

## ✓ X-Mentor

### Introduction

Mental health challenges among university students, particularly at Dedan Kimathi University, present a significant concern with potential consequences on academic performance, daily functioning, and overall well-being. Despite the increasing prevalence of mental health issues, there is often a lack of proactive mechanisms for early detection and support within the university community. The existing gap necessitates an innovative solution that aligns with the contemporary communication landscape, particularly on Twitter. The problem at hand is the absence of a dedicated and accessible platform for identifying and addressing potential mental health concerns expressed by Dedan Kimathi University students on Twitter. The traditional support systems may not be readily accessible, and the stigmatization of mental health discussions further compounds the challenge. Existing approaches fall short of providing a real-time, user-centric solution that comprehensively analyzes user language patterns to flag and respond to potential mental health issues. The lack of a targeted AI-based intervention contributes to the delayed or inadequate support for students expressing distress on social media. As such, the development of an AI-based Mental Health Detection and Support Bot is imperative to bridge this gap. The outlined objectives aim to provide a user-friendly tool that not only identifies language indicative of mental health concerns but also responds with timely and supportive messages tailored to the unique context of Dedan Kimathi University. Through the proposed research questions, the project seeks to uncover the language patterns relevant to this specific university context, assess the bot's efficacy, and gather user feedback to continually improve and tailor the intervention for optimal impact within the Dedan Kimathi University student community.

### General Objective

Develop a user-friendly AI-based bot for Twitter that analyzes user language to flag potential mental health concerns, provides help, and specifically target Dedan Kimathi University students.

## Specific objectives

1. Implement a language analysis algorithm to detect key indicators of mental health issues, such as curse words and distress expressions, in Twitter posts from Dedan Kimathi University students.
2. Integrate the bot with the Twitter API to monitor tweets in real-time, flagging and responding to identified mental health concerns with supportive messages.
3. Create a user-friendly web interface for the bot, allowing users to easily access and understand flagged tweets and the provided help messages.

### ✓ 1. Importing Libraries

```
1 import numpy as np
2 import pandas as pd
3
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6
7 import matplotlib.pyplot as plt
8 import plotly.graph_objects as go
9 from plotly.subplots import make_subplots
10 import plotly.io as pio
11
12
13 import warnings
```

### ✓ 2. Viewing Data



```
1 train = pd.read_csv('/content/train_tweet.csv')
2 test = pd.read_csv('/content/test_tweets.csv')
3
4 print(train.shape)
```

```
5 print(test.shape)
```



```
(31962, 3)
(17197, 2)
```

*There are 31962 training tweets and 17197 testing tweets.*

```
1 train.head()
```

	id	label	tweet	
0	1	0	@user when a father is dysfunctional and is s...	
1	2	0	@user @user thanks for #lyft credit i can't us...	
2	3	0	bihday your majesty	
3	4	0	#model i love u take with u all the time in ...	
4	5	0	factsguide: society now #motivation	

```
1 test.head()
```

	id	tweet	
0	31963	#studiolife #aislife #requires #passion #dedic...	
1	31964	@user #white #supremacists want everyone to s...	
2	31965	safe ways to heal your #acne!! #altwaystohe...	
3	31966	is the hp and the cursed child book up for res...	
4	31967	3rd #bihday to my amazing, hilarious #nephew...	



```
1 train.isnull().any()
2 test.isnull().any()
```

```
id      False
tweet   False
dtype: bool
```

*Conclusion: The data is fairly clean and has no missing values*


## ✓ Checking out the negative comments from the train set

```
1 train[train['label'] == 0].head(10)
```

	id	label	tweet	
0	1	0	@user when a father is dysfunctional and is s...	
1	2	0	@user @user thanks for #lyft credit i can't us...	
2	3	0	bihday your majesty	
3	4	0	#model i love u take with u all the time in ...	
4	5	0	factsguide: society now #motivation	
5	6	0	[2/2] huge fan fare and big talking before the...	
6	7	0	@user camping tomorrow @user @user @user @use...	
7	8	0	the next school year is the year for exams.ð□□...	
8	9	0	we won!!! love the land!!! #allin #cavs #champ...	
9	10	0	@user @user welcome here ! i'm it's so #gr...	

## ✓ Checking out the positive comments from the train set

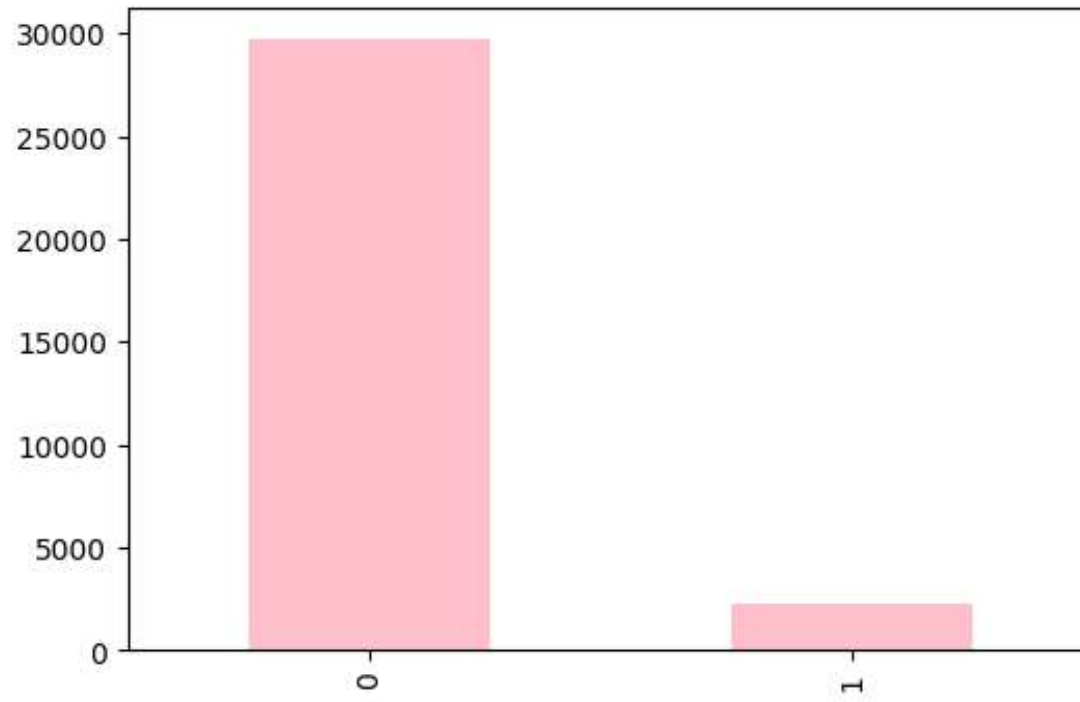
```
1 train[train['label'] == 1].head(10)
```

	id	label	tweet	
13	14	1	@user #cnn calls #michigan middle school 'bui...	
14	15	1	no comment! in #australia #opkillingbay #se...	
17	18	1	retweet if you agree!	
23	24	1	@user @user lumpy says i am a . prove it lumpy.	
34	35	1	it's unbelievable that in the 21st century we'...	
56	57	1	@user lets fight against #love #peace	
68	69	1	ðŸŒŸ@the white establishment can't have blk fol...	
77	78	1	@user hey, white people: you can call people '...	
82	83	1	how the #altright uses & insecurity to lu...	
111	112	1	@user i'm not interested in a #linguistics tha...	

