Automated Code

In [112]:
```
import os
import numpy as np
import pandas as pd
path_to_json = r'C:\Users\shakshi\Desktop\phemernrdataset\pheme-rnr-dataset\charliehebdo'
folders = [pos_json for pos_json in os.listdir(path_to_json) ]
print(folders)
```

```
['non-rumours', 'rumours']
```

In [113]:
```
#folders[0]
import json
import csv
```

In [114]:
```
base = r'C:\Users\shakshi\Desktop\phemernrdataset\pheme-rnr-dataset\charliehebdo'
```

In [115]:
```
# to display full text
pd.set_option('display.max_colwidth', -1)
```

Extract features from json and write to csv file on disk

In [116]:

```
for folder in folders:
    data = []
    if folder == 'non-rumours':
            within_tweetid = os.path.join(base, folder)
            #print(within_tweetid)
            within_folders = [pos for pos in os.listdir(within_tweetid)]
            #print(within_folders)
            for file in within_folders:
                within_tweetfolder = os.path.join(base, folder, file)
                #print (within_tweetfolder)
                src_tweet = [pos for pos in os.listdir(within_tweetfolder)]
                #print(each_tweet)
                if src_tweet[1] == 'source-tweet':
                    source = os.path.join(base, folder, file, within_tweetfolder, src_t
                    #print(source)
                    tweets = [pos for pos in os.listdir(source)]
                    #print(tweets)
                    for tweet in tweets:
                        ids = os.path.join(base, folder, file, within_tweetfolder, src_
                        if ids.endswith('.json'):
                            data = []
                            #print(ids)
                            with open(ids,'r') as f:
                                src_json = json.load(f)
                            src_df = pd.DataFrame(src_json)
                            src_df = src_df['text']
                            src_df = src_df.reset_index()
                            src_df.drop(['index'], axis = 1, inplace = True)
                            src_df = src_df.iloc[0,:]
                            src_df = src_df.reset_index()
                            src_df.drop("index", axis = 1, inplace = True)
                            #print(src_df.iloc[0,0])

                                #annot = annotations_df_slice.iloc[0,0]
                                #print(annot)
                            data.append(src_df.iloc[0,0])
                            data.append(0)
                            print("**************************************************
                            with open(r'C:\Users\shakshi\Desktop\phemernrdataset\pheme-
                                print(data)
                                writer = csv.writer(csvFile)
                                writer.writerow(data)
                            csvFile.close()



    elif folder == 'rumours':
        within_tweetid_r = os.path.join(base, folder)
            #print(within_tweetid)
        within_folders_r = [pos for pos in os.listdir(within_tweetid_r)]
            #print(within_folders)
        for file_r in within_folders_r:
            within_tweetfolder_r = os.path.join(base, folder, file_r)
                #print (within_tweetfolder)
            src_tweet_r = [pos for pos in os.listdir(within_tweetfolder_r)]
                #print(each_tweet)
            if src_tweet_r[1] == 'source-tweet':
                source_r = os.path.join(base, folder, file_r, within_tweetfolder_r, src_
                    #print(source)
                tweets_r = [pos for pos in os.listdir(source_r)]
                    #print(tweets)
                for tweet_r in tweets_r:
                    ids_r = os.path.join(base, folder, file_r, within_tweetfolder_r, sr
                    if ids_r.endswith('.json'):
                        data = []
```

```
                        #print(ids)
                    with open(ids_r,'r') as f:
                        src_json_r = json.load(f)
                    src_df_r = pd.DataFrame(src_json_r)
                    src_df_r = src_df_r['text']
                    src_df_r = src_df_r.reset_index()
                    src_df_r.drop(['index'], axis = 1, inplace = True)
                    src_df_r = src_df_r.iloc[0,:]
                    src_df_r = src_df_r.reset_index()
                    src_df_r.drop("index", axis = 1, inplace = True)
                            #print(src_df.iloc[0,0])

                            #annot = annotations_df_slice.iloc[0,0]
                                #print(annot)
                    data.append(src_df_r.iloc[0,0])
                    data.append(1)
                    print("************************************************************
                    with open(r'C:\Users\shakshi\Desktop\phemernrdataset\pheme-rnr-
                        print(data)
                        writer = csv.writer(csvFile)
                        writer.writerow(data)
                    csvFile.close()
    del src_df
    del src_df_r
    import gc
    gc.collect()

print(within_tweetid)
```

```
**********************************************************************
['Charlie Hebdo became well known for publishing the Muhammed cartoons two years ag
o', 0]
**********************************************************************
['Charlie Hebdo’s Last Tweet Before Shootings http://t.co/9Oa2xAqOcM (http://t.co/9
Oa2xAqOcM) http://t.co/skJHNEQcn0', (http://t.co/skJHNEQcn0',) 0]
**********************************************************************
["Prediction: the #CharlieHebdo massacre will not dent the political class's compla
cency one iota", 0]
**********************************************************************
['10:28am Charlie Hebdo account mocks ISIS leader, wishing him "good health and bes
t wishes" http://t.co/9Oa2xAqOcM (http://t.co/9Oa2xAqOcM) http://t.co/uYXayKLA7q',
 (http://t.co/uYXayKLA7q',) 0]
**********************************************************************
["If your faith isn't strong enough to cope with satirical poke, it oughtn't be str
ong enough to induce you to kill. Barbaric #CharlieHebdo", 0]
**********************************************************************
['Just arrived at scene of massacre #Paris #charliehebdo http://t.co/bxDwfuqRz8',
 (http://t.co/bxDwfuqRz8',) 0]
**********************************************************************
```

