In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries ins
talled
# It is defined by the kaggle/python docker image: https://github.com/kaggl
e/docker-python
# For example, here's several helpful packages to load in
import gc
import re
import nltk
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
np.random.seed(0)
from keras.models import Model
from keras.layers import Dense, Input, Dropout, LSTM, Activation
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.corpus import stopwords
from nltk.util import ngrams
from sklearn.feature_extraction.text import CountVectorizer
from collections import defaultdict
from collections import  Counter
plt.style.use('ggplot')
stop=set(stopwords.words('english'))
import re
from nltk.tokenize import word_tokenize
import gensim
import string
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from tqdm import tqdm
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional
, Dropout,SpatialDropout1D
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tqdm import tqdm
np.random.seed(1)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will
 list all files under the input directory
```

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
Using TensorFlow backend.
```

```
/kaggle/input/nlp-getting-started/test.csv
/kaggle/input/nlp-getting-started/sample_submission.csv
/kaggle/input/nlp-getting-started/train.csv
/kaggle/input/glove-global-vectors-for-word-representation/glove.6B.50
d.txt
/kaggle/input/glove-global-vectors-for-word-representation/glove.6B.10
0d.txt
/kaggle/input/glove-global-vectors-for-word-representation/glove.6B.20
0d.txt
```

In [2]:

```
train= pd.read_csv('../input/nlp-getting-started/train.csv')
test=pd.read_csv('../input/nlp-getting-started/test.csv')
train.head()
```

Out[2]:

	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

In [3]:
```
train.isnull().sum(axis=0)
```

Out[3]:
```
id             0
keyword       61
location    2533
text           0
target         0
dtype: int64
```

In [4]:
```
test.isnull().sum(axis=0)
```

Out[4]:
```
id             0
keyword       26
location    1105
text           0
dtype: int64
```

In [5]:
```
keyword_cnt = train.keyword.value_counts()
keyword_cnt
```

Out[5]:
```
fatalities              45
deluge                  42
armageddon              42
damage                  41
harm                    41
                        ..
forest%20fire           19
epicentre               12
threat                  11
inundation              10
radiation%20emergency    9
Name: keyword, Length: 221, dtype: int64
```

In [6]:

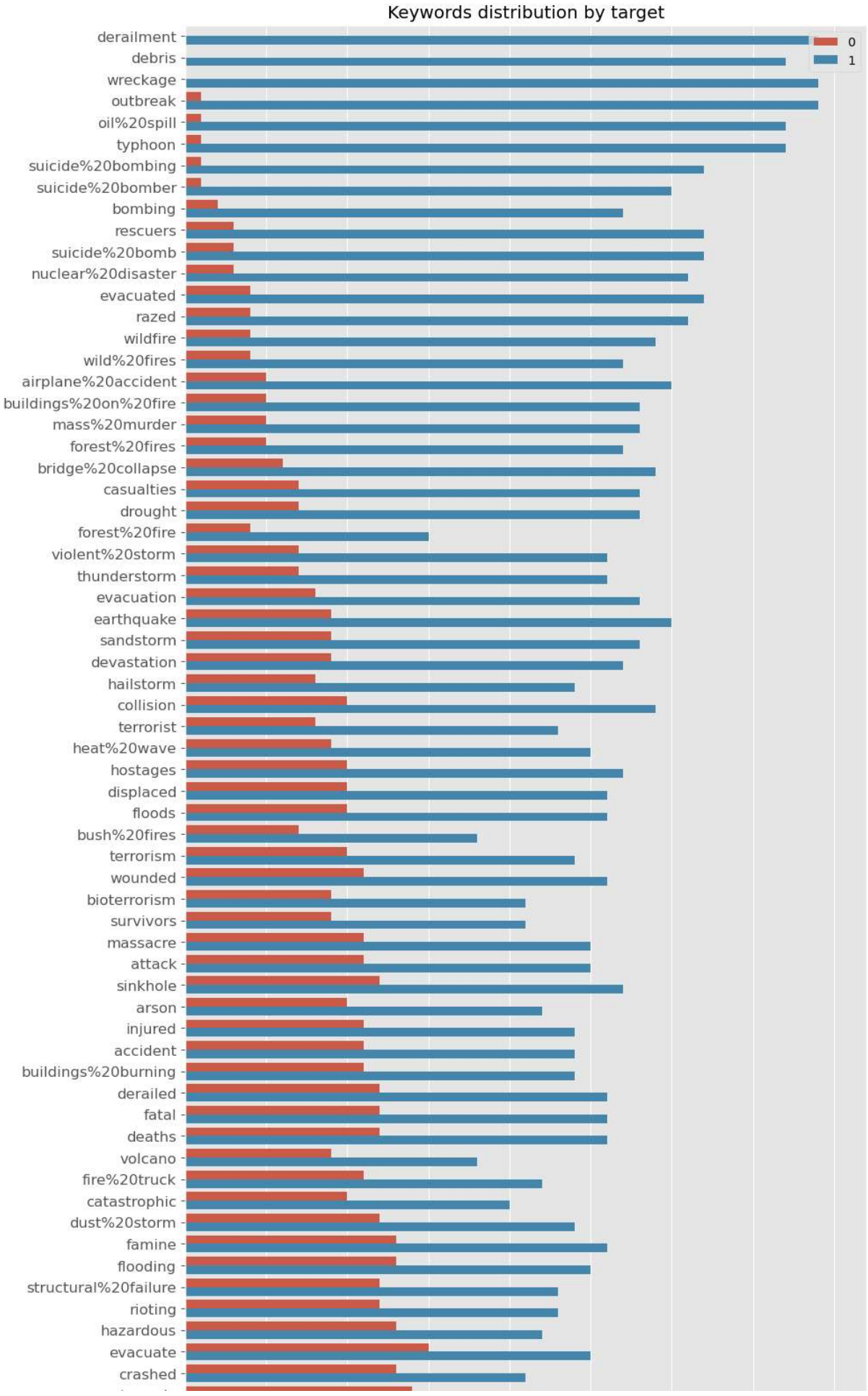
```
train_fake = train[train['target'] == 1]
keyword_cnt_fake = train_fake.keyword.value_counts()
keyword_cnt_fake
```