## ✓ Value Counts Visualization

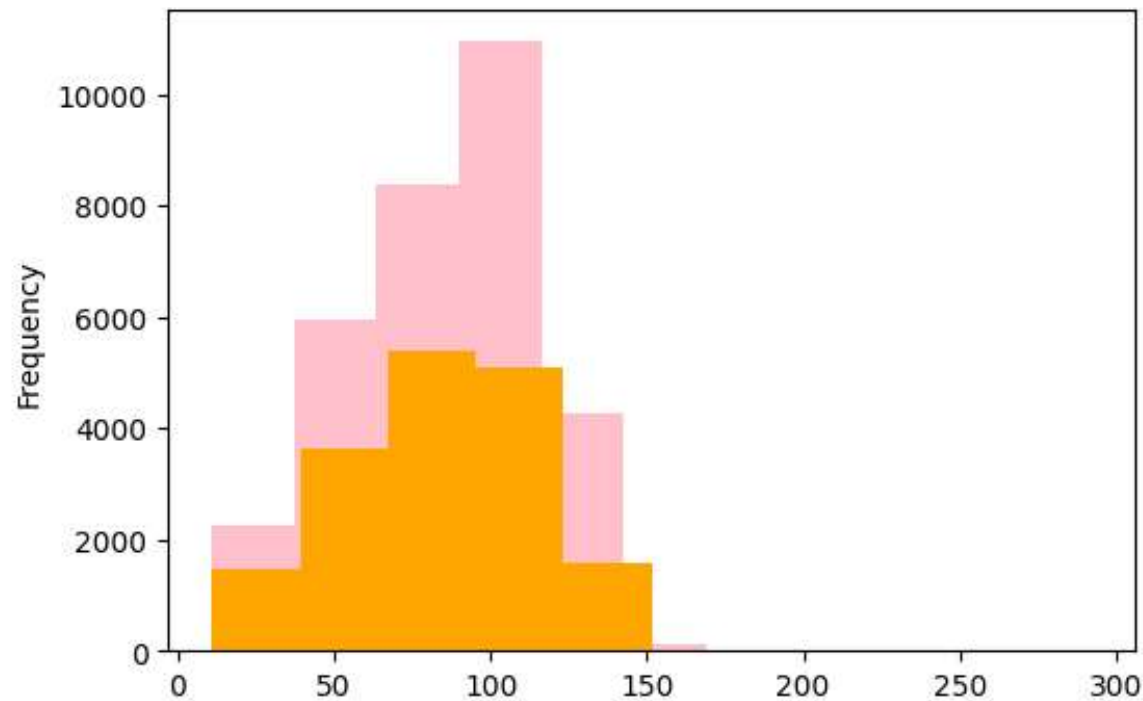
```
1 train['label'].value_counts().plot.bar(color = 'pink', figsize = (6, 4))
```

&lt;Axes: &gt;



### ✓ Checking the distribution of tweets in the data

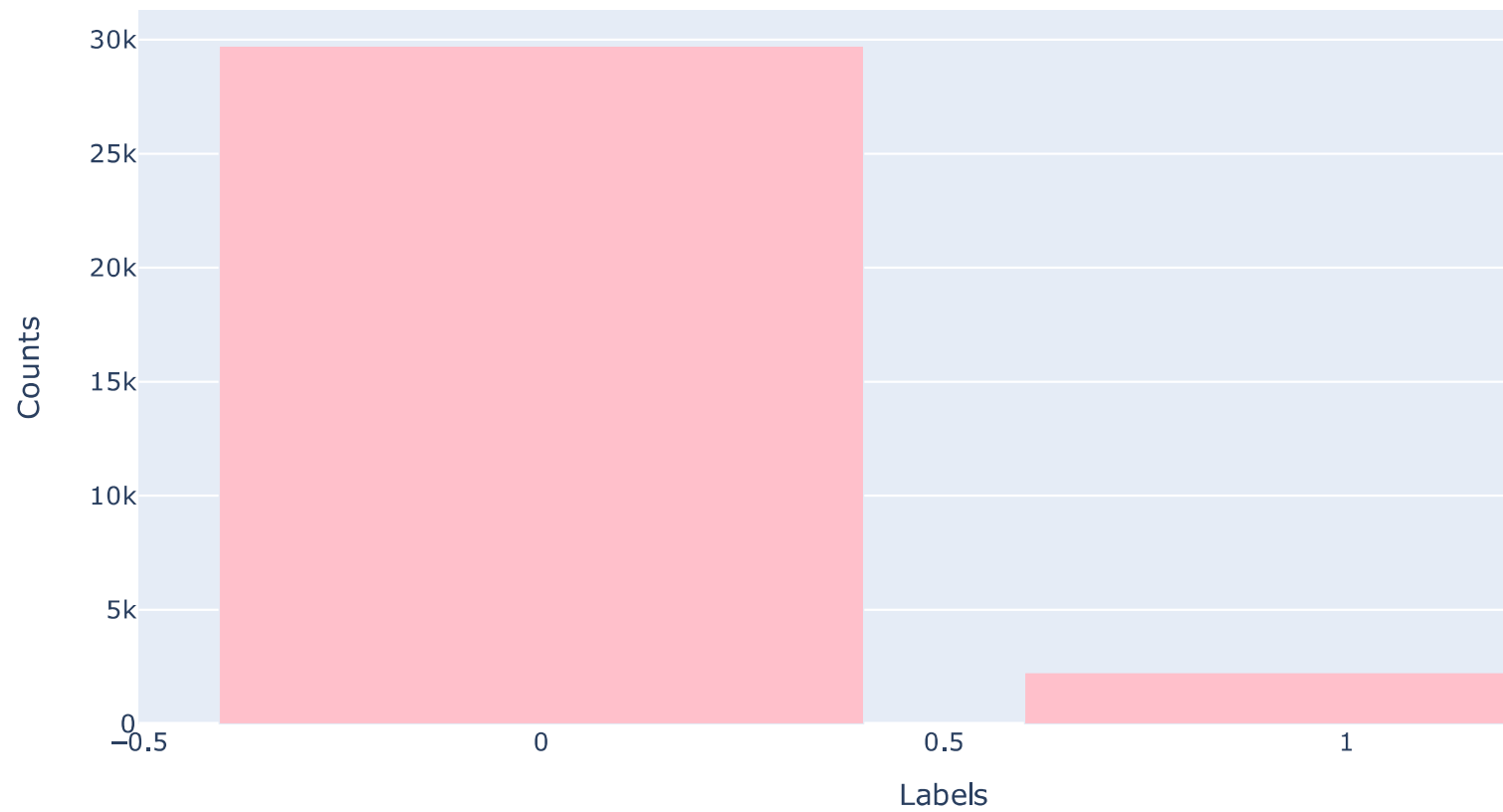
```
1 length_train = train['tweet'].str.len().plot.hist(color = 'pink', figsize = (6, 4))  
2 length_test = test['tweet'].str.len().plot.hist(color = 'orange', figsize = (6, 4))
```