Load csv file where data is written

In [117]:

```
df = pd.read_csv(r'C:\Users\shakshi\Desktop\phemernrdataset\pheme-rnr-dataset\dump_char
```

In [118]:
```
df.head()
```

Out[118]:

	0	1
0	Charlie Hebdo became well known for publishing the Muhammed cartoons two years ago	0
1	Charlie Hebdo’s Last Tweet Before Shootings http://t.co/9Oa2xAqOcM http://t.co/skJHNEQcn0	0
2	Prediction: the #CharlieHebdo massacre will not dent the political class's complacency one iota	0
3	10:28am Charlie Hebdo account mocks ISIS leader, wishing him "good health and best wishes" http://t.co/9Oa2xAqOcM http://t.co/uYXayKLA7q	0
4	If your faith isn't strong enough to cope with satirical poke, it oughtn't be strong enough to induce you to kill. Barbaric #CharlieHebdo	0

In [119]:
```
df.columns = ['text','rumour']
```

In [120]:
```
df.head()
```

Out[120]:

	text	rumour
0	Charlie Hebdo became well known for publishing the Muhammed cartoons two years ago	0
1	Charlie Hebdo’s Last Tweet Before Shootings http://t.co/9Oa2xAqOcM http://t.co/skJHNEQcn0	0
2	Prediction: the #CharlieHebdo massacre will not dent the political class's complacency one iota	0
3	10:28am Charlie Hebdo account mocks ISIS leader, wishing him "good health and best wishes" http://t.co/9Oa2xAqOcM http://t.co/uYXayKLA7q	0
4	If your faith isn't strong enough to cope with satirical poke, it oughtn't be strong enough to induce you to kill. Barbaric #CharlieHebdo	0

In [123]:
```
#rumour_df = df.sample(frac = 1)
rumour_df = df
```

In [124]:
```
rumour_df.head()
```

Out[124]:

	text	rumour
0	Charlie Hebdo became well known for publishing the Muhammed cartoons two years ago	0
1	Charlie Hebdo’s Last Tweet Before Shootings http://t.co/9Oa2xAqOcM http://t.co/skJHNEQcn0	0
2	Prediction: the #CharlieHebdo massacre will not dent the political class's complacency one iota	0
3	10:28am Charlie Hebdo account mocks ISIS leader, wishing him "good health and best wishes" http://t.co/9Oa2xAqOcM http://t.co/uYXayKLA7q	0
4	If your faith isn't strong enough to cope with satirical poke, it oughtn't be strong enough to induce you to kill. Barbaric #CharlieHebdo	0

In [125]:
```
rumour_df.shape
```

Out[125]:
```
(2079, 2)
```

NLP Stuff

In [126]:
```
import re
# function for cleaning data
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for i in r:
        input_txt = re.sub(i, '', input_txt)
    return input_txt
```

In [127]:
```
rumour_df['clean'] = np.vectorize(remove_pattern)(rumour_df['text'], "@[\w]*")
```

In [128]:
```
rumour_df.head()
```

Out[128]:

	text	rumour	clean
0	Charlie Hebdo became well known for publishing the Muhammed cartoons two years ago	0	Charlie Hebdo became well known for publishing the Muhammed cartoons two years ago
1	Charlie Hebdo’s Last Tweet Before Shootings http://t.co/9Oa2xAqOcM http://t.co/skJHNEQcn0	0	Charlie Hebdo’s Last Tweet Before Shootings http://t.co/9Oa2xAqOcM http://t.co/skJHNEQcn0
2	Prediction: the #CharlieHebdo massacre will not dent the political class's complacency one iota	0	Prediction: the #CharlieHebdo massacre will not dent the political class's complacency one iota
3	10:28am Charlie Hebdo account mocks ISIS leader, wishing him "good health and best wishes" http://t.co/9Oa2xAqOcM http://t.co/uYXayKLA7q	0	10:28am Charlie Hebdo account mocks ISIS leader, wishing him "good health and best wishes" http://t.co/9Oa2xAqOcM http://t.co/uYXayKLA7q
4	If your faith isn't strong enough to cope with satirical poke, it oughtn't be strong enough to induce you to kill. Barbaric #CharlieHebdo	0	If your faith isn't strong enough to cope with satirical poke, it oughtn't be strong enough to induce you to kill. Barbaric #CharlieHebdo

In [129]:
```
rumour_df['clean'] = rumour_df['clean'].str.replace("[^a-zA-Z#]", " ")
```