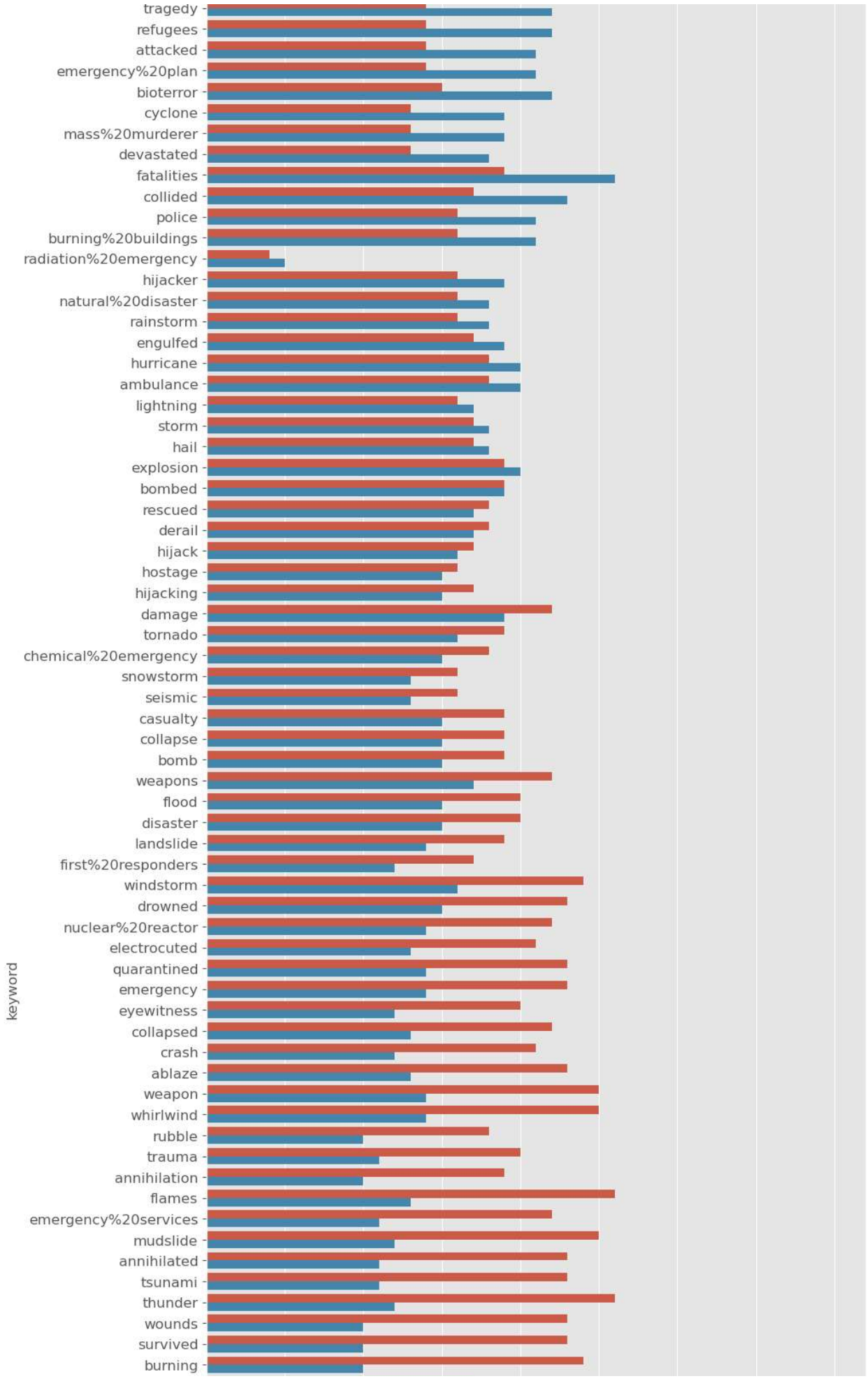
Out[6]:

