AI_phase4

Sentiment Analysis for Twitter Data

Problem Statement

Study the subjects of recent tweets about the vaccine made in collaboration by Pfizer and BioNTech, perform various NLP tasks on this data source

About Data Set

Data is collected from recent tweets about Pfizer and BioNTech vaccine.

The data is collected using tweepy Python package to access Twitter API.

Importing Libraries

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../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))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
 ↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
 ↪outside of the current session
```

```
/kaggle/input/twitter-sentiment-analysis/__results__.html
/kaggle/input/twitter-sentiment-analysis/encoder.pkl
/kaggle/input/twitter-sentiment-analysis/model.h5
```

```
/kaggle/input/twitter-sentiment-analysis/__output__.json
/kaggle/input/twitter-sentiment-analysis/model.w2v
/kaggle/input/twitter-sentiment-analysis/tokenizer.pkl
/kaggle/input/twitter-sentiment-analysis/custom.css
/kaggle/input/twitter-sentiment-analysis/__results___files/__results___14_1.png
/kaggle/input/twitter-sentiment-analysis/__results___files/__results___59_0.png
/kaggle/input/twitter-sentiment-analysis/__results___files/__results___49_1.png
/kaggle/input/twitter-sentiment-analysis/__results___files/__results___49_0.png
/kaggle/input/twitter-sentiment-analysis-hatred-speech/train.csv
/kaggle/input/twitter-sentiment-analysis-hatred-speech/test.csv
/kaggle/input/twitter-sentiment-analysis-for-beginners/Sentiment-LR.pickle
/kaggle/input/twitter-sentiment-analysis-for-beginners/vectoriser-
ngram-(1,2).pickle
/kaggle/input/twitter-sentiment-analysis-for-beginners/__results__.html
/kaggle/input/twitter-sentiment-analysis-for-beginners/Sentiment-BNB.pickle
/kaggle/input/twitter-sentiment-analysis-for-beginners/__resultx__.html
/kaggle/input/twitter-sentiment-analysis-for-beginners/__notebook__.ipynb
/kaggle/input/twitter-sentiment-analysis-for-beginners/__output__.json
/kaggle/input/twitter-sentiment-analysis-for-beginners/custom.css
/kaggle/input/twitter-sentiment-analysis-for-
beginners/__results___files/__results___25_1.png
/kaggle/input/twitter-sentiment-analysis-for-
beginners/__results___files/__results___27_1.png
/kaggle/input/twitter-sentiment-analysis-for-
beginners/__results___files/__results___12_1.png
/kaggle/input/twitter-sentiment-analysis-for-
beginners/__results___files/__results___23_1.png
/kaggle/input/twitter-sentiment-analysis-for-
beginners/__results___files/__results___10_1.png
/kaggle/input/twitter-sentiment-analysis-for-
beginners/__results___files/__results___3_0.png
/kaggle/input/tweet-sentiment-extraction/sample_submission.csv
/kaggle/input/tweet-sentiment-extraction/train.csv
/kaggle/input/tweet-sentiment-extraction/test.csv
/kaggle/input/pfizer-vaccine-tweets/vaccination_tweets.csv
/kaggle/input/twitter-entity-sentiment-analysis/twitter_validation.csv
/kaggle/input/twitter-entity-sentiment-analysis/twitter_training.csv
/kaggle/input/twitter-sentiment-dataset/Twitter_Data.csv
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results__.html
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-model/submission.csv
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__resultx__.html
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__notebook__.ipynb
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-model/__output__.json
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-model/custom.css
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
```

```
model/models/model_neg/tokenizer
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_neg/meta.json
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_neg/vocab/vectors
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_neg/vocab/key2row
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_neg/vocab/lexemes.bin
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_neg/vocab/strings.json
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_neg/ner/model
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_neg/ner/moves
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_neg/ner/cfg
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_pos/tokenizer
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_pos/meta.json
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_pos/vocab/vectors
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_pos/vocab/key2row
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_pos/vocab/lexemes.bin
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_pos/vocab/strings.json
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_pos/ner/model
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_pos/ner/moves
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/models/model_pos/ner/cfg
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___20_1.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___41_1.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___93_0.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___39_1.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___37_1.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___86_0.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
```

```
model/__results___files/__results___36_0.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___34_0.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___84_0.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___82_0.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___94_0.png
/kaggle/input/twitter-sentiment-extaction-analysis-eda-and-
model/__results___files/__results___92_0.png
```

[2]:
```
#For basic table operation
import pandas as pd

#For work with arrays
import numpy as np

#For find pattern in text
import re

#For visualization
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
style.use("ggplot")