In [130]:
```
rumour_df.head()
```

Out[130]:

	text	rumour	clean
0	Charlie Hebdo became well known for publishing the Muhammed cartoons two years ago	0	Charlie Hebdo became well known for publishing the Muhammed cartoons two years ago
1	Charlie Hebdo’s Last Tweet Before Shootings http://t.co/9Oa2xAqOcM http://t.co/skJHNEQcn0	0	Charlie Hebdo s Last Tweet Before Shootings http t co Oa xAqOcM http t co skJHNEQcn
2	Prediction: the #CharlieHebdo massacre will not dent the political class's complacency one iota	0	Prediction the #CharlieHebdo massacre will not dent the political class s complacency one iota
3	10:28am Charlie Hebdo account mocks ISIS leader, wishing him "good health and best wishes" http://t.co/9Oa2xAqOcM http://t.co/uYXayKLA7q	0	am Charlie Hebdo account mocks ISIS leader wishing him good health and best wishes http t co Oa xAqOcM http t co uYXayKLA q
4	If your faith isn't strong enough to cope with satirical poke, it oughtn't be strong enough to induce you to kill. Barbaric #CharlieHebdo	0	If your faith isn t strong enough to cope with satirical poke it oughtn t be strong enough to induce you to kill Barbaric #CharlieHebdo

In [131]:
```
rumour_df['clean'] = rumour_df.clean.apply(lambda x: ' '.join([w for w in x.split() if
```

In [132]:
```
rumour_df.head()
```

Out[132]:

	text	rumour	clean
0	Charlie Hebdo became well known for publishing the Muhammed cartoons two years ago	0	Charlie Hebdo became well known publishing Muhammed cartoons years
1	Charlie Hebdo's Last Tweet Before Shootings http://t.co/9Oa2xAqOcM http://t.co/skJHNEQcn0	0	Charlie Hebdo Last Tweet Before Shootings http xAqOcM http skJHNEQcn
2	Prediction: the #CharlieHebdo massacre will not dent the political class's complacency one iota	0	Prediction #CharlieHebdo massacre will dent political class complacency iota
3	10:28am Charlie Hebdo account mocks ISIS leader, wishing him "good health and best wishes" http://t.co/9Oa2xAqOcM http://t.co/uYXayKLA7q	0	Charlie Hebdo account mocks ISIS leader wishing good health best wishes http xAqOcM http uYXayKLA
4	If your faith isn't strong enough to cope with satirical poke, it oughtn't be strong enough to induce you to kill. Barbaric #CharlieHebdo	0	your faith strong enough cope with satirical poke oughtn strong enough induce kill Barbaric #CharlieHebdo

In [133]:
```
from tensorflow.python.keras.preprocessing.text import Tokenizer
```

In [134]:
```
from tensorflow.python.keras.preprocessing.sequence import pad_sequences
```

In [135]:
```
token = Tokenizer()
```

In [136]:
```
#tweets = X_train + X_test
```

In [137]:
```
#token.fit_on_texts(tweets)
```

In [138]:
```
max_len = max([len(tweet.split()) for tweet in rumour_df])
```

In [139]:
```
max_len
```

Out[139]:
```
1
```

In [140]:
```
rumour_df.clean = rumour_df.clean.apply(lambda x: x.split())
```

In [141]:
```
rumour_df.head()
```

Out[141]:

	text	rumour	clean
0	Charlie Hebdo became well known for publishing the Muhammed cartoons two years ago	0	[Charlie, Hebdo, became, well, known, publishing, Muhammed, cartoons, years]
1	Charlie Hebdo's Last Tweet Before Shootings http://t.co/9Oa2xAqOcM http://t.co/skJHNEQcn0	0	[Charlie, Hebdo, Last, Tweet, Before, Shootings, http, xAqOcM, http, skJHNEQcn]
2	Prediction: the #CharlieHebdo massacre will not dent the political class's complacency one iota	0	[Prediction, #CharlieHebdo, massacre, will, dent, political, class, complacency, iota]
3	10:28am Charlie Hebdo account mocks ISIS leader, wishing him "good health and best wishes" http://t.co/9Oa2xAqOcM http://t.co/uYXayKLA7q	0	[Charlie, Hebdo, account, mocks, ISIS, leader, wishing, good, health, best, wishes, http, xAqOcM, http, uYXayKLA]
4	If your faith isn't strong enough to cope with satirical poke, it oughtn't be strong enough to induce you to kill. Barbaric #CharlieHebdo	0	[your, faith, strong, enough, cope, with, satirical, poke, oughtn, strong, enough, induce, kill, Barbaric, #CharlieHebdo]

In [142]:
```
from nltk.stem.porter import *
stemmer = PorterStemmer()
tokenized_tweet = rumour_df.clean.apply(lambda x: [stemmer.stem(i) for i in x]) # stemm
```

In [143]:
```
tokenized_tweet.head()
```

Out[143]:
```
0    [charli, hebdo, becam, well, known, publish, muham, cartoon, year]
1    [charli, hebdo, last, tweet, befor, shoot, http, xaqocm, http, skjhneqcn]
2    [predict, #charliehebdo, massacr, will, dent, polit, class, complac, iota]
3    [charli, hebdo, account, mock, isi, leader, wish, good, health, best, wish, ht
tp, xaqocm, http, uyxaykla]
4    [your, faith, strong, enough, cope, with, satir, poke, oughtn, strong, enough,
induc, kill, barbar, #charliehebdo]
Name: clean, dtype: object
```