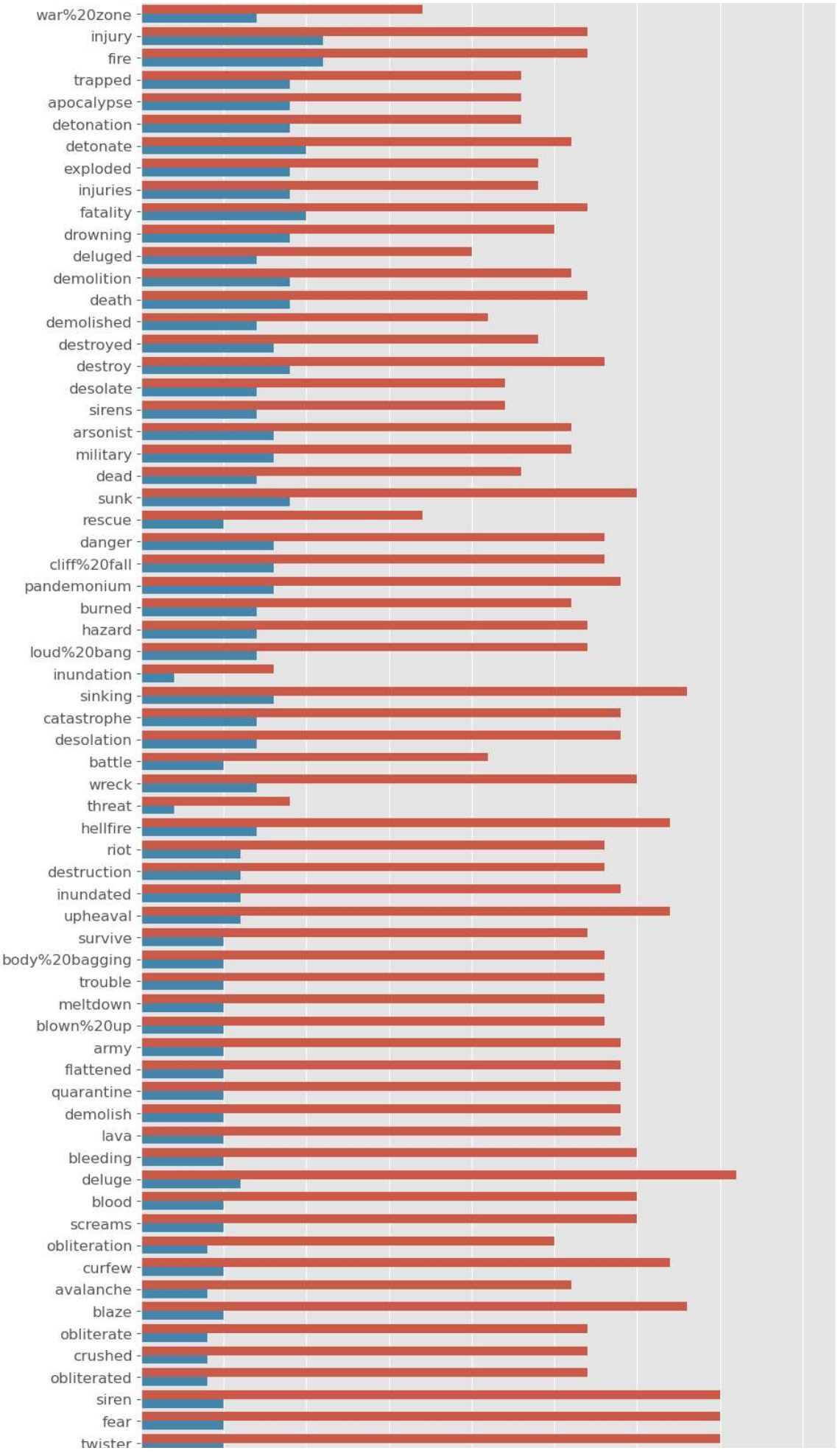
```
wreckage       39
derailment     39
outbreak       39
oil%20spill    37
typhoon        37
               ..
body%20bag      1
body%20bags     1
blazing         1
ruin            1
epicentre       1
Name: keyword, Length: 220, dtype: int64
```