✓ Exporting positive\_vs\_negative\_plot.html

```
1 fig = make_subplots(rows=1, cols=1)
2 fig.add_trace(go.Bar(x=train['label'].value_counts().index, y=train['label'].value_counts().values, marker_color
3
4 fig.update_layout(title='Positive vs Negative Counts', xaxis_title='Labels', yaxis_title='Counts')
5
```

## Positive vs Negative Counts



### Exporting tweets\_distribution.html

```
1 fig = make_subplots(rows=1, cols=1)
2
3 length_train = train['tweet'].str.len()
4 fig.add_trace(go.Histogram(x=length_train, marker_color='pink'), row=1, col=1)
5
```



```

5
6 length_test = test['tweet'].str.len()
7 fig.add_trace(go.Histogram(x=length_test, marker_color='orange'), row=1, col=1)
8
9 fig.update_layout(title_text='Distribution of Tweet Lengths',
10                   xaxis_title_text='Tweet Length',
11                   yaxis_title_text='Frequency')
12
13 pio.write_html(fig, file='tweets_distribution.html')



```

## ✓ Adding a column to represent the length of the tweet

```

1 train['len'] = train['tweet'].str.len()
2 test['len'] = test['tweet'].str.len()
3
4 train.head(10)

```

	id	label	tweet	len	
0	1	0	@user when a father is dysfunctional and is s...	102	
1	2	0	@user @user thanks for #lyft credit i can't us...	122	
2	3	0	bihday your majesty	21	
3	4	0	#model i love u take with u all the time in ...	86	
4	5	0	factsguide: society now #motivation	39	
5	6	0	[2/2] huge fan fare and big talking before the...	116	
6	7	0	@user camping tomorrow @user @user @user @use...	74	
7	8	0	the next school year is the year for exams.ð□□...	143	
8	9	0	we won!!! love the land!!! #allin #cavs #champ...	87	
9	10	0	@user @user welcome here ! i'm it's so #gr...	50	

```
1
2 train.groupby('label').describe()
```

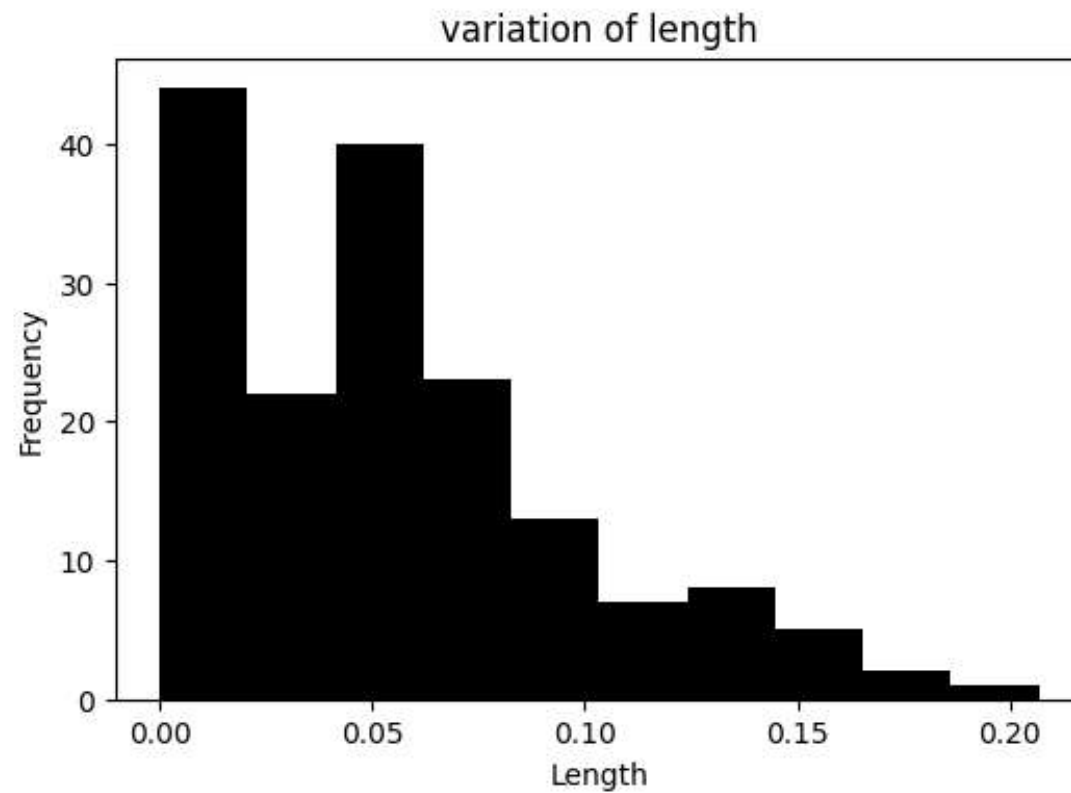
	id								len	
	count	mean	std	min	25%	50%	75%	max	count	mean
label										
0	29720.0	15974.454441	9223.783469	1.0	7981.75	15971.5	23965.25	31962.0	29720.0	84.32
1	2242.0	16074.896075	9267.955758	14.0	8075.25	16095.0	24022.00	31961.0	2242.0	90.18

## ✓ Variation of tweets' length visualization

```
1 train.groupby('len').mean()['label'].plot.hist(color = 'black', figsize = (6, 4),)
2 plt.title('variation of length')
3 plt.xlabel('Length')
4 plt.show()
```

```
<ipython-input-16-250f85d53a70>:1: FutureWarning:
```

The default value of numeric\_only in DataFrameGroupBy.mean is deprecated. In a future version,



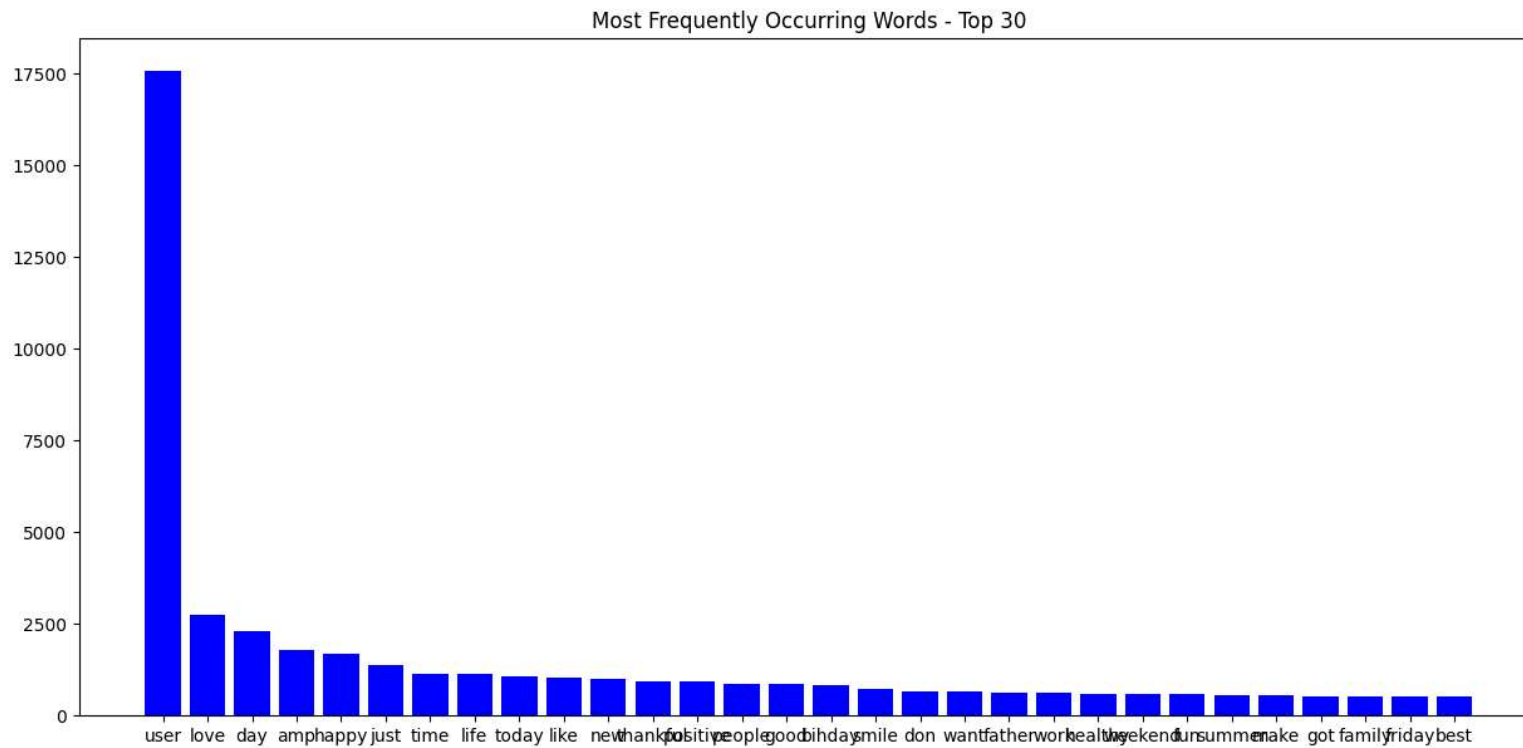
✓ Exporting tweet\_length\_variation.html

```
1 fig = go.Figure()  
2  
3 fig.add_trace(go.Histogram(x=train['len'], marker_color='black'))  
4  
5 fig.update_layout(title_text='Variation of Length',  
6                     xaxis_title_text='Length',  
7                     yaxis_title_text='Mean Label',  
8                     bargap=0.1)  
9  
10 pio.write_html(fig, file='tweet_length_variation.html')
```

## ✓ Most frequently occurring words visualization

```
1 from sklearn.feature_extraction.text import CountVectorizer
2 import matplotlib.pyplot as plt
3 import plotly.graph_objects as go
4 import plotly.io as pio
5 import pandas as pd
6
7 cv = CountVectorizer(stop_words='english')
8 words = cv.fit_transform(train.tweet)
9
10 sum_words = words.sum(axis=0)
11
12 words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.items()]
13 words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
14
15 frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])
16
17 plt.figure(figsize=(15, 7))
18 plt.bar(frequency['word'].head(30), frequency['freq'].head(30), color='blue')
19 plt.title("Most Frequently Occurring Words - Top 30")
20
21 plt.savefig('most_frequently_occurring_words.png')
22
23 fig = go.Figure()
24
25 fig.update_layout(images=[go.layout.Image(
26     source='most_frequently_occurring_words.png',
27     x=0,
28     y=1,
29     xref='paper',
30     yref='paper',
31     sizex=0.5,
32     sizey=0.5,
33     opacity=1,
34     layer='below'
35 )])
36
37 fig.update_layout(title_text='Most Frequently Occurring Words - Top 30', xaxis_title='Word', yaxis_title='Freq
```

```
38  
39 pio.write_html(fig, file='most_frequently_occurring_words.html')  
40
```



```
1 from sklearn.feature_extraction.text import CountVectorizer
2 import plotly.graph_objects as go
3 import plotly.io as pio
4 import pandas as pd
5
6 cv = CountVectorizer(stop_words='english')
7 words = cv.fit_transform(train.tweet)
8
9 sum_words = words.sum(axis=0)
10
11 words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.items()]
12 words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
13
14 frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])
15
16 # Plot with Plotly directly
17 fig = go.Figure()
18
19 fig.add_trace(go.Bar(x=frequency['word'].head(30), y=frequency['freq'].head(30), marker_color='blue'))
20
21 fig.update_layout(title_text='Most Frequently Occurring Words - Top 30', xaxis_title='Word', yaxis_title='Freq
22
23 pio.write_html(fig, file='most_frequently_occurring_words.html')
24
```

## ✓ Exporting most\_frequently\_occurring\_words.html

```
1 from wordcloud import WordCloud
2
3 wordcloud = WordCloud(background_color = 'white', width = 1000, height = 1000).generate_from_frequencies(dict(
4
5 plt.figure(figsize=(10,8))
6 plt.imshow(wordcloud)
7 plt.title("WordCloud - Vocabulary from Reviews", fontsize = 22)
8
```

## WordCloud - Vocabulary from Reviews





## ✓ WordCloud - Vocabulary from Reviews Visualization

```
1 from wordcloud import WordCloud
2 import plotly.graph_objects as go
3 import matplotlib.pyplot as plt
4
5 fig = go.Figure()
6
7 wordcloud = WordCloud(background_color='white', width=1000, height=1000).generate_from_frequencies(dict(words_
8
9 fig.add_trace(go.Image(z=wordcloud.to_array())))
10
11 fig.update_layout(title_text="WordCloud - Vocabulary from Reviews", title_font_size=22)
12
13 fig.write_html('wordCloud_vocabulary_from_reviews.html')
14
```

## ✓ Exporting neutral\_words.html

```
1 from wordcloud import WordCloud
2 import plotly.graph_objects as go
3 import matplotlib.pyplot as plt
4
5 normal_words = ' '.join([text for text in train['tweet'][train['label'] == 0]])
6
7 fig = go.Figure()
8
9 wordcloud = WordCloud(background_color='white', width=800, height=500, random_state = 0, max_font_size = 110).;
10
11 fig.add_trace(go.Image(z=wordcloud.to_array()))
12
13 fig.update_layout(title_text="The Neutral Words", title_font_size=22)
14
15 fig.write_html('neutral_words.html')
```

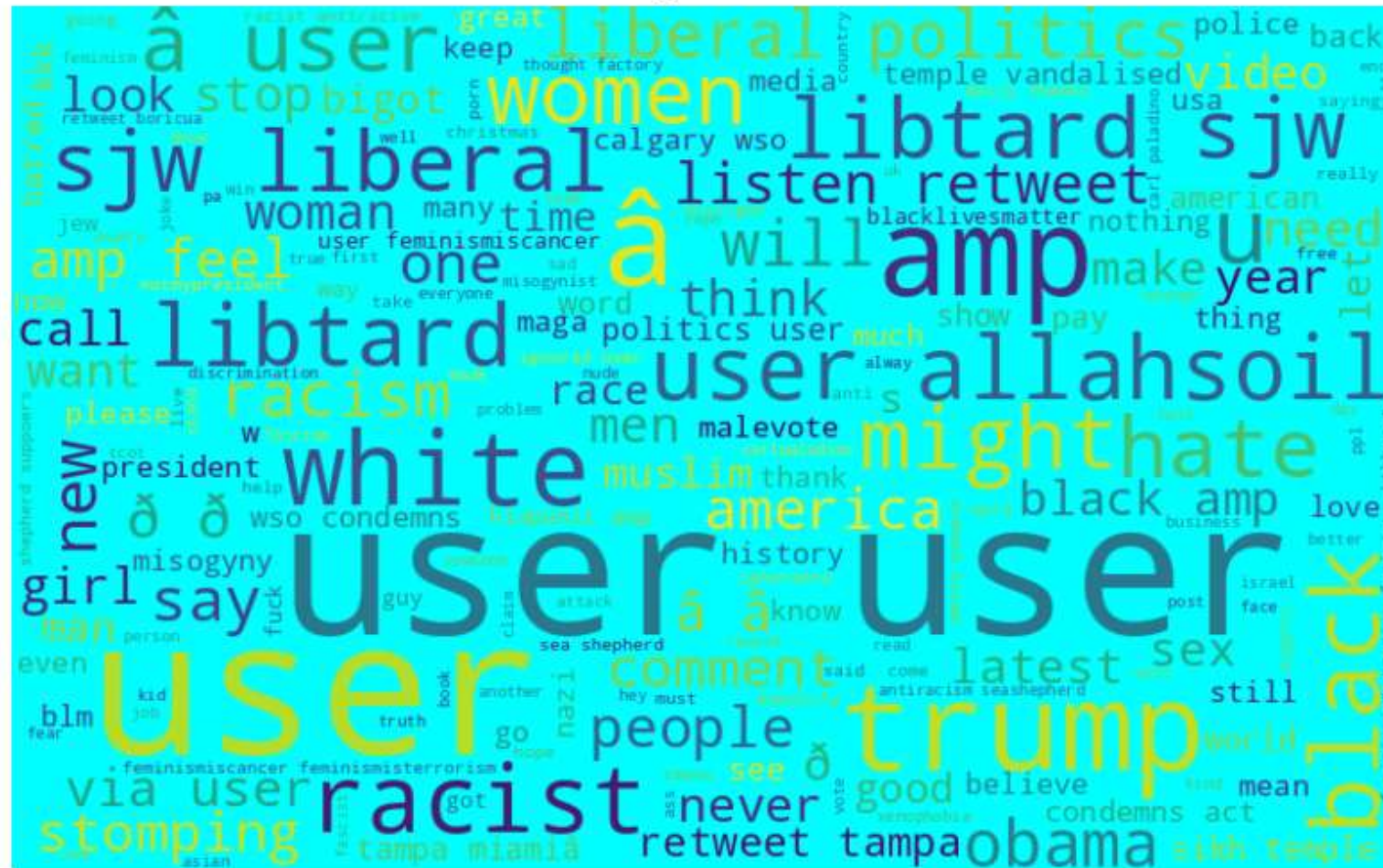
## ✓ The Neutral Words Visualization

```
1 normal_words = ' '.join([text for text in train['tweet'][train['label'] == 0]])
2
3 wordcloud = WordCloud(width=800, height=500, random_state = 0, max_font_size = 110).generate(normal_words)
4 plt.figure(figsize=(10, 7))
5 plt.imshow(wordcloud, interpolation="bilinear")
6 plt.axis('off')
7 plt.title('The Neutral Words')
8 plt.show()
9
```

[illegible]

```
1 negative_words = ' '.join([text for text in train['tweet'][train['label'] == 1]])
2
3 wordcloud = WordCloud(background_color = 'cyan', width=800, height=500, random_state = 0, max_font_size = 110)
4 plt.figure(figsize=(10, 7))
5 plt.imshow(wordcloud, interpolation="bilinear")
6 plt.axis('off')
7 plt.title('The Negative Words')
8 plt.show()
9 fig.write_html('negative_words.html')
```

## The Negative Words



- ▼ Exporting `negative_words.html`

```
1 from wordcloud import WordCloud
2 import plotly.graph_objects as go
3 import matplotlib.pyplot as plt
4
5 negative_words = ' '.join([text for text in train['tweet'][train['label'] == 1]])
6
7 fig = go.Figure()
8
9 wordcloud = WordCloud(background_color = 'white', width=800, height=500, random_state = 0, max_font_size = 110)
10
11 fig.add_trace(go.Image(z=wordcloud.to_array()))
12
13 fig.update_layout(title_text="The Negative Words", title_font_size=22)
14
15 fig.write_html('negative_words.html')
```

## ✓ Collecting the hashtags

```
1 import re
2
3 def hashtag_extract(x):
4     hashtags = []
5
6     for i in x:
7         ht = re.findall(r"#(\w+)", i)
8         hashtags.append(ht)
9
10     return hashtags
```

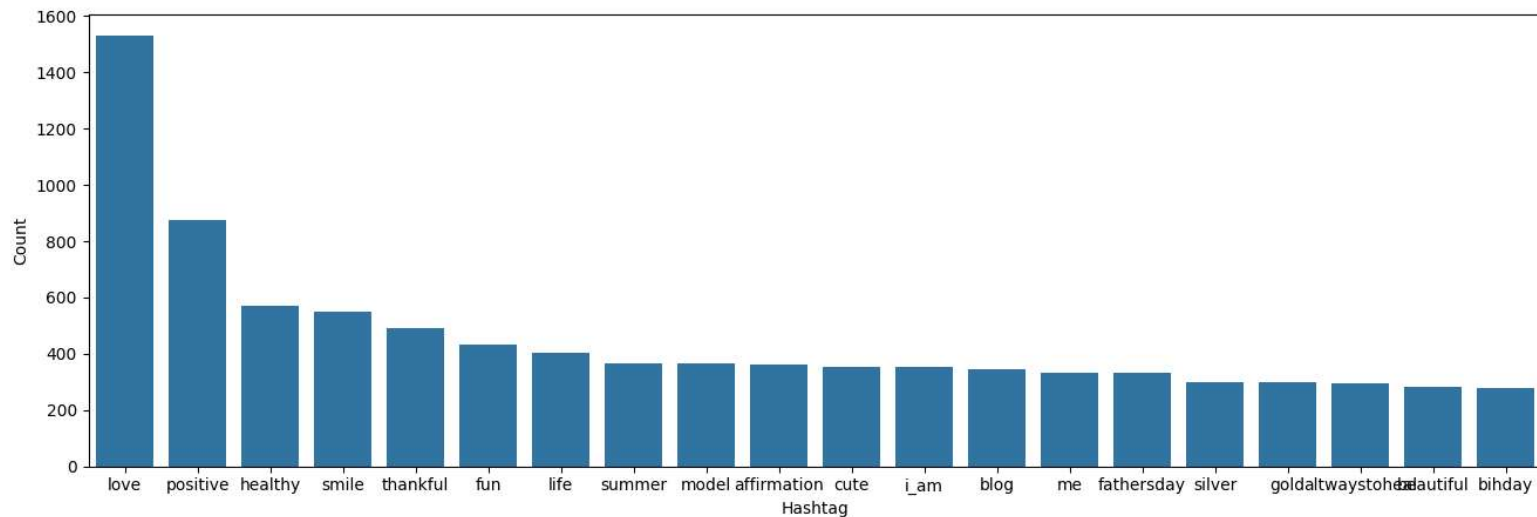
## ✓ Extracting hashtags from non racist/sexist tweets

```
1 HT_regular = hashtag_extract(train['tweet'][train['label'] == 0])
2
3 HT_negative = hashtag_extract(train['tweet'][train['label'] == 1])
4
5 HT_regular = sum(HT_regular,[])
6 HT_negative = sum(HT_negative,[])
```

## ✓ Selecting top 20 most frequent hashtags

```
1 import nltk
2
3 a = nltk.FreqDist(HT_regular)
4 d = pd.DataFrame({'Hashtag': list(a.keys()),
5                   'Count': list(a.values())})
6
7 d = d.nlargest(columns="Count", n = 20)
8 plt.figure(figsize=(16,5))
9 ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
10 ax.set(ylabel = 'Count')
11 plt.show()
12
```





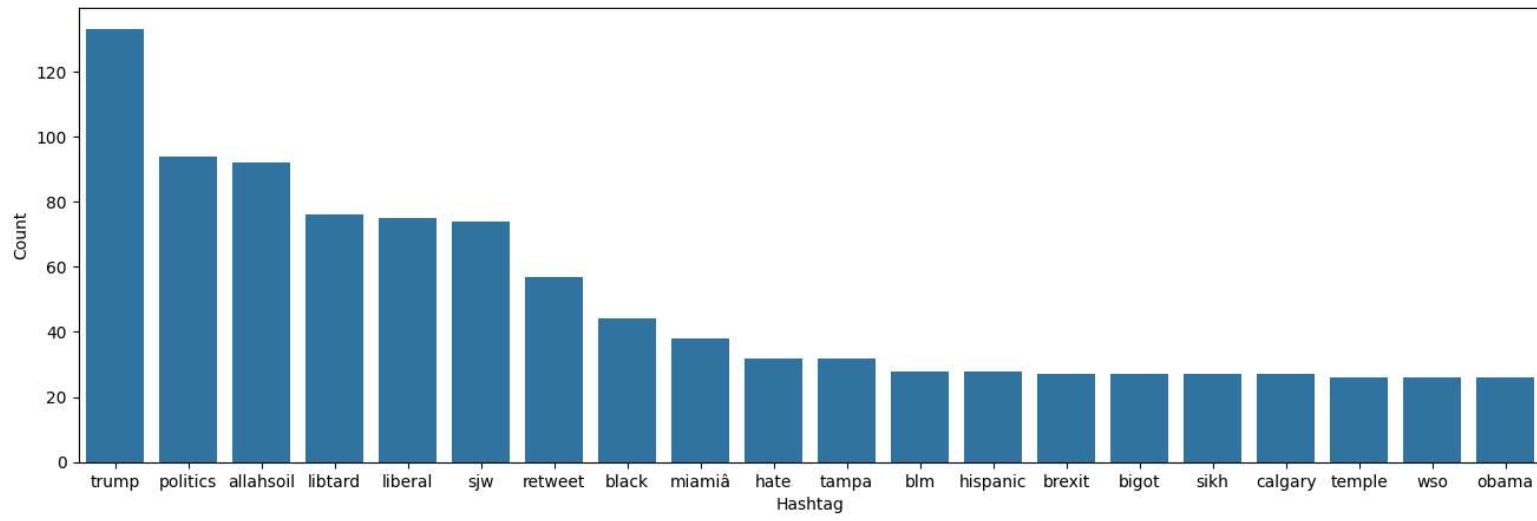
### ✓ Exporting 20\_most\_frequent\_negative\_hashtags.html

```

1 a = nltk.FreqDist(HT_negative)
2 d = pd.DataFrame({'Hashtag': list(a.keys()),
3                  'Count': list(a.values())})
4
5 d = d.nlargest(columns="Count", n = 20)
6 plt.figure(figsize=(16,5))
7 ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
8 ax.set(ylabel = 'Count')
9 plt.show()
10 fig.write_html('20_most_frequent_negative_hashtags.html')

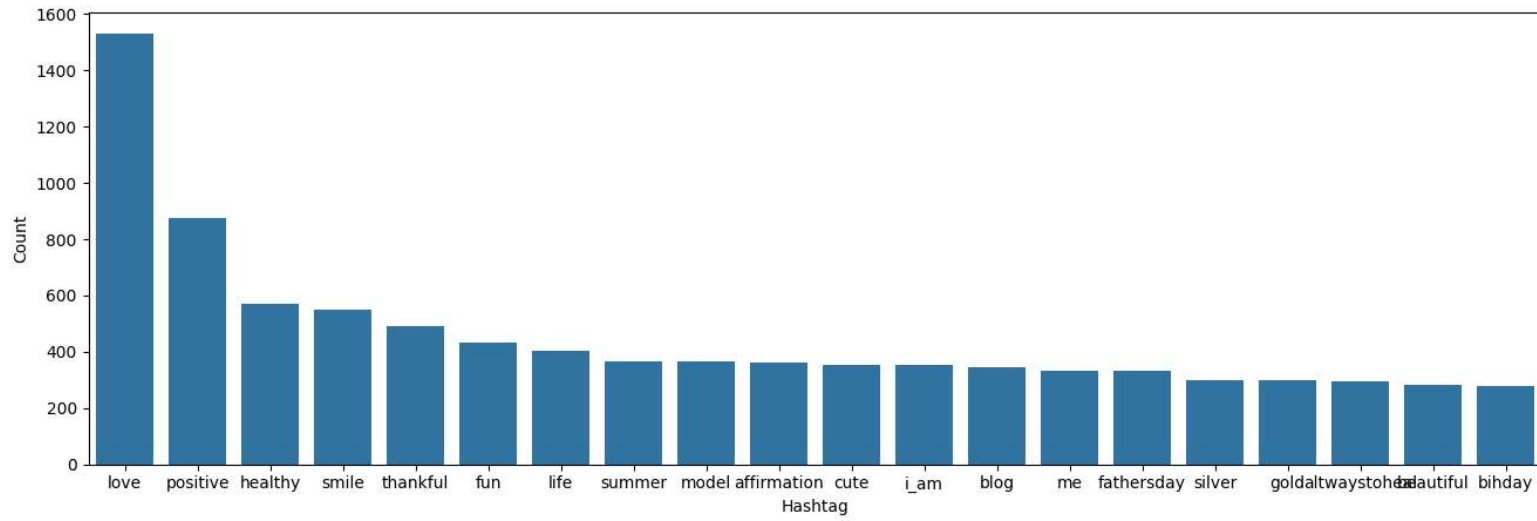
```