#For processing textial data
from textblob import TextBlob

#For Tokenizing segments
from nltk.tokenize import word_tokenize

#For Stemming text
from nltk.stem import PorterStemmer

#For removing StopWords
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

#For Plotting Words
from wordcloud import WordCloud

# Convert a collection of text documents to a matrix of token counts.
from sklearn.feature_extraction.text import CountVectorizer

#To split data into train and test
```

```
from sklearn.model_selection import train_test_split

#For fitting model
from sklearn.linear_model import LogisticRegression

#For evaluation of model
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix, ConfusionMatrixDisplay

#For Hyper-tuning model
from sklearn.model_selection import GridSearchCV
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A
NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy
(detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

[3]:
```
df = pd.read_csv("/kaggle/input/pfizer-vaccine-tweets/vaccination_tweets.csv")
df.head(4)
```

[3]:
```
                    id      user_name              user_location  \
0  1340539111971516416     Rachel Roh  La Crescenta-Montrose, CA
1  1338158543359250433    Albert Fong          San Francisco, CA
2  1337858199140118533            eli                   Your Bed
3  1337855739918835717  Charles Adler     Vancouver, BC - Canada

                                    user_description         user_created  \
0  Aggregator of Asian American news; scanning di…  2009-04-08 17:52:46
1  Marketing dude, tech geek, heavy metal & '80s …  2009-09-21 15:27:30
2                                      heil, hydra   2020-06-25 23:30:28
3  Hosting "CharlesAdlerTonight" Global News Radi…  2008-09-10 11:28:53

   user_followers  user_friends  user_favourites  user_verified  \
0             405          1692             3247          False
1             834           666              178          False
2              10            88              155          False
3           49165          3933            21853           True

                  date                                               text  \
0  2020-12-20 06:06:44  Same folks said daikon paste could treat a cyt…
1  2020-12-13 16:27:13  While the world has been on the wrong side of …
2  2020-12-12 20:33:45  #coronavirus #SputnikV #AstraZeneca #PfizerBio…
3  2020-12-12 20:23:59  Facts are immutable, Senator, even when you're…

                                hashtags               source  \
0                      ['PfizerBioNTech']  Twitter for Android
1                                    NaN      Twitter Web App
```

```
2  ['coronavirus', 'SputnikV', 'AstraZeneca', 'Pf…  Twitter for Android
3                                                NaN      Twitter Web App

   retweets  favorites  is_retweet
0         0          0       False
1         1          1       False
2         0          0       False
3       446       2129       False
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11020 entries, 0 to 11019
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                11020 non-null  int64
 1   user_name         11020 non-null  object
 2   user_location     8750 non-null   object
 3   user_description  10341 non-null  object
 4   user_created      11020 non-null  object
 5   user_followers    11020 non-null  int64
 6   user_friends      11020 non-null  int64
 7   user_favourites   11020 non-null  int64
 8   user_verified     11020 non-null  bool
 9   date              11020 non-null  object
 10  text              11020 non-null  object
 11  hashtags          8438 non-null   object
 12  source            11019 non-null  object
 13  retweets          11020 non-null  int64
 14  favorites         11020 non-null  int64
 15  is_retweet        11020 non-null  bool
dtypes: bool(2), int64(6), object(8)
memory usage: 1.2+ MB
```

[5]:
```
df.columns
```

[5]:
```
Index(['id', 'user_name', 'user_location', 'user_description', 'user_created',
       'user_followers', 'user_friends', 'user_favourites', 'user_verified',
       'date', 'text', 'hashtags', 'source', 'retweets', 'favorites',
       'is_retweet'],
      dtype='object')
```

[6]:
```
# Extracting only Text attributs for analysis
text_df = df.drop(['id', 'user_name', 'user_location', 'user_description',␣
 ↪'user_created',
       'user_followers', 'user_friends', 'user_favourites', 'user_verified',
        'date', 'hashtags', 'source', 'retweets', 'favorites',
        'is_retweet'],axis=1)
text_df.head()
```

```
[6]:                                                 text
     0  Same folks said daikon paste could treat a cyt…
     1  While the world has been on the wrong side of …
     2  #coronavirus #SputnikV #AstraZeneca #PfizerBio…
     3  Facts are immutable, Senator, even when you're…
     4  Explain to me again why we need a vaccine @Bor…
```

```
[7]: #visualizing Raw data we have from Tweetr
     print(text_df["text"].iloc[0],"\n")
     print(text_df["text"].iloc[1],"\n")
     print(text_df["text"].iloc[2],"\n")
     print(text_df["text"].iloc[3],"\n")
     print(text_df["text"].iloc[4],"\n")
     print(text_df["text"].iloc[5],"\n")
```

```
Same folks said daikon paste could treat a cytokine storm #PfizerBioNTech
https://t.co/xeHhIMg1kF

While the world has been on the wrong side of history this year, hopefully, the
biggest vaccination effort we've ev… https://t.co/dlCHrZjkhm

#coronavirus #SputnikV #AstraZeneca #PfizerBioNTech #Moderna #Covid_19 Russian
vaccine is created to last 2-4 years… https://t.co/ieYlCKBr8P

Facts are immutable, Senator, even when you're not ethically sturdy enough to
acknowledge them. (1) You were born i… https://t.co/jqgV18kch4

Explain to me again why we need a vaccine @BorisJohnson @MattHancock
#whereareallthesickpeople #PfizerBioNTech… https://t.co/KxbSRoBEHq

Does anyone have any useful advice/guidance for whether the COVID vaccine is
safe whilst breastfeeding?… https://t.co/EifsyQoeKN
```