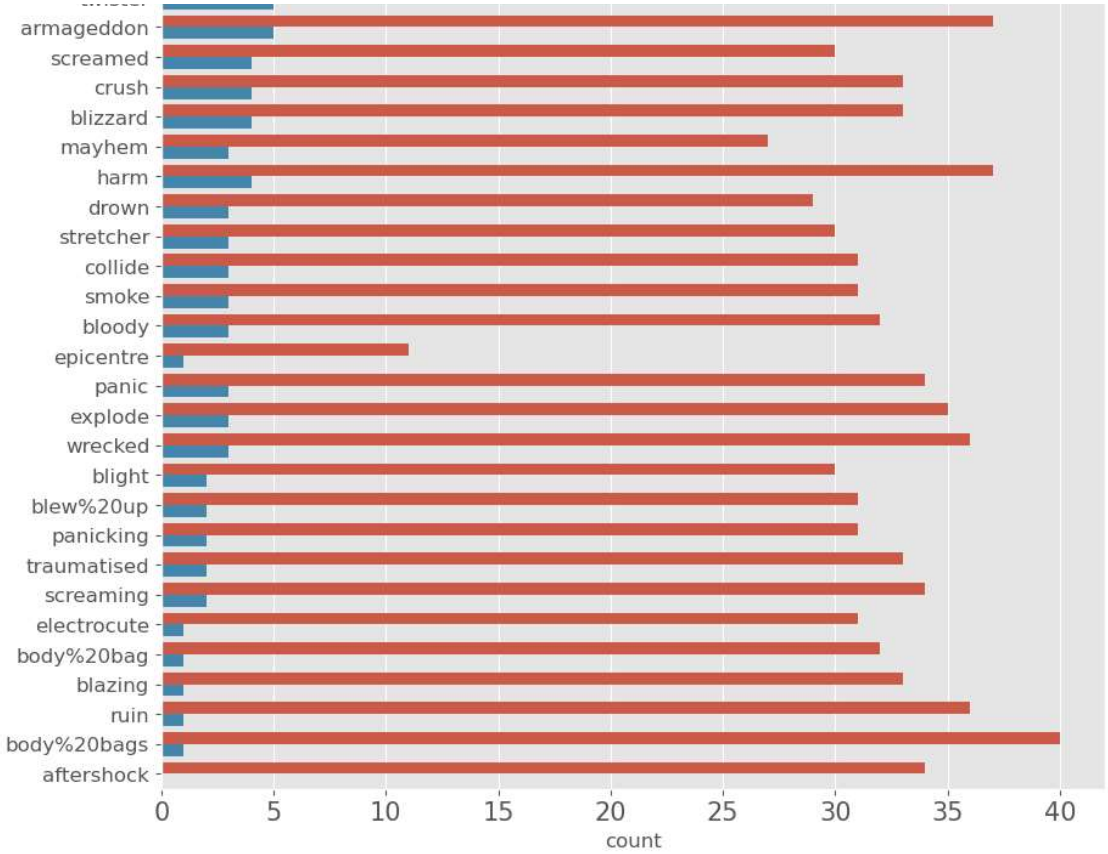
In [7]:

```
train['target_mean']=train.groupby('keyword')['target'].transform('mean')
fig=plt.figure(figsize=(10,72),dpi=100)
sns.countplot(y=train.sort_values(by='target_mean',ascending=False)['keyw
ord'],
              hue=train.sort_values(by='target_mean',ascending=False)['ta
rget'])
plt.tick_params(axis='x',labelsize=15)
plt.tick_params(axis='y',labelsize=12)
plt.legend(loc=1)
plt.title('Keywords distribution by target')
plt.show()
train.drop(columns=['target_mean'],inplace=True)
```