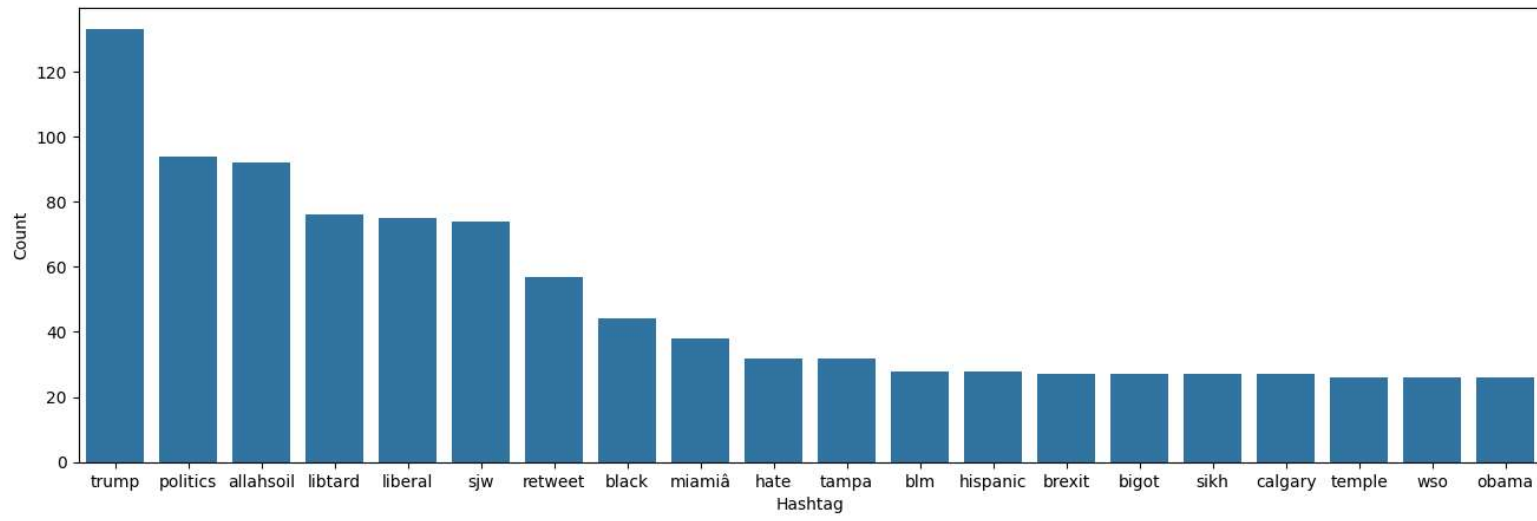
## ✓ Top 20 Most Frequent Neutral Hashtags

```
1 import nltk
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 import plotly.graph_objects as go
6
7 a = nltk.FreqDist(HT_regular)
8 d = pd.DataFrame({'Hashtag': list(a.keys()), 'Count': list(a.values())})
9
10 d = d.nlargest(columns="Count", n=20)
11
12 plt.figure(figsize=(16, 5))
13 ax = sns.barplot(data=d, x="Hashtag", y="Count")
14 ax.set(ylabel='Count')
15 plt.show()
16
17 fig = go.Figure()
18 fig.add_trace(go.Bar(x=d['Hashtag'], y=d['Count'])))
19
20 fig.update_layout(title_text="Top 20 Most Frequent Neutral Hashtags", title_font_size=18)
21
22 fig.write_html('20_most_frequent_neutral_hashtags.html')
23
```



✓ Exporting 20\_most\_frequent\_negative\_hashtags.html

```
1 import nltk
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 import plotly.graph_objects as go
6
7 a = nltk.FreqDist(HT_negative)
8 d = pd.DataFrame({'Hashtag': list(a.keys()), 'Count': list(a.values())})
9
10 d = d.nlargest(columns="Count", n=20)
11
12 plt.figure(figsize=(16, 5))
13 ax = sns.barplot(data=d, x="Hashtag", y="Count")
14 ax.set(ylabel='Count')
15 plt.show()
16
17 fig = go.Figure()
18 fig.add_trace(go.Bar(x=d['Hashtag'], y=d['Count'])))
19
20 fig.update_layout(title_text="Top 20 Most Frequent Negative Hashtags", title_font_size=18)
21
22 fig.write_html('20_most_frequent_negative_hashtags.html')
23
```



✓ Tokenizing the words present in the training set

```
1 tokenized_tweet = train['tweet'].apply(lambda x: x.split())
2
3 import gensim
4
5 model_w2v = gensim.models.Word2Vec(
6     tokenized_tweet,
7     vector_size=200,
8     window=5,
9     min_count=2,
10    sg=1,
11    hs=0,
12    negative=10,
13    workers=2,
14    seed=34
15 )
16
17 model_w2v.train(tokenized_tweet, total_examples=len(train['tweet']), epochs=20)
18
```

WARNING:gensim.models.word2vec:Effective 'alpha' higher than previous training cycles (6109793, 8411580)

## ✓ Testing

```
1 model_w2v.wv.most_similar(positive = "dinner")

[('spaghetti', 0.6713457703590393),
 ('#prosecco', 0.6262232065200806),
 ('#wanderlust', 0.6013128757476807),
 ('fluffy', 0.5971730947494507),
 ('#deutschland', 0.5880066752433777),
 ('#restaurant', 0.5830743312835693),
 ('7!', 0.5817543268203735),
 ('#boardgames', 0.5783932209014893),
 ('coaching', 0.5779764652252197),
```

```
('ð\x9f\x91\x8dð\x9f\x8f»ð\x9f\x91\x8dð\x9f\x8f»ð\x9f\x91\x8dð\x9f\x8f»â\x9dð\x9f\x8fâ\x9dð\x9f\x8f',
0.5766583681106567)]
```

```
1 model_w2v.wv.most_similar(positive = "cancer")
```

```
[('champion,', 0.7048082947731018),
('level.', 0.692138135433197),
('ways.', 0.6886024475097656),
('#merica', 0.6869560480117798),
('ownership', 0.6828452944755554),
('intelligent', 0.6823770403862),
('aol', 0.6817699670791626),
('spots.', 0.681427001953125),
('tolerance', 0.6802441477775574),
('law.', 0.6794952750205994)]
```

```
1 model_w2v.wv.most_similar(positive = "apple")
```

```
[('mytraining', 0.7078200578689575),
('"mytraining"', 0.7039957642555237),
('training"', 0.6851895451545715),
('app,', 0.6437063217163086),
('"my', 0.6053446531295776),
('app', 0.5964861512184143),
('bees', 0.5817535519599915),
('heroku', 0.5761133432388306),
('ta', 0.5743971467018127),
("domino's", 0.5632344484329224)]
```

```
1 model_w2v.wv.most_similar(negative = "hate")
```

```
[('#staup', 0.022724879905581474),
('â\x9c\x88i,\x8f', 0.020933523774147034),
('#ireland', 0.003975582774728537),
('#css', 0.003937653731554747),
('#foodie', 0.0003654570609796792),
('street,', -0.0023828463163226843),
('#ebay', -0.004691518377512693),
```

```
( 'lion', -0.005943953525274992),
( '#babies', -0.005968964658677578),
( '#inlove', -0.006628380157053471)]
```

## ✓ Labelling Tweets

```
1 from tqdm import tqdm
2 tqdm.pandas(desc="progress-bar")
3 from gensim.models.doc2vec import TaggedDocument
```

```
1 def add_label(twt):
2     output = []
3     for i, s in zip(twt.index, twt):
4         output.append(TaggedDocument(words=s, tags=["tweet_" + str(i)]))
5     return output
6
7 labeled_tweets = add_label(tokenized_tweet)
8
9 labeled_tweets[:6]
```

```
[TaggedDocument(words=['@user', 'when', 'a', 'father', 'is', 'dysfunctional', 'and', 'is', 'so', 'selfish',
'he', 'drags', 'his', 'kids', 'into', 'his', 'dysfunction.', '#run'], tags=['tweet_0']),
TaggedDocument(words=['@user', '@user', 'thanks', 'for', '#lyft', 'credit', 'i', "can't", 'use', 'cause',
'they', "don't", 'offer', 'wheelchair', 'vans', 'in', 'pdx.', '#disappointed', '#getthanked'], tags=
['tweet_1']),
TaggedDocument(words=['bihday', 'your', 'majesty'], tags=['tweet_2']),
TaggedDocument(words=['#model', 'i', 'love', 'u', 'take', 'with', 'u', 'all', 'the', 'time', 'in',
'urð\x9f\x93±!!!', 'ð\x9f\x98\x99ð\x9f\x98\x8eð\x9f\x91\x84ð\x9f\x91', 'ð\x9f\x92|ð\x9f\x92|ð\x9f\x92|'],
tags=['tweet_3']),
TaggedDocument(words=['factsguide:', 'society', 'now', '#motivation'], tags=['tweet_4']),
TaggedDocument(words=['[2/2]', 'huge', 'fan', 'fare', 'and', 'big', 'talking', 'before', 'they', 'leave.',
'chaos', 'and', 'pay', 'disputes', 'when', 'they', 'get', 'there.', '#allshowandnogo'], tags=['tweet_5'])]
```



## ✓ Removing unwanted patterns from the data

```
1 import re
2 import nltk
3
4 nltk.download('stopwords')
5 from nltk.corpus import stopwords
6 from nltk.stem.porter import PorterStemmer
7
```

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

## ✓ Train set

```
1 train_corpus = []
2
3 for i in range(0, 31962):
4     review = re.sub('[^a-zA-Z]', ' ', train['tweet'][i])
5     review = review.lower()
6     review = review.split()
7
8     ps = PorterStemmer()
9
10    # stemming
11    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
12
13    # joining them back with space
14    review = ' '.join(review)
15    train_corpus.append(review)
```

```
1 test_corpus = []
2
3 for i in range(0, 17197):
4     review = re.sub('[^a-zA-Z]', ' ', test['tweet'][i])
5     review = review.lower()
6     review = review.split()
7
8     ps = PorterStemmer()
9
10    # stemming
11    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
12
13    # joining them back with space
14    review = ' '.join(review)
15    test_corpus.append(review)
```

## ✓ Creating bag of words

```
1 from sklearn.feature_extraction.text import CountVectorizer
2
3 cv = CountVectorizer(max_features = 2500)
4 x = cv.fit_transform(train_corpus).toarray()
```