2] = [0] + enc + [2]
            masks[k,:len(enc)+2] = 1
        else:
            tokens[k,:max_len] = [0] + enc[:max_len-2] + [2]
            masks[k,:max_len] = 1
    return tokens,masks,segs
```

In [13]:

```
MAX_LEN=64
def build_model(max_len):
    ids = Input((max_len,), dtype=tf.int32)
    attention = Input((max_len,), dtype=tf.int32)
    token = Input((max_len,), dtype=tf.int32)

    bertweet = TFAutoModel.from_pretrained("vinai/bertweet-base")
    x= bertweet(ids,attention_mask=attention,token_type_ids=token)

    out = Dense(1,activation='sigmoid')(x[0][:,0,:])

    model = Model(inputs=[ids, attention, token], outputs = out)
    optimizer = Adam(learning_rate=1e-5)
    model.compile(loss='binary_crossentropy',
                  optimizer=optimizer,
                  metrics=['accuracy'])
    return model


model = build_model(MAX_LEN)
model.summary()
```

```
Some layers from the model checkpoint at vinai/bertweet-base were not used when initializing TFRob
ertaModel: ['lm_head']
- This IS expected if you are initializing TFRobertaModel from the checkpoint of a model trained o
n another task or with another architecture (e.g. initializing a BertForSequenceClassification
model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFRobertaModel from the checkpoint of a model that
you expect to be exactly identical (initializing a BertForSequenceClassification model from a
BertForSequenceClassification model).
All the layers of TFRobertaModel were initialized from the model checkpoint at vinai/bertweet-base
.
If your task is similar to the task the model of the checkpoint was trained on, you can already us
e TFRobertaModel for predictions without further training.
```

```
WARNING:tensorflow:AutoGraph could not transform <bound method Socket.send of
<zmq.sugar.socket.Socket object at 0x7fb8cbf04660>> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux,
`export AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: <cyfunction Socket.send at 0x7fb8e374fe58> is not a module, class, method, function,
traceback, frame, or code object
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING: AutoGraph could not transform <bound method Socket.send of <zmq.sugar.socket.Socket objec
t at 0x7fb8cbf04660>> and will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the verbosity to 10 (on Linux,
`export AUTOGRAPH_VERBOSITY=10`) and attach the full output.
Cause: <cyfunction Socket.send at 0x7fb8e374fe58> is not a module, class, method, function,
traceback, frame, or code object
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
```

```
The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when
calling a model.They have to be set to True/False in the config object (i.e.:
`config=XConfig.from_pretrained('name', output_attentions=True)`).
```

```
WARNING:tensorflow:AutoGraph could not transform <function wrap at 0x7fb8e10e38c8> and will run it
as-is.
Cause: while/else statement not yet supported
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
```

```
The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
```

```
WARNING: AutoGraph could not transform <function wrap at 0x7fb8e10e38c8> and will run it as-is.
Cause: while/else statement not yet supported
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 64)]         0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, 64)]         0
__________________________________________________________________________________________________
input_3 (InputLayer)            [(None, 64)]         0
__________________________________________________________________________________________________
tf_roberta_model (TFRobertaMode TFBaseModelOutputWit 134899968   input_1[0][0]
                                                                 input_2[0][0]
                                                                 input_3[0][0]
__________________________________________________________________________________________________
tf.__operators__.getitem (Slici (None, 768)          0           tf_roberta_model[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 1)            769         tf.__operators__.getitem[0][0]
==================================================================================================
Total params: 134,900,737
Trainable params: 134,900,737
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [14]:

```
optimizer = Adam(learning_rate=1e-5)

model.compile(loss = 'binary_crossentropy', optimizer = optimizer, metrics = ["accuracy"])
```

In [15]:

```
tokenizer = AutoTokenizer.from_pretrained("vinai/bertweet-base")
```

```
emoji is not installed, thus not converting emoticons or emojis into text. Please install emoji: p
ip3 install emoji
Special tokens have been added in the vocabulary, make sure the associated word embedding are fine
-tuned or trained.
```

In [16]:

```
train_tokens, train_masks, train_segs = encode_tweets(tokenizer, train["text"].to_list(), MAX_LEN)
```

```
train_labels = train["target"]
```

In [23]:

```
test_tokens, test_masks, test_segs = encode_tweets(tokenizer, test["text"].to_list(), MAX_LEN)
```

In [17]:

```
from keras.callbacks import EarlyStopping, ModelCheckpoint

#CKPT = ModelCheckpoint('./ckpt.h5', save_best_only=True, monitor='val_loss', mode='min')
ES = EarlyStopping(monitor='val_loss', mode='min', patience=10, restore_best_weights=True, verbose=
1)
```

In [18]:

```
hist = model.fit([train_tokens,train_masks,train_segs], train_labels,
                 batch_size = 32, epochs = 4,
                 validation_split = 0.1, callbacks= [ES])
```

```
The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when
calling a model.They have to be set to True/False in the config object (i.e.:
`config=XConfig.from_pretrained('name', output_attentions=True)`).
The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
```

```
Epoch 1/4
WARNING:tensorflow:Gradients do not exist for variables
['tf_roberta_model/roberta/pooler/dense/kernel:0', 'tf_roberta_model/roberta/pooler/dense/bias:0']
when minimizing the loss.
```

```
The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when
calling a model.They have to be set to True/False in the config object (i.e.:
`config=XConfig.from_pretrained('name', output_attentions=True)`).
The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
```

```
WARNING:tensorflow:Gradients do not exist for variables
['tf_roberta_model/roberta/pooler/dense/kernel:0', 'tf_roberta_model/roberta/pooler/dense/bias:0']
when minimizing the loss.
215/215 [==============================] - ETA: 0s - loss: 0.5611 - accuracy: 0.7012
```

```
The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when
calling a model.They have to be set to True/False in the config object (i.e.:
`config=XConfig.from_pretrained('name', output_attentions=True)`).
The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
```

```
215/215 [==============================] - 109s 444ms/step - loss: 0.5607 - accuracy: 0.7015 - val
_loss: 0.3628 - val_accuracy: 0.8399
Epoch 2/4
215/215 [==============================] - 96s 446ms/step - loss: 0.3922 - accuracy: 0.8328 - val_
loss: 0.4497 - val_accuracy: 0.8005
Epoch 3/4
215/215 [==============================] - 96s 446ms/step - loss: 0.3859 - accuracy: 0.8394 - val_
loss: 0.3941 - val_accuracy: 0.8438
Epoch 4/4
215/215 [==============================] - 96s 445ms/step - loss: 0.3110 - accuracy: 0.8755 - val_
loss: 0.3766 - val_accuracy: 0.8399
```

In [24]:

```
test["target"] = model.predict([test_tokens, test_masks, test_segs]).round().astype(int)
submission = test[["id", "target"]]
submission.to_csv("drive/MyDrive/Colab Notebooks/Amazon/submission2.csv",index=False)
```

```
The parameters `output_attentions`, `output_hidden_states` and `use_cache` cannot be updated when
calling a model.They have to be set to True/False in the config object (i.e.:
`config=XConfig.from_pretrained('name', output_attentions=True)`).
The parameter `return_dict` cannot be set in graph mode and will always be set to `True`.
```

Conclusion

In [7]:

```
from prettytable import PrettyTable
```

In [8]:

```
pt=PrettyTable()
pt.field_names=["Architecture","Train Accuracy","CV Accuracy","Test Accuracy"]
pt.add_row(["Glove+LSTM","0.8046","0.8109","0.7912"])
pt.add_row(["BERT","0.839","0.8438","0.8374"])
print(pt)
```

```
+--------------+----------------+-------------+---------------+
| Architecture | Train Accuracy | CV Accuracy | Test Accuracy |
+--------------+----------------+-------------+---------------+
|  Glove+LSTM  |     0.8046     |    0.8109   |     0.7912    |
|     BERT     |     0.839      |    0.8438   |     0.8374    |
+--------------+----------------+-------------+---------------+
```

In [ ]:

```

```