Keywords distribution by target
0
1
derailment
debris
wreckage
outbreak
oil%20spill
typhoon
suicide%20bombing
suicide%20bomber
bombing
rescuers
suicide%20bomb
nuclear%20disaster
evacuated
razed
wildfire
wild%20fires
airplane%20accident
buildings%20on%20fire
mass%20murder
forest%20fires
bridge%20collapse
casualties
drought
forest%20fire
violent%20storm
thunderstorm
evacuation
earthquake
sandstorm
devastation
hailstorm
collision
terrorist
heat%20wave
hostages
displaced
floods
bush%20fires
terrorism
wounded
bioterrorism
survivors
massacre
attack
sinkhole
arson
injured
accident
buildings%20burning
derailed
fatal
deaths
volcano
fire%20truck
catastrophic
dust%20storm
famine
flooding
structural%20failure
rioting
hazardous
evacuate
crashed


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

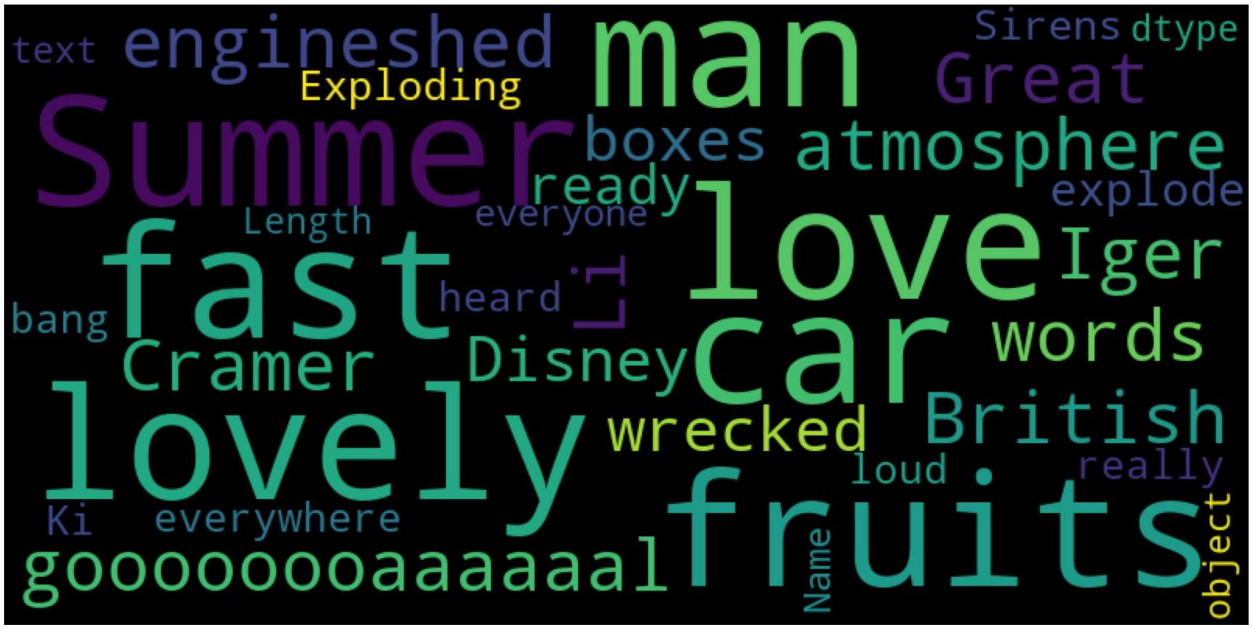

war%20zone
injury
fire
trapped
apocalypse
detonation
detonate
exploded
injuries
fatality
drowning
deluged
demolition
death
demolished
destroyed
destroy
desolate
sirens
arsonist
military
dead
sunk
rescue
danger
cliff%20fall
pandemonium
burned
hazard
loud%20bang
inundation
sinking
catastrophe
desolation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