Data Preprocessing

```
[8]: def data_processing(text):
         text = text.lower()      #Converting to text to lowercase
         text = re.sub(r'https\S+|www\S+https\S+','',text,flags=re.MULTILINE)  ␣
      ↪#Removing URL
         text = re.sub(r'\@w+|\#','',text)          #Removing hashtags
         text = re.sub(r'[^\w\s]','',text)          #Removing hashtags
         text_tokens = word_tokenize(text)          #Getting tokens
         filtered_text = [w for w in text_tokens if not w in stop_words]
```

```
    return " ".join(filtered_text)
```

[9]:
```
# Applying Data Processing function
text_df.text = text_df["text"].apply(data_processing)
```

[10]:
```
# Removing Duplicates if any
text_df = text_df.drop_duplicates('text')
```

[11]:
```
# Performing Stemming
stemmer = PorterStemmer()
def stemming(data):
    text = [stemmer.stem(word) for word in data]
    return data
```

[12]:
```
text_df["text"] = text_df["text"].apply(lambda x: stemming(x))
```

[13]:
```
#visualizing Processed text
print(text_df["text"].iloc[0],"\n")
print(text_df["text"].iloc[1],"\n")
print(text_df["text"].iloc[2],"\n")
print(text_df["text"].iloc[3],"\n")
print(text_df["text"].iloc[4],"\n")
print(text_df["text"].iloc[5],"\n")
```

```
folks said daikon paste could treat cytokine storm pfizerbiontech

world wrong side history year hopefully biggest vaccination effort weve ev

coronavirus sputnikv astrazeneca pfizerbiontech moderna covid_19 russian vaccine
created last 24 years

facts immutable senator even youre ethically sturdy enough acknowledge 1 born

explain need vaccine borisjohnson matthancock whereareallthesickpeople
pfizerbiontech

anyone useful adviceguidance whether covid vaccine safe whilst breastfeeding
```

[14]:
```
#Checking data shape
print("Shape of data after processing:",text_df["text"].shape)
```

```
Shape of data after processing: (10543,)
```

[15]:
```
#calculating polarity for categorizing text
def polarity(text):
    return TextBlob(text).sentiment.polarity
```

[16]:
```
text_df["polarity"] = text_df["text"].apply(polarity)
text_df.head(10)
```

[16]:
```
                                                text  polarity
0  folks said daikon paste could treat cytokine s...     0.000
1  world wrong side history year hopefully bigges...    -0.500
2  coronavirus sputnikv astrazeneca pfizerbiontec...     0.000
3  facts immutable senator even youre ethically s...     0.100
4  explain need vaccine borisjohnson matthancock ...     0.000
5  anyone useful adviceguidance whether covid vac...     0.400
6  bit sad claim fame success vaccination patriot...    -0.100
7  many bright days 2020 best 1 bidenharris winni...     0.675
8  covid vaccine getting covidvaccine covid19 pfi...     0.000
9  covidvaccine states start getting covid19vacci...     0.000
```