In [144]:
```
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = ' '.join(tokenized_tweet[i])
rumour_df['clean'] = tokenized_tweet
```

In [145]:
```
rumour_df.head()
```

Out[145]:

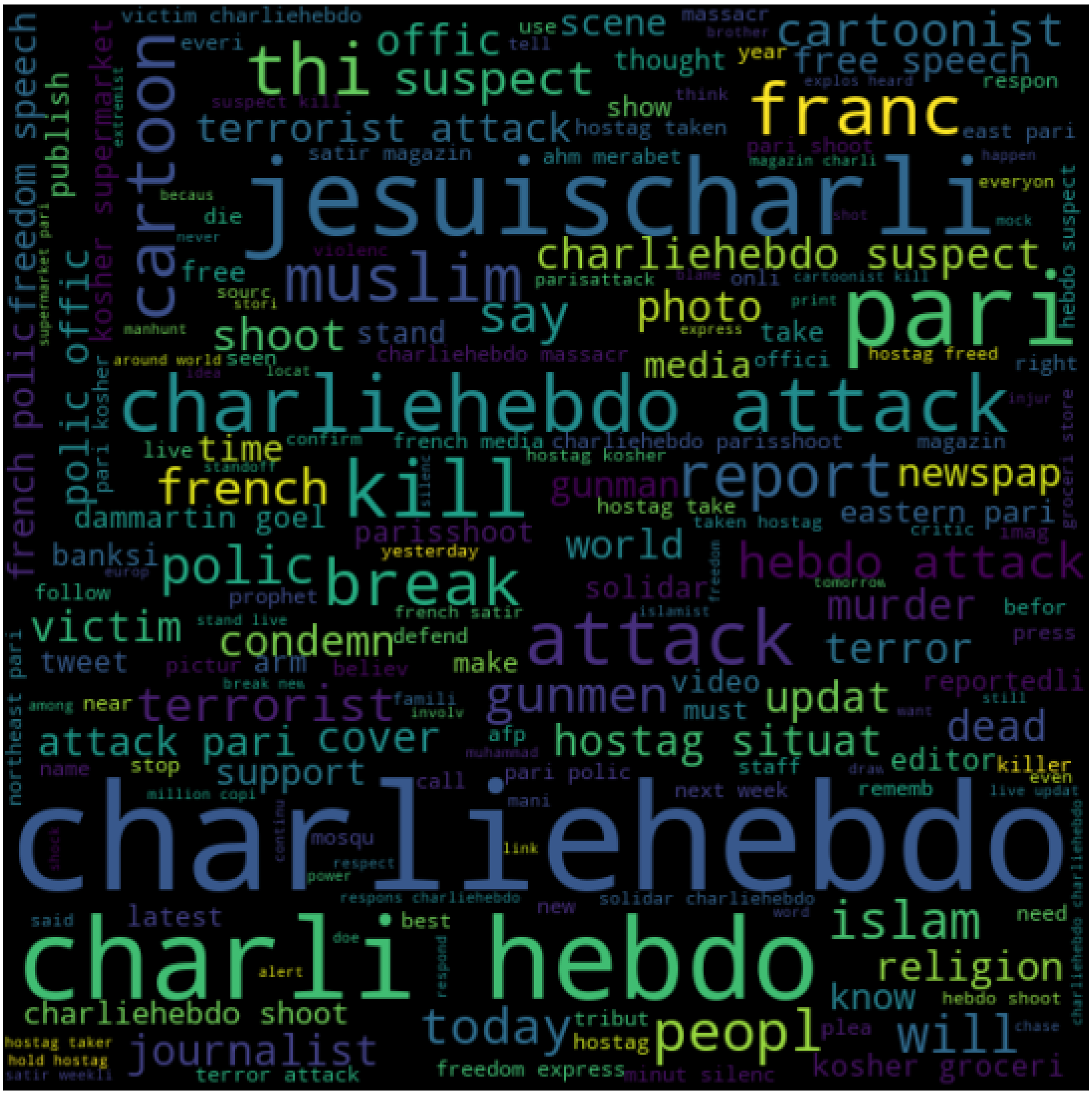
	text	rumour	clean
0	Charlie Hebdo became well known for publishing the Muhammed cartoons two years ago	0	charli hebdo becam well known publish muham cartoon year
1	Charlie Hebdo’s Last Tweet Before Shootings http://t.co/9Oa2xAqOcM http://t.co/skJHNEQcn0	0	charli hebdo last tweet befor shoot http xaqocm http skjhneqcn
2	Prediction: the #CharlieHebdo massacre will not dent the political class's complacency one iota	0	predict #charliehebdo massacr will dent polit class complac iota
3	10:28am Charlie Hebdo account mocks ISIS leader, wishing him "good health and best wishes" http://t.co/9Oa2xAqOcM http://t.co/uYXayKLA7q	0	charli hebdo account mock isi leader wish good health best wish http xaqocm http uyxaykla
4	If your faith isn't strong enough to cope with satirical poke, it oughtn't be strong enough to induce you to kill. Barbaric #CharlieHebdo	0	your faith strong enough cope with satir poke oughtn strong enough induc kill barbar #charliehebdo

Analysis

In [146]:

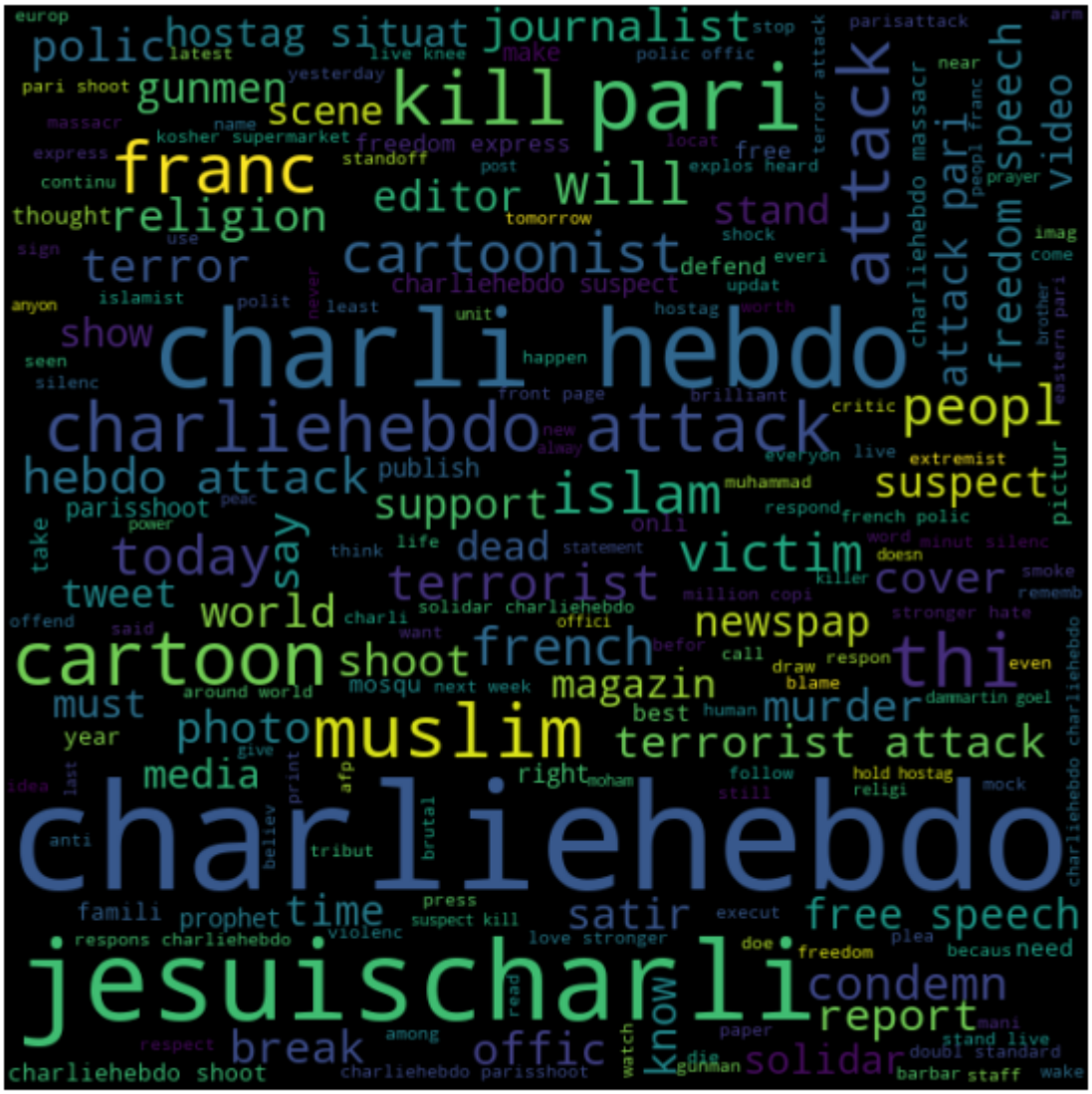
```
import matplotlib.pyplot as plt
words = ' '.join([text for text in rumour_df['clean']])

from wordcloud import WordCloud
wordcloud = WordCloud(width=600, height=600, random_state=129, max_font_size=120).gener
plt.figure(figsize=(10, 10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```



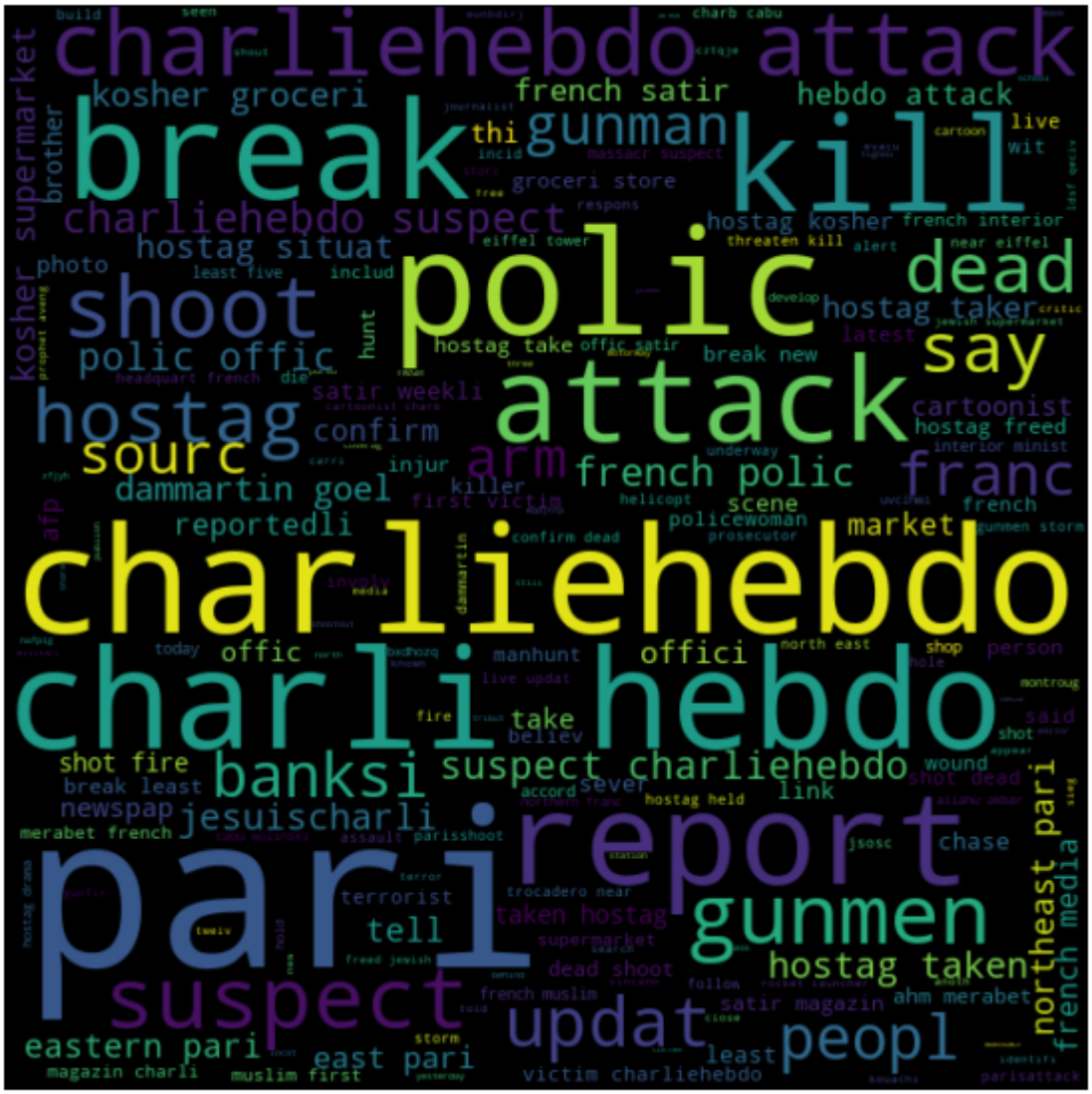
In [147]:
```
normal_words =' '.join([text for text in rumour_df['clean'][rumour_df['rumour'] == 0]])

wordcloud = WordCloud(width=600, height=600, random_state=129, max_font_size=110).gener
plt.figure(figsize=(10, 10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```



In [148]:
```
normal_words =' '.join([text for text in rumour_df['clean'][rumour_df['rumour'] == 1]])

wordcloud = WordCloud(width=600, height=600, random_state=129, max_font_size=120).gener
plt.figure(figsize=(10, 10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```



In [149]:
```
# function to collect hashtags

def collect_hashtag(x):
    hashtags = []    # Loop over the words in the tweet
    for i in x:
        ht = re.findall(r"#(\w+)", i)
        hashtags.append(ht)
    return hashtags
```

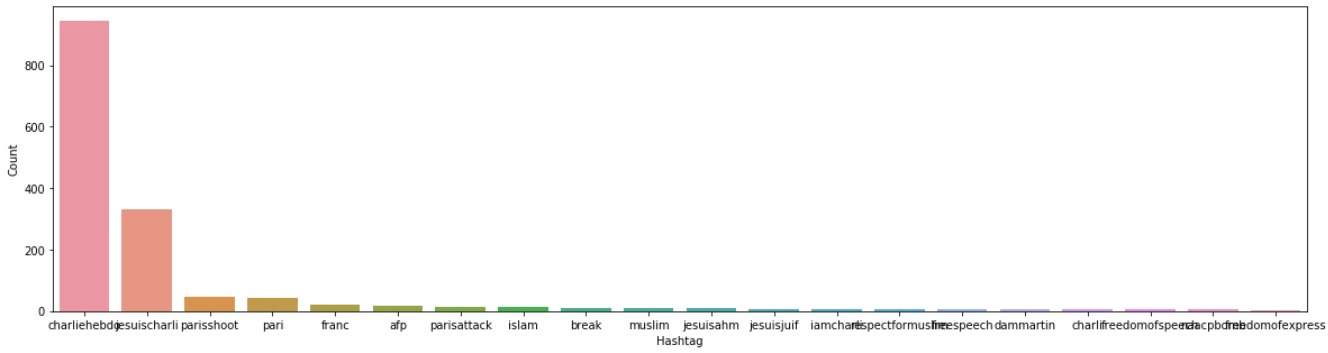
In [150]:
```
# extracting hashtags from non rumour

hashtags_nr = collect_hashtag(rumour_df['clean'][rumour_df['rumour'] == 0])
hashtags_r = collect_hashtag(rumour_df['clean'][rumour_df['rumour'] == 1])
```

In [151]:
```
hashtags_nr = sum(hashtags_nr,[])
hashtags_r = sum(hashtags_r,[])
```

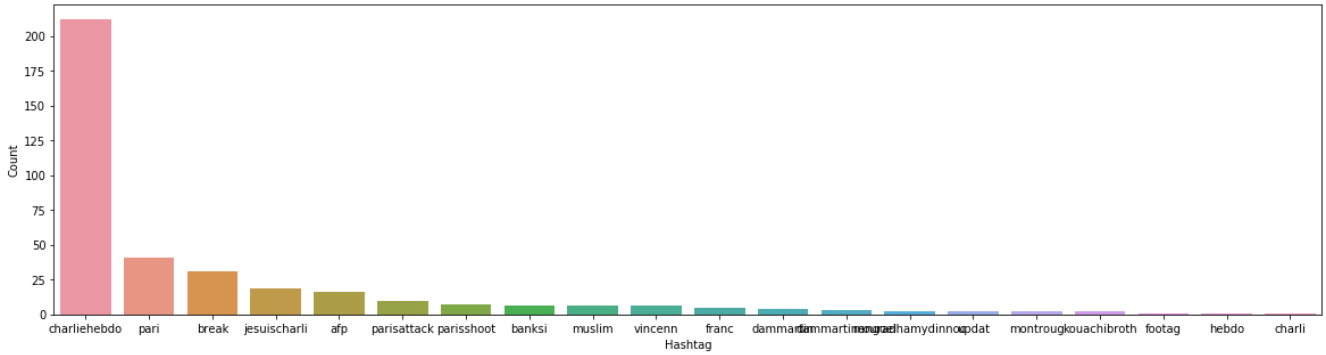
In [152]:
```
import nltk
freq = nltk.FreqDist(hashtags_nr)
df_nr = pd.DataFrame(
    {
    'Hashtag': list(freq.keys()),
    'Count': list(freq.values())
    }
)
```

In [153]:
```
# selecting top 20 most frequent hashtags for non rumour
import seaborn as sns
df_nr = df_nr.nlargest(columns="Count", n = 20)
plt.figure(figsize=(20,5))
ax = sns.barplot(data=df_nr, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
# plt.xticks(rotation=90)
plt.show()
```



In [154]:
```
import nltk
freq_r = nltk.FreqDist(hashtags_r)
df_r = pd.DataFrame(
    {
    'Hashtag': list(freq_r.keys()),
    'Count': list(freq_r.values())
    }
)

# selecting top 20 most frequent hashtags for non rumour
import seaborn as sns
df_r = df_r.nlargest(columns="Count", n = 20)
plt.figure(figsize=(20,5))
ax = sns.barplot(data=df_r, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
# plt.xticks(rotation=90)
plt.show()
```



Generating Doc2Vec Word Embeddings

In [155]:
```
from tqdm import tqdm
tqdm.pandas(desc="progress-bar")
from gensim.models.doc2vec import LabeledSentence
```

In [156]:
```
def tags(tweet_no):
    output = []
    for i, s in zip(tweet_no.index, tweet_no):
        output.append(LabeledSentence(s, ["tweet_" + str(i)]))
    return output

tagged_tweets = tags(tokenized_tweet) # label all the tweets
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: DeprecationWarnin
g: Call to deprecated `LabeledSentence` (Class will be removed in 4.0.0, use TaggedDoc
ument instead).
  after removing the cwd from sys.path.
```

In [157]:
```
tagged_tweets[:1]
```

Out[157]:
```
[LabeledSentence(words='charli hebdo becam well known publish muham cartoon year', tag
s=['tweet_0'])]
```

In [158]:
```
import gensim
```

In [189]:
```
model_doc2vec = gensim.models.Doc2Vec(dm=1, # dm = 1 for 'distributed memory' model
                                      dm_mean=1, # dm_mean = 1 for using mean of the contex
                                      vector_size=100, # no. of desired features
                                      window=5, # width of the context window
                                      negative=7, # if > 0 then negative sampling will be u
                                      min_count=5, # Ignores all words with total frequency
                                      workers=32, # no. of cores
                                      alpha=0.1, # learning rate
                                      seed = 23, # for reproducibility
                                     )

vocab = model_doc2vec.build_vocab([i for i in tqdm(tagged_tweets)])

model_doc2vec.train(tagged_tweets, total_examples= len(rumour_df['clean']), epochs=20)
```

```
100%|██████████████████████████████████████████████████████████████████████████| 2079/2
079 [00:00<00:00, 1042184.54it/s]
```

In [160]:
```
arrays = np.zeros((len(tokenized_tweet), 100))
for i in range(len(rumour_df)):
    arrays[i,:] = model_doc2vec.docvecs[i].reshape((1,100))

doc2vec_df = pd.DataFrame(arrays)
doc2vec_df.shape
```

Out[160]: (2079, 100)

In [161]:
```
doc2vec_df.head()
```

Out[161]:

	0	1	2	3	4	5	6	7	8	
0	0.163647	-0.081417	-0.258556	0.029476	-0.033844	-0.045694	-0.019156	0.155504	-0.025553	-0.11388
1	-0.045149	-0.104434	0.006182	0.041315	-0.120619	0.079116	0.070732	0.028829	0.066363	-0.24611
2	0.614088	-0.158158	-0.231412	0.369314	-0.080479	-0.215577	0.135134	-0.071799	-0.058652	-0.02694
3	0.031000	0.077355	0.011719	-0.112992	-0.073069	0.175858	-0.061835	0.095325	0.039988	-0.16216
4	0.210198	0.076235	-0.295209	-0.041230	0.023388	0.174380	-0.086518	0.136229	0.083955	-0.03795

5 rows × 100 columns

Attaching labels after word embeddings for each sentence

In [164]:
```
doc2vec_df['label'] = rumour_df['rumour']
```

In [165]:
```
doc2vec_df.head()
```

Out[165]:

	0	1	2	3	4	5	6	7	8	
0	0.163647	-0.081417	-0.258556	0.029476	-0.033844	-0.045694	-0.019156	0.155504	-0.025553	-0.1
1	-0.045149	-0.104434	0.006182	0.041315	-0.120619	0.079116	0.070732	0.028829	0.066363	-0.2
2	0.614088	-0.158158	-0.231412	0.369314	-0.080479	-0.215577	0.135134	-0.071799	-0.058652	-0.0
3	0.031000	0.077355	0.011719	-0.112992	-0.073069	0.175858	-0.061835	0.095325	0.039988	-0.1
4	0.210198	0.076235	-0.295209	-0.041230	0.023388	0.174380	-0.086518	0.136229	0.083955	-0.0

5 rows × 101 columns

Split Data into training and testing

In [167]:
```
doc2vec_df = doc2vec_df.sample(frac = 1)
```

In [168]:
```
doc2vec_df.head()
```

Out[168]:

	0	1	2	3	4	5	6	7	8	
1503	0.288692	0.012739	-0.035888	0.044083	-0.102667	0.087308	-0.191001	0.105903	0.158970	-0.091
465	0.316335	0.017365	-0.101848	0.009252	-0.130105	0.030111	0.108854	0.084407	0.041771	0.077
2041	0.446431	-0.209347	-0.258609	0.328840	0.024910	-0.155450	0.071426	-0.133219	-0.072286	-0.015
1676	0.329207	-0.084758	-0.171365	0.238556	-0.160819	-0.101741	-0.110255	-0.050976	0.043506	0.034
1296	0.219836	0.036532	-0.119205	0.008497	-0.076282	0.031978	-0.038002	0.190173	0.141635	-0.110

5 rows × 101 columns

In [171]:
```
X = doc2vec_df.iloc[:,:-1]
```

In [173]:
```
y = doc2vec_df.iloc[:,-1]
```

In [169]:
```
from sklearn.model_selection import train_test_split
```

In [175]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=
```

In [177]:
```
y_train.head()
```

Out[177]:
```
480     0
729     0
839     0
1492    0
1174    0
Name: label, dtype: int64
```

Classifier - Logistic Regression

In [178]:
```
from sklearn.linear_model import LogisticRegression
```

In [179]:
```
clf = LogisticRegression(random_state=0).fit(X_train, y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: Futur
eWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silen
ce this warning.
  FutureWarning)
```

In [181]:
```
y_pred = clf.predict(X_test)
```

Accuracy

In [182]:
```
clf.score(X_test, y_test)
```

Out[182]:
```
0.7692307692307693
```

Evaluating Results

In [183]:
```
from sklearn.metrics import precision_recall_fscore_support
recall_logistic = precision_recall_fscore_support(y_test, y_pred)

from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))

from sklearn.metrics import confusion_matrix
confusion_mat = pd.DataFrame(confusion_matrix(y_test, y_pred))

print('Confusion matrix \n',confusion_mat)
```

```
              precision    recall  f1-score   support

           0       0.77      0.99      0.87       320
           1       0.50      0.03      0.06        96

   micro avg       0.77      0.77      0.77       416
   macro avg       0.64      0.51      0.46       416
weighted avg       0.71      0.77      0.68       416

Confusion matrix
      0  1
0  317  3
1  93   3
```