armageddon
screamed
crush
blizzard
mayhem
harm
drown
stretcher
collide
smoke
bloody
epicentre
panic
explode
wrecked
blight
blew%20up
panicking
traumatised
screaming
electrocute
body%20bag
blazing
ruin
body%20bags
aftershock
0
5
10
15
20
25
30
35
40
count

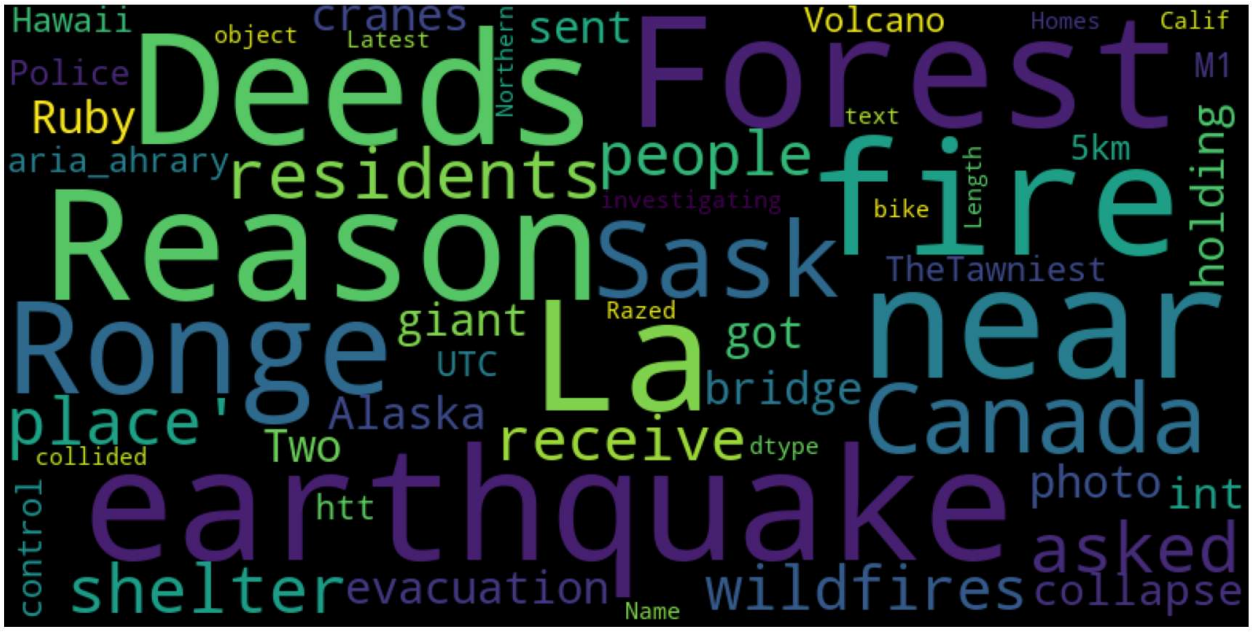
In [8]:

```
from wordcloud import WordCloud, STOPWORDS
def plot_wordcloud(text, mask=None, max_words=200, max_font_size=100, fig
ure_size=(24.0,16.0),
                   title = None, title_size=40, image_color=False):
    stopwords = set(STOPWORDS)
    more_stopwords = {'one', 'br', 'Po', 'th', 'sayi', 'fo', 'Unknown'}
    stopwords = stopwords.union(more_stopwords)

    wordcloud = WordCloud(background_color='black',
                    stopwords = stopwords,
                    max_words = max_words,
                    max_font_size = max_font_size,
                    random_state = 42,
                    width=800,
                    height=400,
                    mask = mask)
    wordcloud.generate(str(text))

    plt.figure(figsize=figure_size)
    if image_color:
        image_colors = ImageColorGenerator(mask);
        plt.imshow(wordcloud.recolor(color_func=image_colors), interpolat
ion="bilinear");
        plt.title(title, fontdict={'size': title_size,
                                   'verticalalignment': 'bottom'})
    else:
        plt.imshow(wordcloud);
        plt.title(title, fontdict={'size': title_size, 'color': 'black',
                                   'verticalalignment': 'bottom'})
    plt.axis('off');
    plt.tight_layout()
train_df0 = train[train['target']==0]
train_df1 = train[train['target']==1]
plot_wordcloud(train_df0['text'], title="Real Tweet Word Cloud")
plot_wordcloud(train_df1['text'], title="Fake Tweet Word Cloud")
```