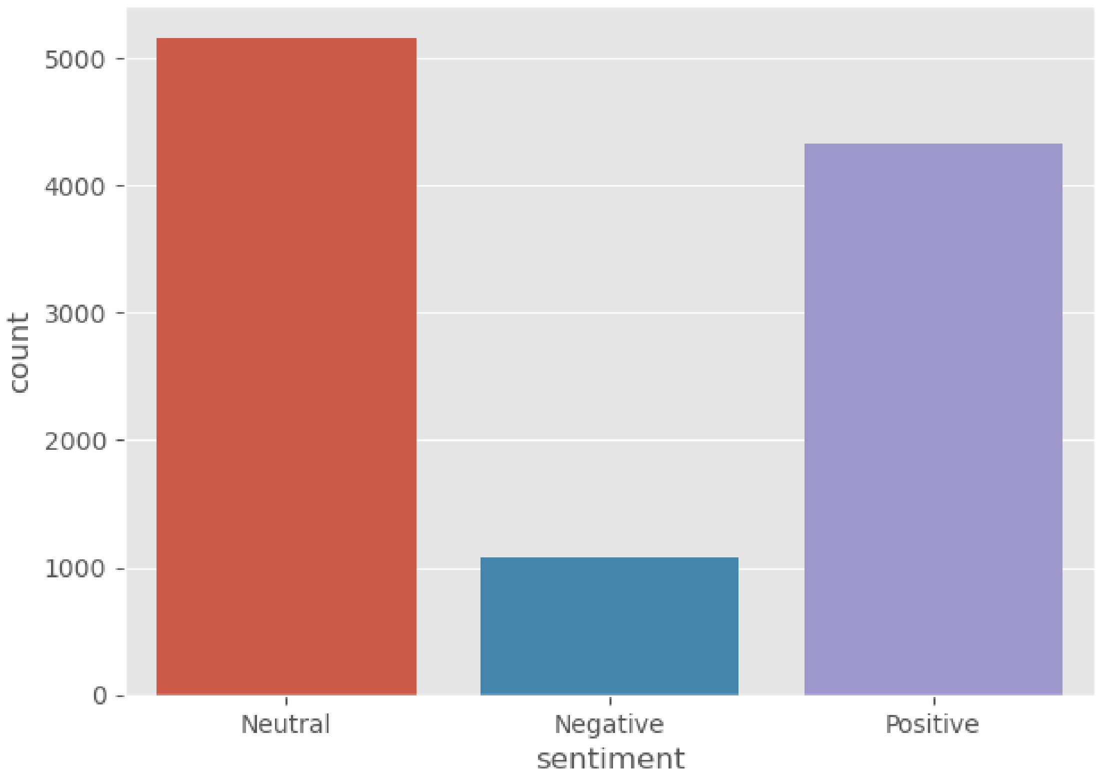
[17]:
```
#  Adding Sentiment to the data frame
def sentiment(label):
    if label <0:
        return "Negative"
    elif label ==0:
        return "Neutral"
    elif label>0:
        return "Positive"
```

[18]:
```
text_df['sentiment'] = text_df['polarity'].apply(sentiment)
text_df.head(10)
```

[18]:
```
                                                text  polarity sentiment
0  folks said daikon paste could treat cytokine s...     0.000   Neutral
1  world wrong side history year hopefully bigges...    -0.500  Negative
2  coronavirus sputnikv astrazeneca pfizerbiontec...     0.000   Neutral
3  facts immutable senator even youre ethically s...     0.100  Positive
4  explain need vaccine borisjohnson matthancock ...     0.000   Neutral
5  anyone useful adviceguidance whether covid vac...     0.400  Positive
6  bit sad claim fame success vaccination patriot...    -0.100  Negative
7  many bright days 2020 best 1 bidenharris winni...     0.675  Positive
8  covid vaccine getting covidvaccine covid19 pfi...     0.000   Neutral
9  covidvaccine states start getting covid19vacci...     0.000   Neutral
```

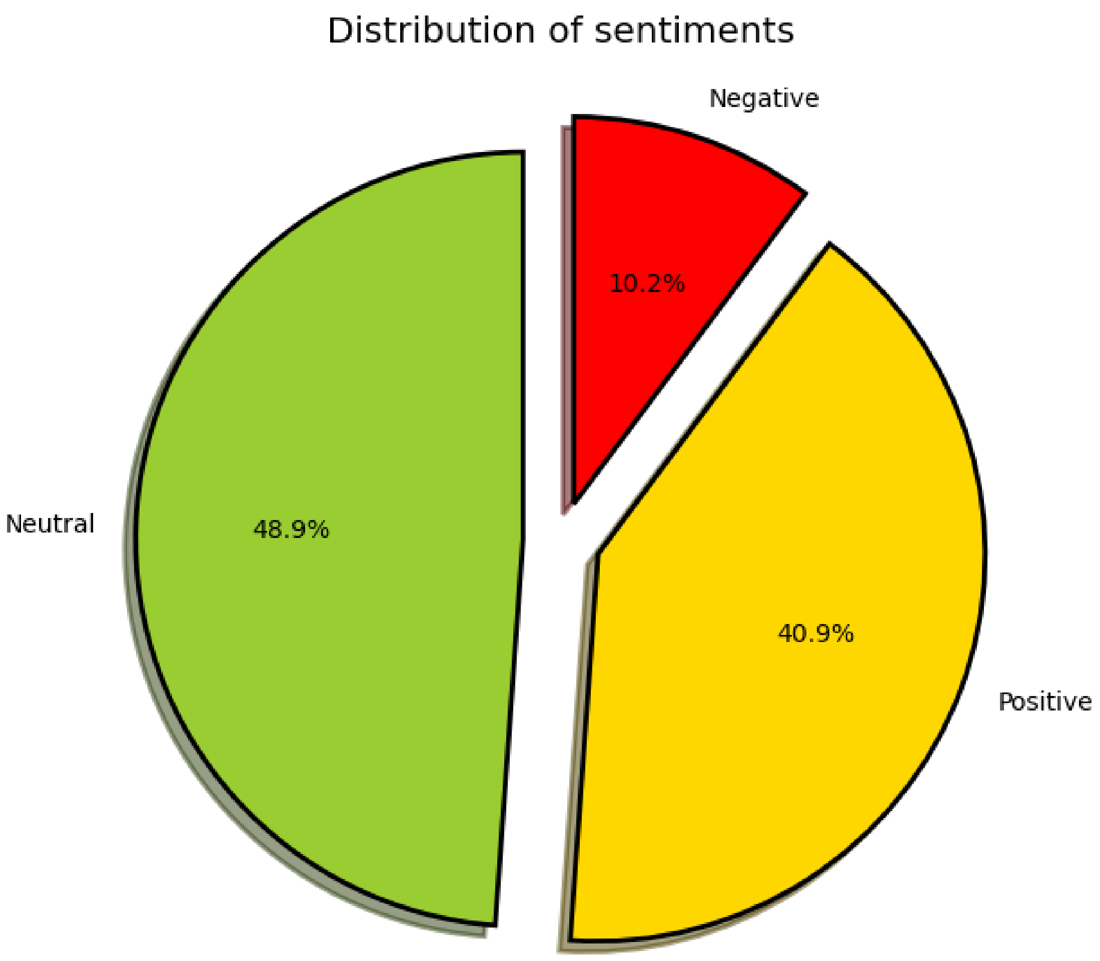
[19]:
```
#Visualizing the Sentiment
fig = plt.figure(figsize=(7,5))
sns.countplot(x="sentiment",data=text_df)
```

[19]:
```
<Axes: xlabel='sentiment', ylabel='count'>
```



[20]:
```
fig = plt.figure(figsize=(7,7))
colors = ("yellowgreen", "gold", "red")
wp = {'linewidth':2, 'edgecolor':"black"}
tags = text_df['sentiment'].value_counts()
explode = (0.1,0.1,0.1)
tags.plot(kind='pie', autopct='%1.1f%%', shadow=True, colors = colors,
         startangle=90, wedgeprops = wp, explode = explode, label='')
plt.title('Distribution of sentiments')
```

[20]:
```
Text(0.5, 1.0, 'Distribution of sentiments')
```

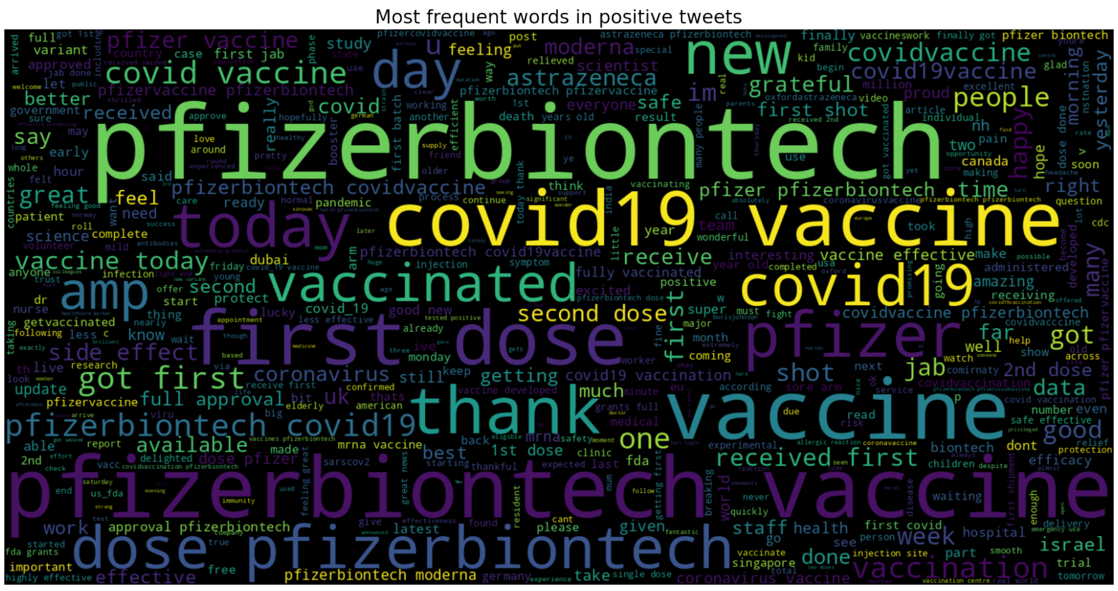


```
[21]: #Visulaizing Top 5 positive Sentiments
      pos_tweets = text_df[text_df.sentiment == 'Positive']
      pos_tweets = pos_tweets.sort_values(['polarity'], ascending= False)
      pos_tweets.head()
```

```
[21]:                                                   text  polarity sentiment
      9317  best way get merrygoround pfizer pfizerbiontec…       1.0  Positive
      2340  applying emotion pfizerbiontech based best evi…       1.0  Positive
      6295  pfizer jab morning efficient wellorganised tha…       1.0  Positive
      5041  get art printed awesome products support redbu…       1.0  Positive
      1055  already vaccinated getting vaccine soon plan t…       1.0  Positive
```

```
[22]: text = ' '.join([word for word in pos_tweets['text']])
      plt.figure(figsize=(20,15), facecolor='None')
      wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
```

```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in positive tweets', fontsize=19)
plt.show()
```

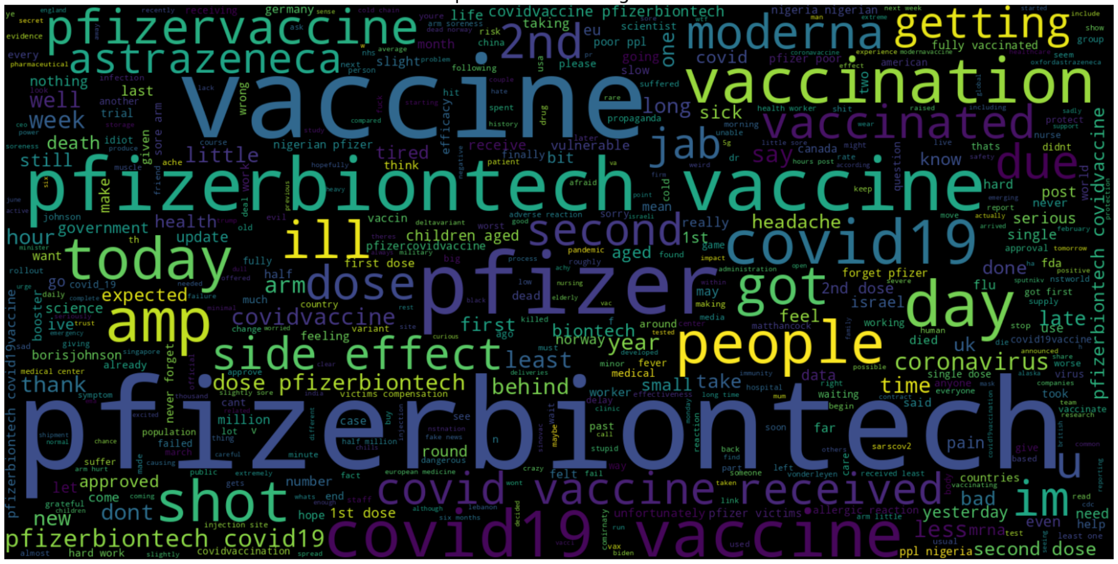


[23]:
```
#Visualizing Negative Words
neg_tweets = text_df[text_df.sentiment == 'Negative']
neg_tweets = neg_tweets.sort_values(['polarity'], ascending= False)
neg_tweets.head()
```

[23]:
```
                                                  text  polarity sentiment
2912  work skilled nursing facility got first vaccin… -0.003333  Negative
7256  200321 752308 vaccinations new daily record da… -0.003409  Negative
2073  ukgovernment cant even vaccinate properly ethi… -0.004762  Negative
7715  got first dose less waiting time airport vacci… -0.005556  Negative
7157  nas_k27 second dose due end next month well fa… -0.006250  Negative
```

[24]:
```
text = ' '.join([word for word in neg_tweets['text']])
plt.figure(figsize=(20,15), facecolor='None')
wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Most frequent words in negative tweets', fontsize=19)
plt.show()
```

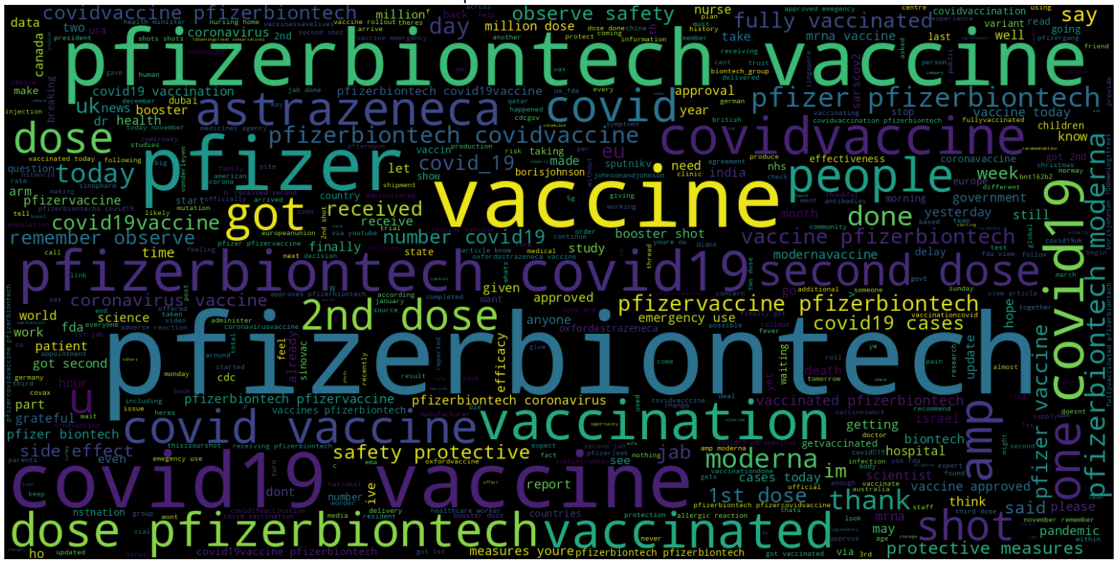
Most frequent words in negative tweets



```
[25]: #Visualizing Neutral Words
      neutral_tweets = text_df[text_df.sentiment == 'Neutral']
      neutral_tweets = neutral_tweets.sort_values(['polarity'], ascending= False)
      neutral_tweets.head()
```

```
[25]:                                                 text  polarity sentiment
      0     folks said daikon paste could treat cytokine s…       0.0   Neutral
      7347  anyone else feel like framing vaccine card pfi…       0.0   Neutral
      7458  looking forward getting second pfizer shot any…       0.0   Neutral
      7454  never thought id running diff vaccine modernav…       0.0   Neutral
      7453  john___m dont get choose one person know asked…       0.0   Neutral
```

```
[26]: text = ' '.join([word for word in neutral_tweets['text']])
      plt.figure(figsize=(20,15), facecolor='None')
      wordcloud = WordCloud(max_words=500, width=1600, height=800).generate(text)
      plt.imshow(wordcloud, interpolation='bilinear')
      plt.axis("off")
      plt.title('Most frequent words in neutral tweets', fontsize=19)
      plt.show()
```



Most frequent words in neutral tweets

Vectorizing Data

```
[27]: # Performing Vectorizing to crate bigram model
      vect = CountVectorizer(ngram_range=(1,2)).fit(text_df['text'])
```

```
[28]: #Getting Features
      feature_names = vect.get_feature_names_out()
      print("Number of features: {}\n".format(len(feature_names)))
      print("First 20 features:\n {}".format(feature_names[:20]))
```

```
Number of features: 78583

First 20 features:
 ['000' '000 doses' '000 initial' '000 people' '000 vaccines' '0000001'
 '0000001 covid19' '0011' '0011 abt' '004' '004 covid' '004 israelis' '01'
 '01 getting' '01 june' '01 november' '01aug2021' '01aug2021 doublevaxxed'
 '02' '02 175']
```

Model Development

```
[29]: #seperating Independent and Depentent Variables and tranform X data
      X = text_df['text']
      Y = text_df['sentiment']
      X = vect.transform(X)
```

```
[30]: # Splitting data with test 20%
      x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,␣
       ↪random_state=42)
```

[31]:
```
#Checking shape of train and test data
print("Size of x_train:", (x_train.shape))
print("Size of y_train:", (y_train.shape))
print("Size of x_test:", (x_test.shape))
print("Size of y_test:", (y_test.shape))
```

```
Size of x_train: (8434, 78583)
Size of y_train: (8434,)
Size of x_test: (2109, 78583)
Size of y_test: (2109,)
```

[32]:
```
import warnings
warnings.filterwarnings('ignore')

#Training logisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
logreg_pred = logreg.predict(x_test)
logreg_acc = accuracy_score(logreg_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
```

```
Test accuracy: 84.64%
```

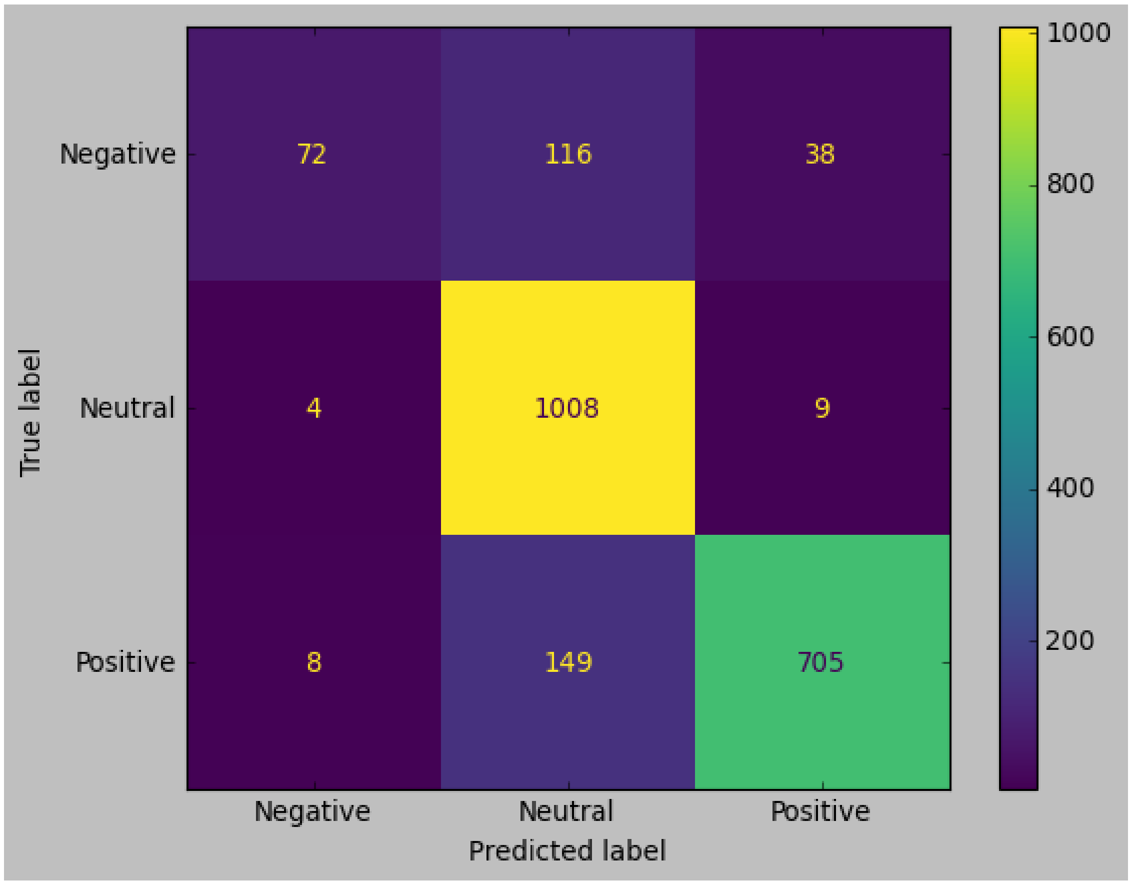
[33]:
```
#Confusion matrix
print(confusion_matrix(y_test, logreg_pred))
print("\n")
print(classification_report(y_test, logreg_pred))
```

```
[[  72  116   38]
 [   4 1008    9]
 [   8  149  705]]


              precision    recall  f1-score   support

    Negative       0.86      0.32      0.46       226
     Neutral       0.79      0.99      0.88      1021
    Positive       0.94      0.82      0.87       862

    accuracy                           0.85      2109
   macro avg       0.86      0.71      0.74      2109
weighted avg       0.86      0.85      0.83      2109
```

[34]:
```
style.use('classic')
cm = confusion_matrix(y_test, logreg_pred, labels=logreg.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels=logreg.
 ↪classes_)
```

```
disp.plot()
```

[34]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7e1c18912aa0>
```



Tuning Model

[35]:
```
#Lets perform Hyper-Parameter to modulate performance of model

param_grid={'C':[0.001, 0.01, 0.1, 1, 10]}                   #Taking random ␣
 ↪alpha values
grid = GridSearchCV(LogisticRegression(), param_grid)
grid.fit(x_train, y_train)
```

[35]:
```
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10]})
```

[36]:
```
print("Best parameters:", grid.best_params_)
```

```
Best parameters: {'C': 10}
```

[37]:
```
y_pred = grid.predict(x_test)
logreg_acc = accuracy_score(y_pred, y_test)
print("Test accuracy: {:.2f}%".format(logreg_acc*100))
```

```
Test accuracy: 85.92%
```

1 we can see increase in accurancy by impementing hyperparameter

Thank you for Joining, Happy Kaggling