Real Tweet Word Cloud



Fake Tweet Word Cloud



In [9]:
```
from nltk.stem.porter import PorterStemmer
n_corpus=[]
for text in tqdm(train['text']):
    text = re.sub(r'https?://\S+|www\.\S+', '', text)
    text = re.sub(r'<.*?>', '', text)
    text = re.sub(r'[^a-zA-Z0-9]+', ' ', text)
    text = re.sub(r'[0-9]', '', text)
    text = text.lower()
    text = nltk.word_tokenize(text)
    ps = PorterStemmer()
    text = [ps.stem(word) for word in text if not word in set(stopwords.w
ords('english'))]
    text = ' '.join(text)
    n_corpus.append(text)
```

```
100%|██████████| 7613/7613 [00:24<00:00, 314.26it/s]
```

In [10]:
```
train['text_n']=n_corpus
train.drop('text',axis=1)
```

Out[10]:

	id	keyword	location	target	text_n
0	1	NaN	NaN	1	deed reason earthquak may allah forgiv us
1	4	NaN	NaN	1	forest fire near la rong sask canada
2	5	NaN	NaN	1	resid ask shelter place notifi offic evacu she...
3	6	NaN	NaN	1	peopl receiv wildfir evacu order california
4	7	NaN	NaN	1	got sent photo rubi alaska smoke wildfir pour ...
...	...	...	...	...	...
7608	10869	NaN	NaN	1	two giant crane hold bridg collaps nearbi home
7609	10870	NaN	NaN	1	aria ahrari thetawniest control wild fire cali...
7610	10871	NaN	NaN	1	utc km volcano hawaii
7611	10872	NaN	NaN	1	polic investig e bike collid car littl portug ...
7612	10873	NaN	NaN	1	latest home raze northern california wildfir a...

7613 rows × 5 columns

In [11]:
```
n_corpus=[]
for text in tqdm(test['text']):
    text = re.sub(r'https?://\S+|www\.\S+', '', text)
    text = re.sub(r'<.*?>', '', text)
    text = re.sub(r'[^a-zA-Z0-9]+', ' ', text)
    text = re.sub(r'[0-9]', '', text)
    text = text.lower()
    text = nltk.word_tokenize(text)
    ps = PorterStemmer()
    text = [ps.stem(word) for word in text if not word in set(stopwords.w
ords('english'))]
    text = ' '.join(text)
    n_corpus.append(text)
```

```
100%|██████████| 3263/3263 [00:10<00:00, 315.52it/s]
```

In [12]:
```
test['text_n']=n_corpus
test.drop('text',axis=1)
```

Out[12]:

	id	keyword	location	text_n
0	0	NaN	NaN	happen terribl car crash
1	2	NaN	NaN	heard earthquak differ citi stay safe everyon
2	3	NaN	NaN	forest fire spot pond gees flee across street ...
3	9	NaN	NaN	apocalyps light spokan wildfir
4	11	NaN	NaN	typhoon soudelor kill china taiwan
...	...	...	...	...
3258	10861	NaN	NaN	earthquak safeti lo angel safeti fasten xrwn
3259	10865	NaN	NaN	storm ri wors last hurrican citi amp other har...
3260	10868	NaN	NaN	green line derail chicago
3261	10874	NaN	NaN	meg issu hazard weather outlook hwo
3262	10875	NaN	NaN	cityofcalgari activ municip emerg plan yycstorm

3263 rows × 4 columns

**Creating BERT model

In [13]:
```
!wget --quiet https://raw.githubusercontent.com/tensorflow/models/master/official/nlp/bert/tokenization.py
```

In [14]:
```
import tensorflow as tf
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
import tensorflow_hub as hub
import tokenization
```

In [15]:

```
def bert_encode(texts, tokenizer, max_len=512):
    all_tokens = []
    all_masks = []
    all_segments = []

    for text in texts:
        text = tokenizer.tokenize(text)

        text = text[:max_len-2]
        input_sequence = ["[CLS]"] + text + ["[SEP]"]
        pad_len = max_len - len(input_sequence)

        tokens = tokenizer.convert_tokens_to_ids(input_sequence)
        tokens += [0] * pad_len
        pad_masks = [1] * len(input_sequence) + [0] * pad_len
        segment_ids = [0] * max_len

        all_tokens.append(tokens)
        all_masks.append(pad_masks)
        all_segments.append(segment_ids)

    return np.array(all_tokens), np.array(all_masks), np.array(all_segmen
ts)
```

In [16]:
```
def build_model(bert_layer, max_len=512):
    input_word_ids = Input(shape=(max_len,), dtype=tf.int32, name="input_word_ids")
    input_mask = Input(shape=(max_len,), dtype=tf.int32, name="input_mask")
    segment_ids = Input(shape=(max_len,), dtype=tf.int32, name="segment_ids")

    _, sequence_output = bert_layer([input_word_ids, input_mask, segment_ids])
    clf_output = sequence_output[:, 0, :]
    out = Dense(1, activation='sigmoid')(clf_output)

    model = Model(inputs=[input_word_ids, input_mask, segment_ids], outputs=out)
    model.compile(Adam(lr=2e-6), loss='binary_crossentropy', metrics=['accuracy'])

    return model
```

In [17]:
```
#downloading bert layer
module_url = "https://tfhub.dev/tensorflow/bert_en_uncased_L-24_H-1024_A-16/1"
bert_layer = hub.KerasLayer(module_url, trainable=True)
```

In [18]:
```
vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()
tokenizer = tokenization.FullTokenizer(vocab_file, do_lower_case)
```

In [19]:
```
train_input = bert_encode(train.text.values, tokenizer, max_len=160)
test_input = bert_encode(test.text.values, tokenizer, max_len=160)
train_labels = train.target.values
```

In [20]:

```
model = build_model(bert_layer, max_len=160)
model.summary()
```

```
Model: "model"
__________________________________________________________________________
____________________________
Layer (type)                    Output Shape         Param #     Conne
cted to
==========================================================================
============================
input_word_ids (InputLayer)     [(None, 160)]        0
__________________________________________________________________________
____________________________
input_mask (InputLayer)         [(None, 160)]        0
__________________________________________________________________________
____________________________
segment_ids (InputLayer)        [(None, 160)]        0
__________________________________________________________________________
____________________________
keras_layer (KerasLayer)        [(None, 1024), (None 335141889   input
_word_ids[0][0]
                                                                 input
_mask[0][0]
                                                                 segme
nt_ids[0][0]
__________________________________________________________________________
____________________________
tf_op_layer_strided_slice (Tens [(None, 1024)]       0           keras
_layer[0][1]
__________________________________________________________________________
____________________________
dense (Dense)                   (None, 1)            1025        tf_op
_layer_strided_slice[0][0]
==========================================================================
============================
Total params: 335,142,914
Trainable params: 335,142,913
Non-trainable params: 1
__________________________________________________________________________
____________________________
```

In [21]:

```
train_history = model.fit(
    train_input, train_labels,
    validation_split=0.2,
    epochs=3,
    batch_size=16
)

model.save('model.h5')
```

```
Train on 6090 samples, validate on 1523 samples
Epoch 1/3
6090/6090 [==============================] - 419s 69ms/sample - loss:
0.4737 - accuracy: 0.7847 - val_loss: 0.3905 - val_accuracy: 0.8247
Epoch 2/3
6090/6090 [==============================] - 374s 61ms/sample - loss:
0.3458 - accuracy: 0.8593 - val_loss: 0.3880 - val_accuracy: 0.8332
Epoch 3/3
6090/6090 [==============================] - 374s 61ms/sample - loss:
0.2681 - accuracy: 0.8929 - val_loss: 0.4341 - val_accuracy: 0.8339
```

In [22]:

```
test_pred = model.predict(test_input)
submission=pd.read_csv('/kaggle/input/nlp-getting-started/sample_submissi
on.csv')
submission['target'] = test_pred.round().astype(int)
submission.to_csv('submission.csv', index=False)
```

In [ ]: