Clustering Assignment

Part 1:

Visualization task

The first step I did was to load all the tweets into one Dataframe, with each row corresponding to tweets from a single account. Further, to distinguish tweets for each account, I added the Filename column to the dataframe, which had names of each account corresponding to their tweets. Also, some of the files had a different encoding than 'utf-8'. Hence, while reading the files, I read it using 'latin1' encoding.

```
In [2]: all_tweet_files = glob.glob('E://MSIM//MSIM Spring 2019//MLTP//Health-Tweets' + '//*.txt')

In [3]: tweet_data = pd.DataFrame()
        for file in all_tweet_files:
            data = pd.read_csv(file, sep='|', header = None, names = ('Account','Date','Tweet'), encoding='latin1')
            data['Filename'] = os.path.basename(file)
            tweet_data = tweet_data.append(data)
```

Fig 1a: Loading the Data

1a. Graph the probability of occurrence for the 10 most common words from each Twitter account.

The column of concern to us was of the Tweets. The Tweets consisted of links which was not necessary to us. It also consisted of punctuations and digits. So, As the first step of Data cleaning, I removed all the links, punctuations and digits from the Tweets.

```
tweet_data['Tweet'] = tweet_data['Tweet'].str.replace('(http:).*$','')
tweet_data['Tweet'] = tweet_data['Tweet'].str.replace('["-,<>;.''""?@#%^&_~:!--]','')
tweet_data['Tweet'] = tweet_data['Tweet'].str.replace('[0-9]+','')
```

```
file_tweet = pd.DataFrame(tweet_data.groupby('Filename')['Tweet'].sum().reset_index())
```

Fig 1b: Cleaning the Data

The next step I was to count the top 10 words for each tweet. To do this, I used the 'Counter' from the 'Collections' Package. The Counter function gives output as a tuple, with first element representing the word and the second element is the count of the word in the tweet. I got count of words for tweets for all accounts and made a new column in the dataframe with a list of tuples having the word and the word count. To avoid same words being considered as different due to capitalization, I changed all the words to lowercase.

```
file_tweet['most_common_words'] = file_tweet.Tweet.apply(lambda x: Counter(str(x).lower().split(' ')).most_common(10))
```

```
total_words = []
for i in range(len(file_tweet['Tweet'])):
    total_words.append(len(file_tweet['Tweet'][i]))
file_tweet['total_words'] = total_words
```

```
def most_common_words(dataset):
    for i,file in enumerate(dataset['Filename']):
        tweet_common_words = dataset['most_common_words'][i]
        print(file + ' - ',tweet_common_words)
```

```
most_common_words(file_tweet)
```

Fig 1c: 10 Most Common Words

The word count for each account is as follows:

KaiserHealthNews.txt -
[('to', 1123), ('the', 1042), ('health', 986), ('', 802), ('for', 715), ('in', 662), ('a', 561), ('on', 525), ('of', 522), ('reports', 461)]

NBChealth.txt -
[('to', 813), ('', 770), ('in', 622), ('for', 595), ('of', 471), ('a', 410), ('new', 386), ('the', 369), ('study', 331), ('ebola', 272)]

bbchealth.txt -
[('video', 813), ('to', 610), ('in', 373), ('for', 361), ('nhs', 348), ('ebola', 348), ('the', 255), ('of', 246), ('cancer', 212), ('health', 194)]

cbchealth.txt -
[('to', 1049), ('in', 956), ('', 580), ('for', 561), ('of', 514), ('ebola', 439), ('health', 323), ('the', 323), ('on', 269), ('with', 265)]

cnnhealth.txt -
[('to', 1313), ('', 1309), ('the', 1132), ('a', 930), ('you', 762), ('in', 636), ('for', 622), ('your', 589), ('of', 582), ('rt', 529)]

everydayhealth.txt -
[('to', 1424), ('', 1197), ('a', 1120), ('the', 1048), ('your', 767), ('you', 704), ('for', 619), ('and', 576), ('of', 534), ('in', 418)]

foxnewshealth.txt -
[('', 974), ('to', 589), ('in', 376), ('for', 305), ('of', 295), ('may', 206), ('study', 197), ('with', 165), ('the', 164), ('says', 159)]

gdnhealthcare.txt -
[('the', 1841), ('to', 1302), ('nhs', 928), ('a', 901), ('of', 880), ('in', 830), ('', 718), ('and', 693), ('on', 592), ('for', 591)]

goodhealth.txt -
[('', 3652), ('to', 3481), ('the', 2612), ('a', 2120), ('you', 2088), ('your', 2082), ('for', 1723), ('and', 1715), ('of', 1490), ('with', 1423)]

latimeshealth.txt -
[('to', 1402), ('the', 1277), ('', 1055), ('a', 1029), ('in', 995), ('of', 962), ('for', 722), ('and', 588), ('study', 477), ('is', 456)]

msnhealthnews.txt -
[('', 3488), ('study', 762), ('to', 728), ('may', 652), ('in', 523), ('for', 488), ('of', 423), ('risk', 323), ('cancer', 301), ('kids', 218)]

nprhealth.txt -
[('to', 1245), ('the', 1058), ('a', 970), ('in', 864), ('for', 779), ('of', 720), ('health', 523), ('and', 466), ('ebola', 454), ('on', 420)]

nytimeshealth.txt -
[('the', 1971), ('', 1734), ('to', 1702), ('a', 1574), ('in', 1311), ('of', 1267), ('well', 1267), ('for', 1083), ('rt', 829), ('and', 787)]

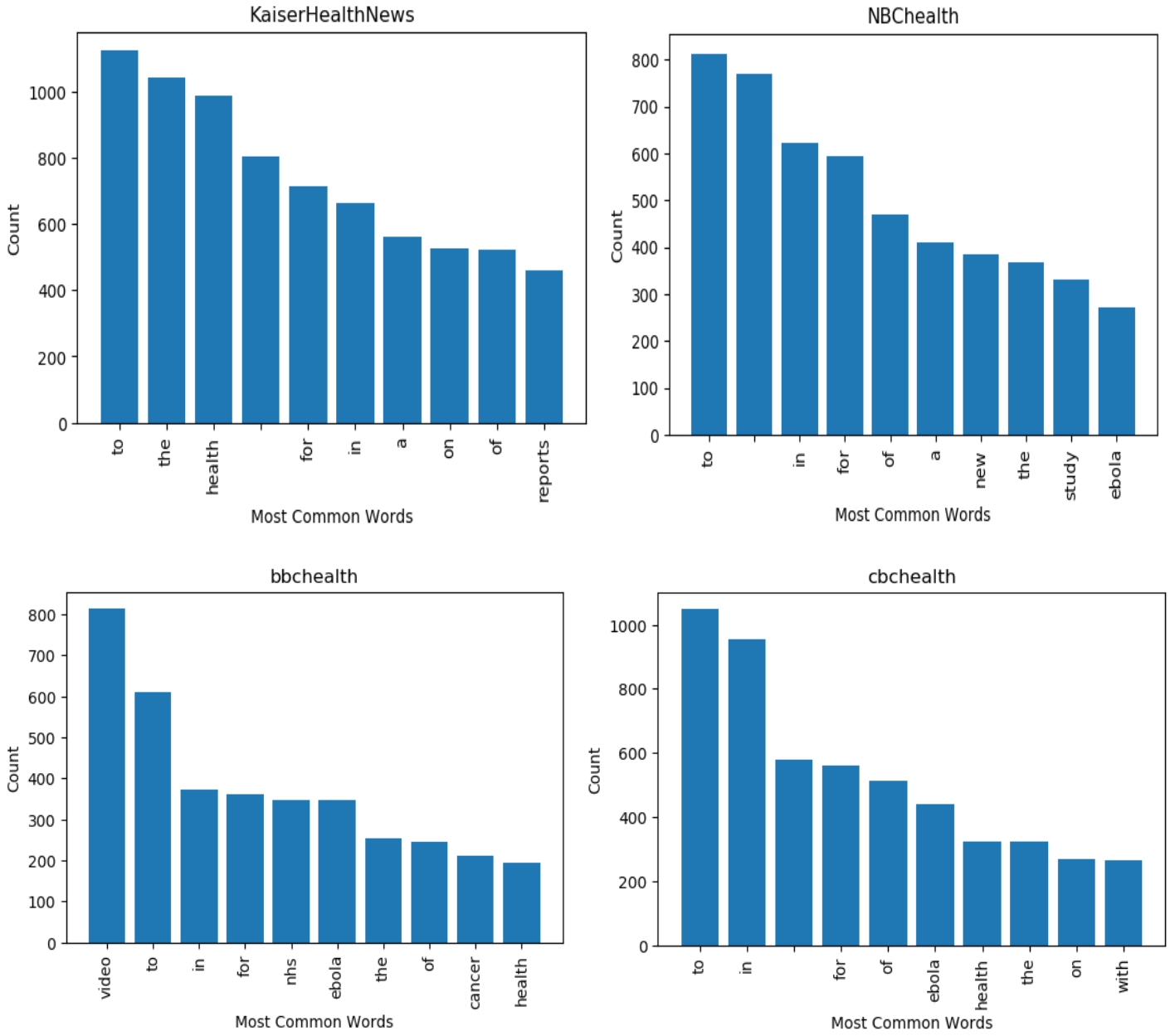
reuters_health.txt -
[('to', 1510), ('ebola', 1337), ('in', 1161), ('for', 802), ('of', 742), ('us', 694), ('', 558), ('drug', 456), ('on', 430), ('may', 348)]

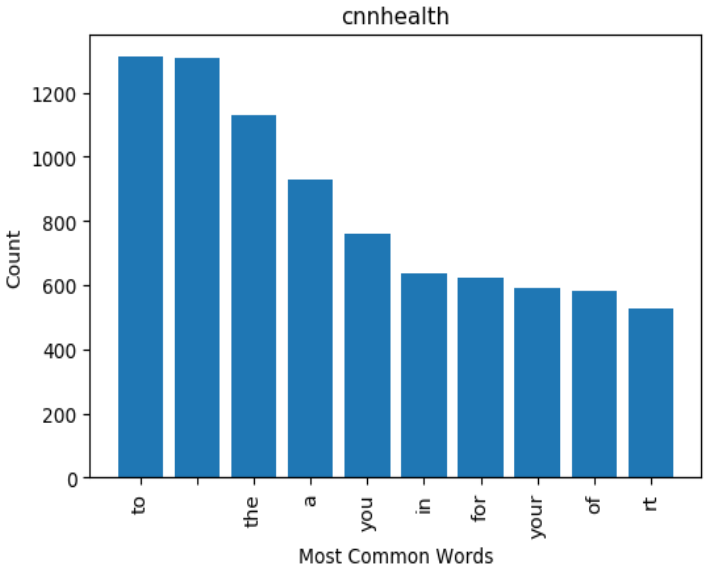
```
usnewshealth.txt -
[('to', 580), ('', 522), ('the', 505), ('a', 428), ('you', 391), ('your', 323), ('fo
r', 252), ('rt', 224), ('of', 215), ('how', 205)]

wsjhealth.txt -
[('rt', 2078), ('to', 1098), ('', 1059), ('the', 962), ('of', 766), ('in', 738), ('s
tefaniei', 725), ('for', 612), ('on', 516), ('a', 513)
```

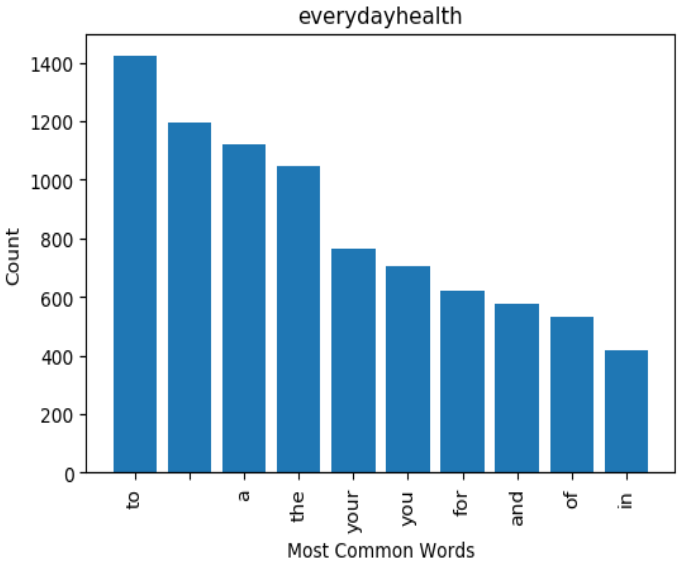
```python
def plot_word_count():
    for i in range(len(file_tweet)):
        plt.figure(dpi = 100)
        plt.bar(*zip(*file_tweet['most_common_words'][i]))
        plt.xlabel('Most Common Words')
        plt.xticks(rotation = 90)
        plt.ylabel('Count')
        plt.title(file_tweet['Filename'][i].strip('.txt'))
plot_word_count()
```

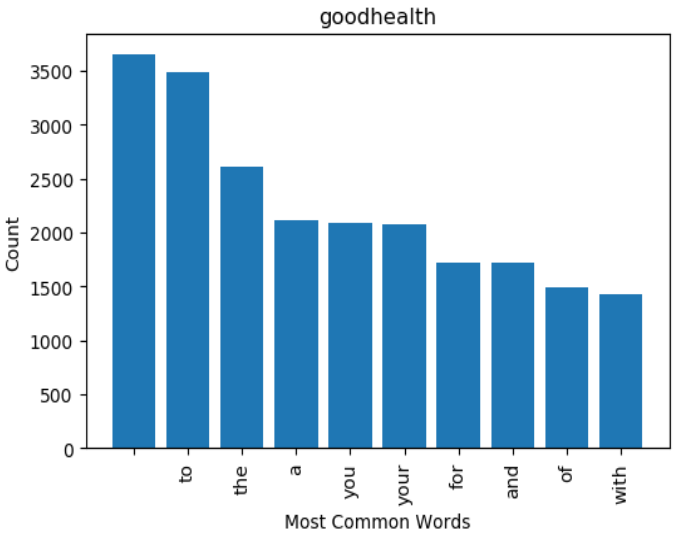
Fig 1d: Plotting 10 Most Common Words

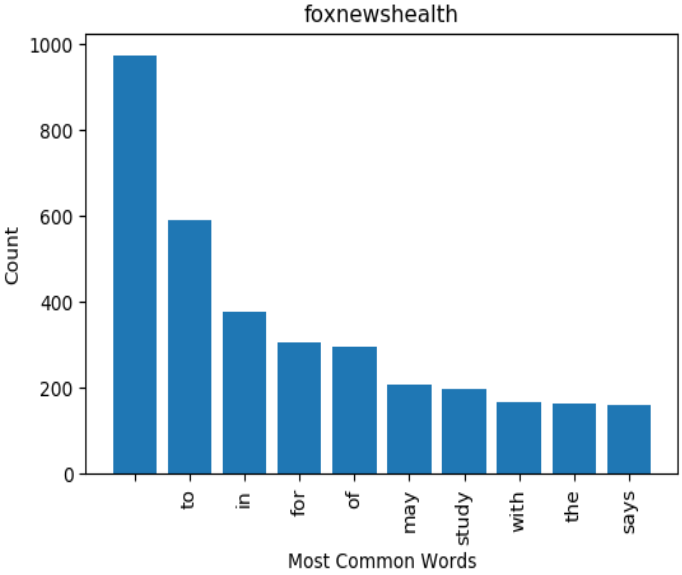
Based on the word count, below is the graphical visualization of the words before complete Cleaning:

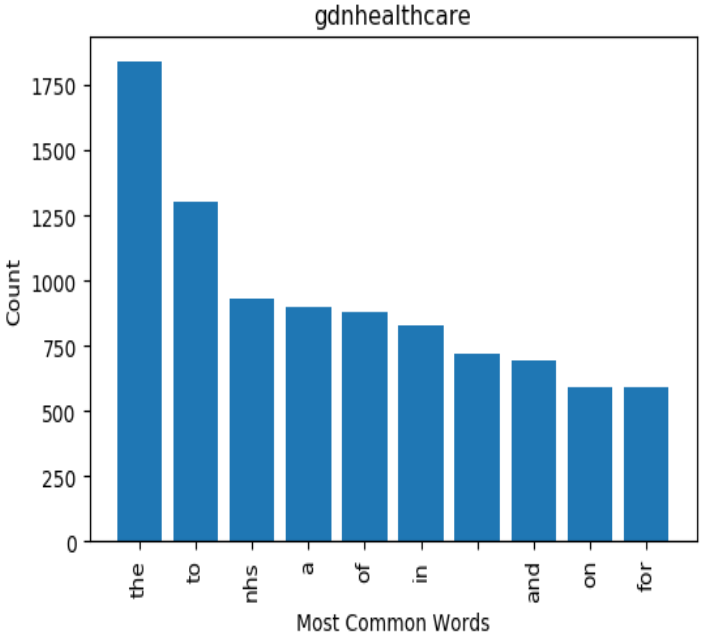


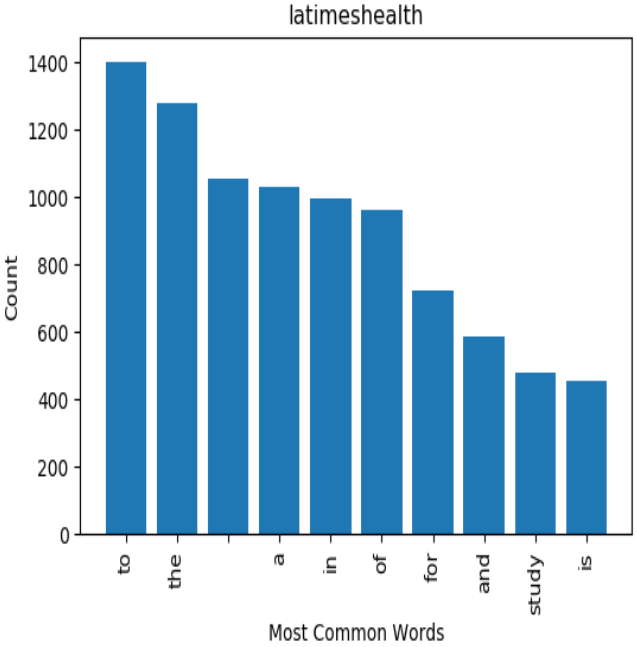

cnnhealth
Count
Most Common Words
to
the
a
you
in
for
your
of
rt


everydayhealth
Count
Most Common Words
to
a
the
your
you
for
and
of
in


goodhealth
Count
Most Common Words
to
the
a
you
your
for
and
of
with


foxnewshealth
Count
Most Common Words
to
in
for
of
may
study
with
the
says


gdnhealthcare
Count
Most Common Words
the
to
nhs
a
of
in
and
on
for


latimeshealth
Count
Most Common Words
to
the
a
in
of
for
and
study
is

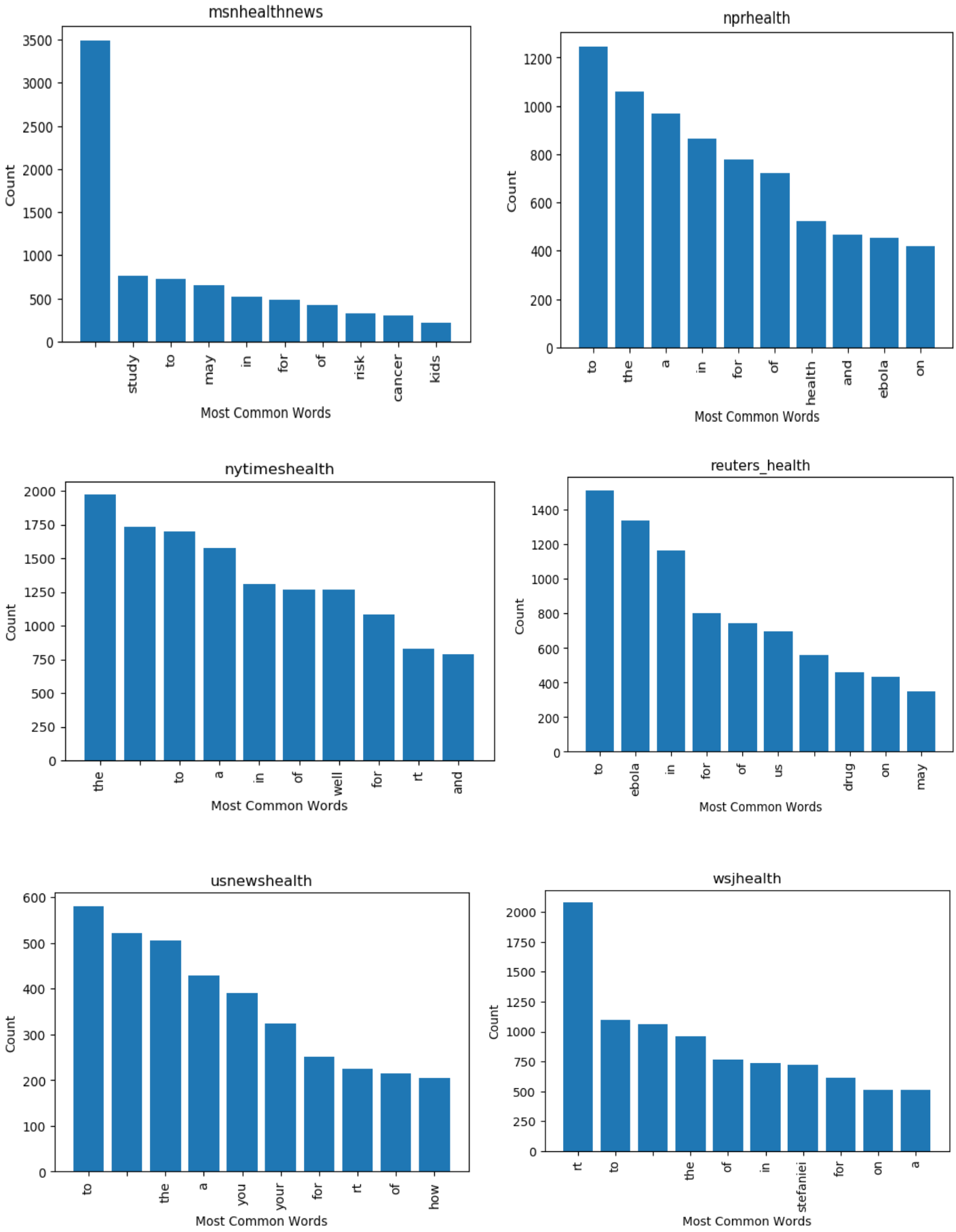


Fig 1e: Word Count without Data Cleaning

1b. Are these most probable words related to health?

From the word count and visualization, it is seen that the top 10 words were not health related. Although there were a couple of words related to health like 'Ebola', most of the words were single character, articles and blank spaces.Also, it contained a lot of words like 'to, a, an, the, for, your, of' etc. These words are called 'stopwords'. These are irrelevant to our analysis and hence they need to be filtered out. Further, there were words like 'Video, Audio, one/two-character words' which had no relation to the health.

1c. If not, can you propose a way to improve the results?

The first visualization contained a lot of stopwords. So, I used the 'Stopwords' feature from NLTK to remove all the stop words from the data. Also, words are 'rt, video, audio' had no relation to health. Hence, these words were added to the stopwords list too. The blank spaces in the data, as represented in above figures, were also filtered out by adding them to punctuations regex pattern. After cleaning, below is the word list and the frequency/probability of the words in the tweets.

```
stop = stopwords.words('english')
more_words = ['video','audio','rt','us','new','says','amp','q']
stop.extend(more_words)
file_tweet['tweet_without_stopwords'] = file_tweet['Tweet'].apply([lambda x: ' '.join(word for word in word_tokenize(x.lower()) i
file_tweet['most_common_words'] = file_tweet.tweet_without_stopwords.apply(lambda x: Counter(str(x).lower().split(' ')).most_comm
most_common_words(file_tweet)
```

Fig 1f: Removing the Stopwords and irrelevant Words

As from the output, we can see that there are higher number of health-related words as compared to previous output before removing stopwords. The words like ebola, health, medicare, Medicaid, obamacare etc. are representative of health.

```
KaiserHealthNews.txt -
[('health', 986), ('reports', 461), ('insurance', 414), ('todays', 348), ('law', 327
), ('obamacare', 314), ('care', 283), ('medicaid', 236), ('medicare', 224), ('cartoo
n', 216)]

NBChealth.txt -
[('study', 331), ('ebola', 272), ('fda', 238), ('health', 232), ('finds', 225), ('ma
y', 213), ('cancer', 181), ('kids', 142), ('flu', 137), ('obamacare', 116)]

bbchealth.txt -
[('nhs', 348), ('ebola', 348), ('cancer', 212), ('health', 194), ('care', 182), ('ho
spital', 135), ('uk', 102), ('mental', 90), ('risk', 89), ('patients', 85)]

cbchealth.txt -
[('ebola', 439), ('health', 323), ('canada', 158), ('cancer', 151), ('outbreak', 148
), ('study', 137), ('medical', 122), ('may', 121), ('hospital', 108), ('doctors', 10
7)]

cnnhealth.txt -
[('health', 238), ('may', 194), ('getfit', 175), ('cnnhealth', 159), ('tip', 156), (
'todays', 152), ('cancer', 148), ('kids', 140), ('cnn', 128), ('know', 128)]

everydayhealth.txt -
[('healthtalkrt', 363), ('foods', 252), ('healthtalk', 239), ('health', 206), ('ever
ydayhealth', 189), ('weight', 162), ('healthy', 145), ('may', 145), ('ways', 142), (
'get', 139)]
```

```
foxnewshealth.txt -
[('may', 206), ('study', 197), ('ebola', 152), ('cancer', 137), ('finds', 86), ('hea
lth', 78), ('brain', 68), ('heart', 67), ('risk', 65), ('disease', 62)]

gdnhealthcare.txt -
[('nhs', 928), ('gdnhealthcare', 391), ('health', 299), ('care', 260), ('healthcare'
, 204), ('patients', 190), ('today', 173), ('miss', 161), ('dont', 158), ('aampe', 1
43)]
goodhealth.txt -
[('goodhealth', 827), ('cynthiasass', 801), ('q', 735), ('get', 582), ('try', 502),
('healthy', 495), ('make', 422), ('ways', 416), ('day', 381), ('recipes', 381)]

latimeshealth.txt -
[('study', 477), ('may', 216), ('health', 197), ('get', 170), ('cancer', 162), ('peo
ple', 151), ('via', 147), ('kids', 145), ('risk', 139), ('help', 138)]

msnhealthnews.txt -
[('study', 762), ('may', 652), ('risk', 323), ('cancer', 301), ('kids', 218), ('hear
t', 201), ('help', 168), ('might', 158), ('linked', 152), ('patients', 139)]

nprhealth.txt -
[('health', 523), ('ebola', 454), ('may', 221), ('care', 196), ('insurance', 173), (
'doctors', 153), ('help', 152), ('kids', 131), ('could', 130), ('obamacare', 125)]

nytimeshealth.txt -
[('well', 1267), ('health', 660), ('ebola', 629), ('age', 277), ('may', 276), ('canc
er', 275), ('old', 263), ('care', 239), ('blog', 230), ('study', 185)]

reuters_health.txt -
[('ebola', 1337), ('drug', 456), ('may', 348), ('study', 288), ('health', 273), ('ca
ncer', 223), ('fda', 193), ('risk', 144), ('hospital', 143), ('africa', 141)]

usnewshealth.txt -
[('diet', 86), ('heart', 75), ('know', 71), ('health', 68), ('dont', 60), ('whats',
58), ('fitness', 57), ('get', 56), ('healthy', 56), ('one', 54)]

wsjhealth.txt -
[('stefaniei', 725), ('health', 407), ('pharmalot', 359), ('ebola', 319), ('drug', 2
26), ('wsj', 215), ('lauralandrowsj', 176), ('fda', 156), ('law', 147), ('tomburtonw
sj', 134)]
```

After getting the word count, I calculated the frequency (Weight) of each word in the Tweets for an account, by taking a division of the count of that word and total words in the account after preprocessing.

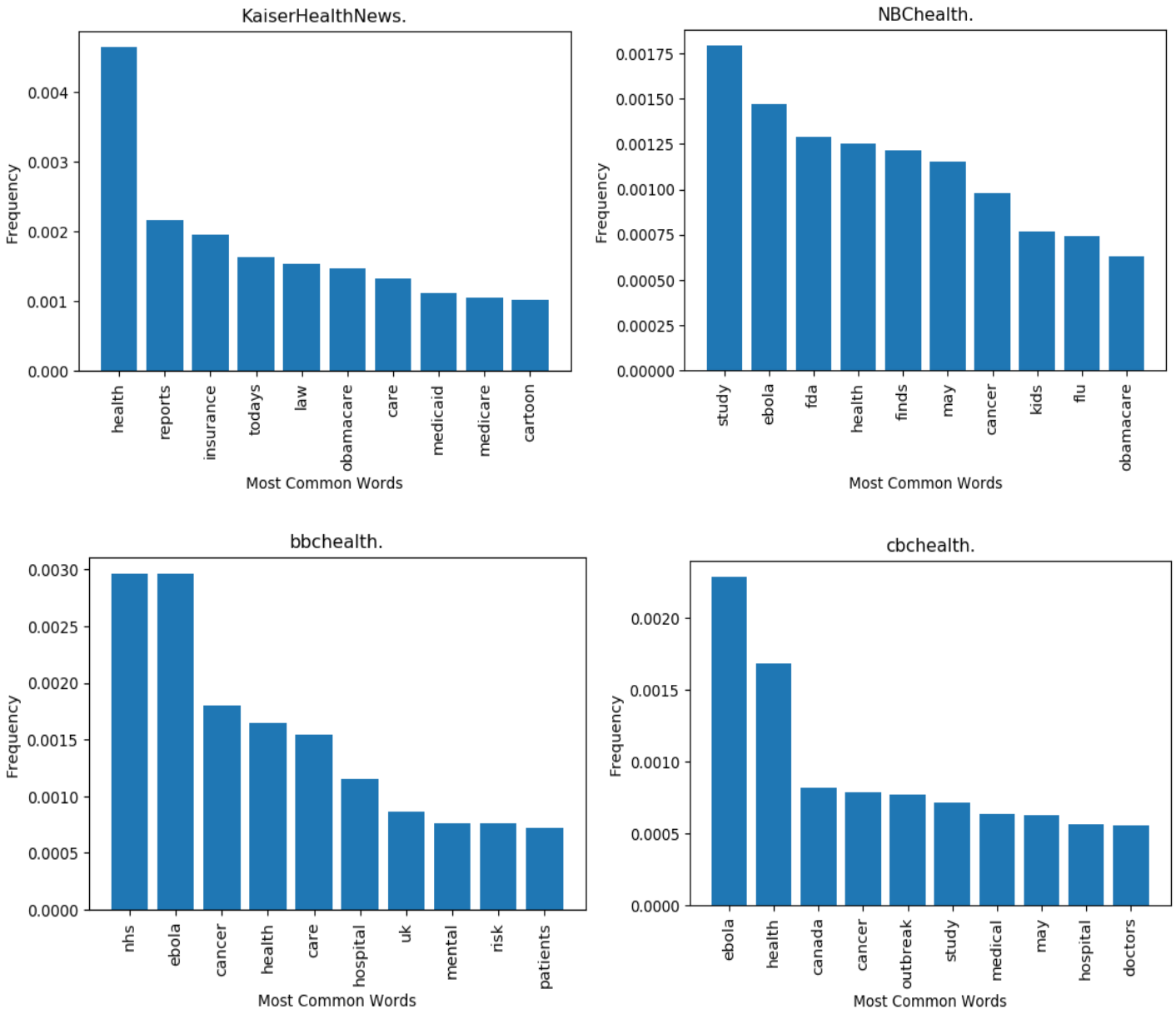
```
total_words = []
for i in range(len(file_tweet['tweet_without_stopwords'])):
    total_words.append(len(file_tweet['tweet_without_stopwords'][i]))
file_tweet['total_words'] = total_words
```

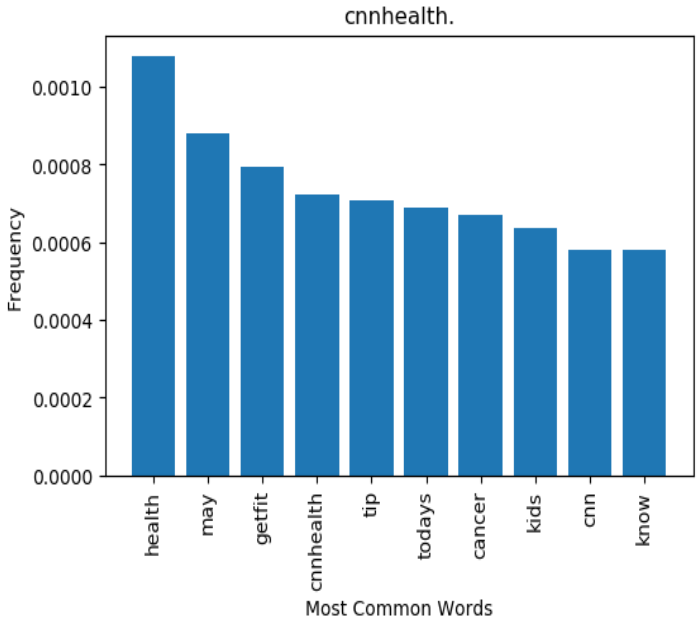
Fig 1g: Calculating Total Words in each Tweet Account

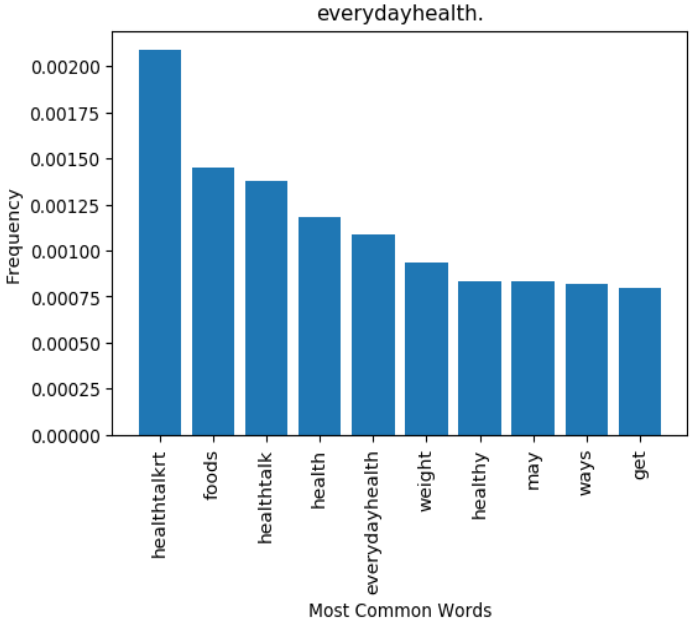
```
def plot_frequency_words():
    dataframes = {}
    for filename, words in zip(file_tweet['Filename'],file_tweet['most_common_words']):
        dataframes[filename] = pd.DataFrame(words)

    for i,frame in enumerate(dataframes):
        dataframes[frame]['frequency'] =     dataframes[frame][1]/file_tweet['total_words'][i]
        dataframes[frame].columns = ['word','word_count','frequency']

    for frame in dataframes:
        plt.figure(dpi = 100)
        plt.bar(x = dataframes[frame]['word'], height = dataframes[frame]['frequency'])
        plt.xlabel('Most Common Words')
        plt.xticks(rotation = 90)
        plt.ylabel('Frequency')
        plt.title(frame.strip('txt'))
```

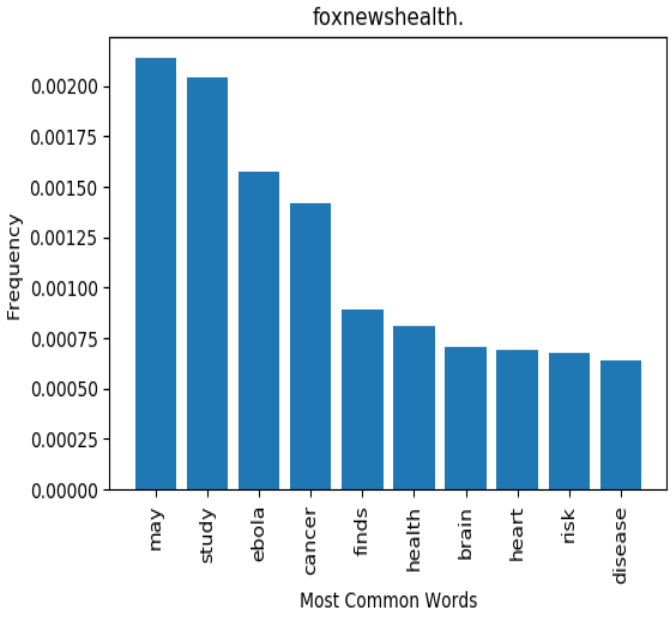
Fig 1h: Plotting Frequency of 10 Most Common Words

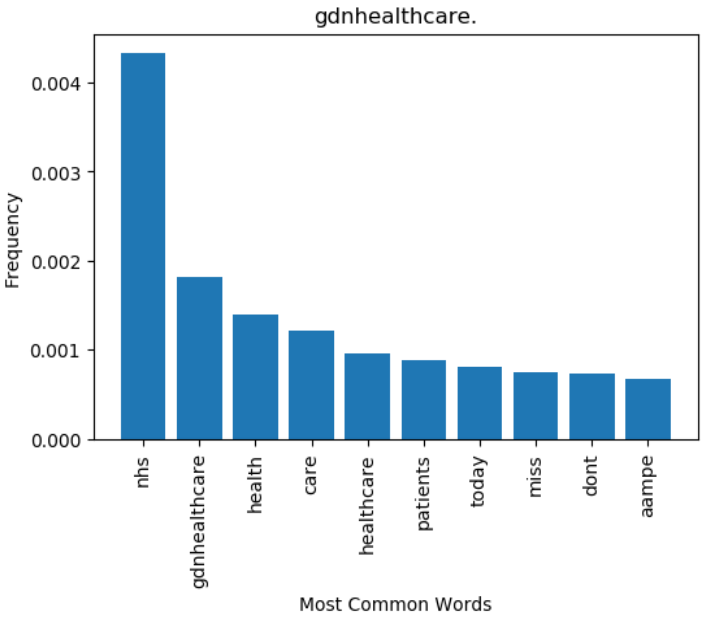
Below graph shows the frequency (weight/Probability) of top 10 words for each account.

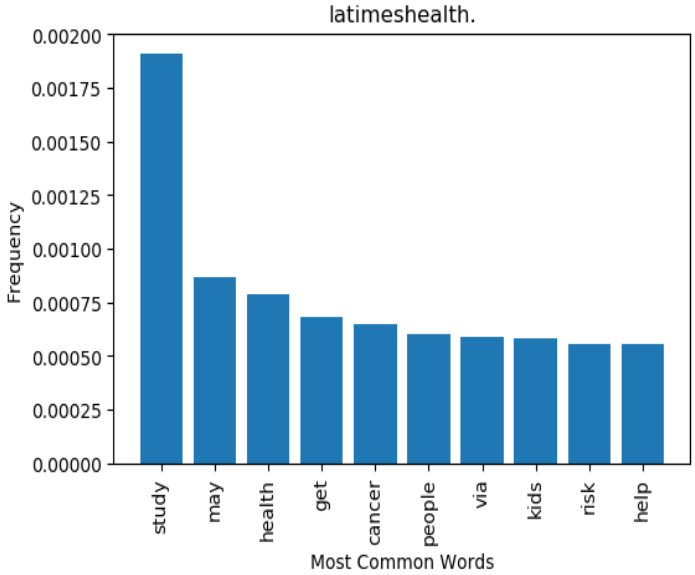


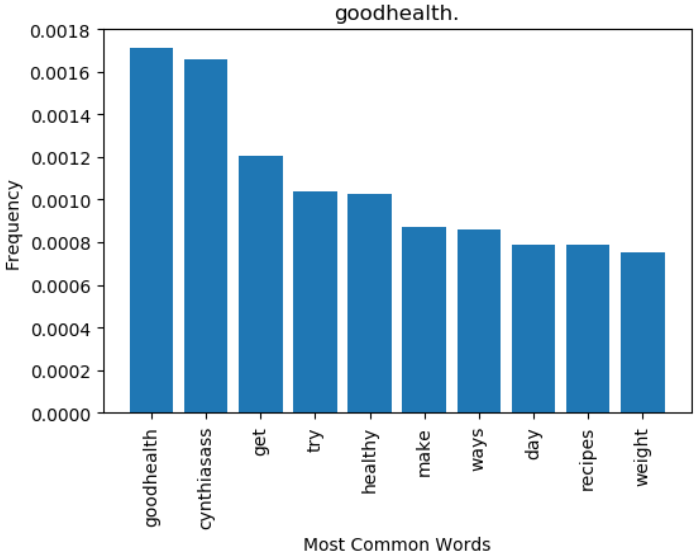

cnnhealth.
Frequency
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
health
may
getfit
cnnhealth
tip
todays
cancer
kids
cnn
know
Most Common Words


everydayhealth.
Frequency
0.00200
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
healthtalkrt
foods
healthtalk
health
everydayhealth
weight
healthy
may
ways
get
Most Common Words


foxnewshealth.
Frequency
0.00200
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
may
study
ebola
cancer
finds
health
brain
heart
risk
disease
Most Common Words


gdnhealthcare.
Frequency
0.004
0.003
0.002
0.001
0.000
nhs
gdnhealthcare
health
care
healthcare
patients
today
miss
dont
aampe
Most Common Words


latimeshealth.
Frequency
0.00200
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
study
may
health
get
cancer
people
via
kids
risk
help
Most Common Words


goodhealth.
Frequency
0.0018
0.0016
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
goodhealth
cynthiasass
get
try
healthy
make
ways
day
recipes
weight
Most Common Words

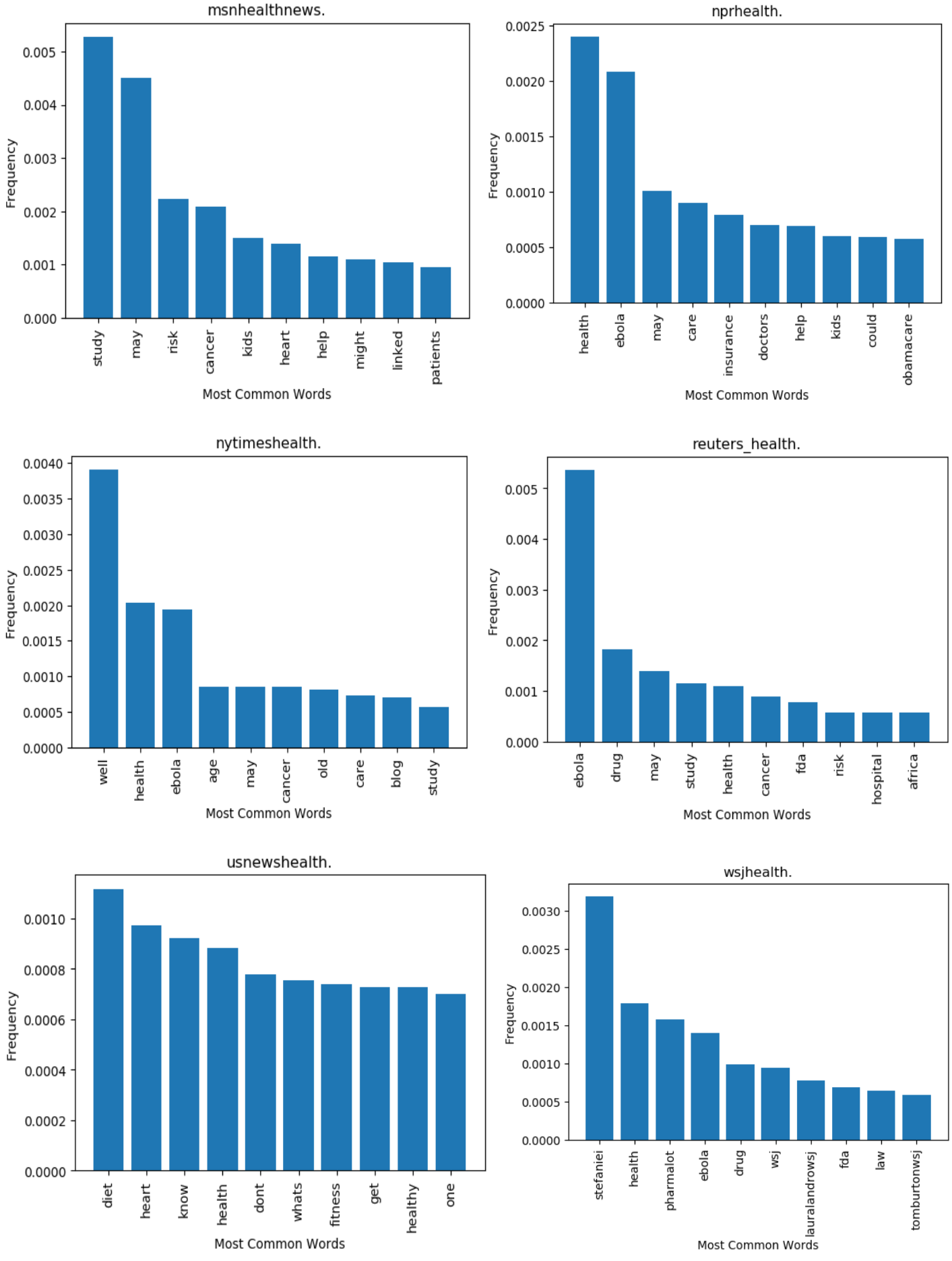


Fig 1i: Frequency of Words after Data Cleaning

Part 2:

Clustering task

2a. Combine tweets from all 16 accounts and cluster the tweets with K-means clustering, using K = 16. Graph these clusters in two dimensions. You may choose to use principal components analysis (PCA) or another method of your choice to choose two dimensions for visualization.

For this step, I combined all the Tweets from the 'Tweet_data' Dataframe, column 'Tweet' together. I decided the **TF-IDF (Term Frequency – Inverse Document Frequency)** vectorizer to convert the words in vectors for Clustering and dimensionality reduction. The TFIDF vector takes a raw file, document or an iterative input. Hence, I converted all the tweets from the Tweet column to a list, with each element in the list being a tweet. The TFIDF converts the tweet into a Sparse matrix, with each element in the sparse matrix being the vector/ frequency of the words in the tweet. It basically shows how important each word/feature is the given documents.

```
v = TfidfVectorizer(lowercase=True)
x = v.fit_transform(tweet_data['Tweet'].tolist())
```

Fig 2a: TFIDF Vectorizer

The output of TFIDF vectorizer looks like shown below:

```
  (0, 3622)    0.37555323720462724
  (0, 4330)    0.2821631357184048
  (0, 25826)   0.3057084897949824
  (0, 30359)   0.37592273891245265
  (0, 7972)    0.7379486055969705
  (1, 12672)   0.3480544542763573
  (1, 33673)   0.4856885799671625
  (1, 13345)   0.4519283760986408
  (1, 4466)    0.22708555700668737
  (1, 3245)    0.5085804257212821
  (1, 23115)   0.3584739089485256
  (2, 25826)   0.30073103262239487
```

As the output from TFIDF vectorizer is a sparse matrix, we cannot use PCA for dimensionality reduction. I deci ded to use Truncated SVD for dimensionality reduction, which reduces the dimensions to 2. Dimensionality re duction is done so that we can visualize the tweets in two dimensions, as the tweets were in thousands and it is impossible to visualize those after clustering. Also, clustering such large dimension consumes a lot of time a nd resources. The dimensionality reduction identifies the two components which explain the maximum varian ce in the data and is a good representation of the overall data. Below is the output after reducing data to two dimensions

```
svd = TruncatedSVD(n_components=2)
reduced_data = svd.fit_transform(x)
```

Fig 2b: Truncated SVD

```
array([[ 0.04808832, -0.0456296 ],
       [ 0.01681244, -0.01482756],
       [ 0.03243663, -0.02172279],
       ...,
       [ 0.11345252, -0.0216742 ],
       [ 0.00850676, -0.00755777],
       [ 0.11143511, -0.1026843 ]])
```

After this, I moved onto K-means clustering. The number of clusters were 16. After fitting the reduced data, I predicted the values using 'predict' function and then plotted the prediction using matplotlib. The x-axis represents first component of reduction and Y axis represents the second one.

```
kmeans = KMeans(n_clusters=16)
model_kmeans = kmeans.fit(reduced_data)
model_predict = kmeans.predict(reduced_data)
```

```
plt.figure(dpi = 100)
plt.scatter(reduced_data[:,0], reduced_data[:,1], c= model_predict)
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.title('K-Means Clustering for 16 Clusters')
```

Fig 2c: K-means Clustering

Below is a visualization of the clustering:

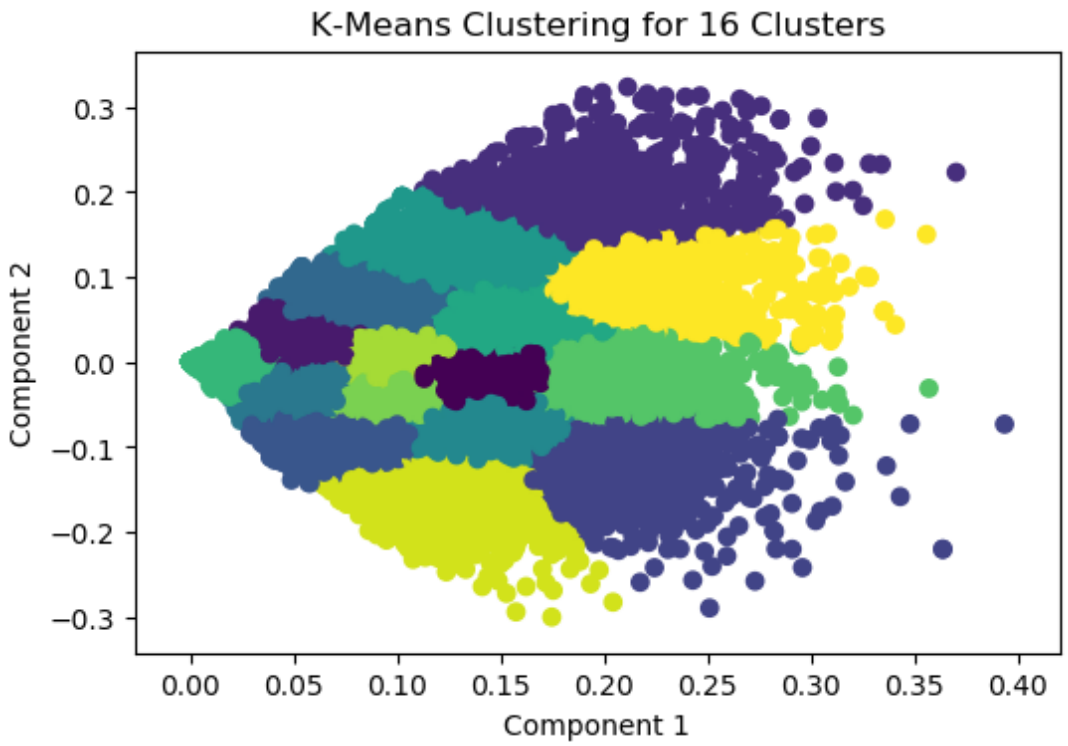


Fig 2d: K-means Clustering with 16 Clusters

2b. Does each Twitter account form its own cluster?

Each color in the above graph represents a cluster. The visualization shows that the data points are clustered nicely together. But, it doesn't form distinct clusters based on the user account. K-means clustering on TFIDF vectorizer reduced data does not cluster based on Twitter accounts.

2c. Why or why not is this the case?

K-means clustering groups words with similar features closely together and it does not know which file/account the words belong to. They cluster it by how closely the words relate to each other. Words with high similarity are put together in one cluster and with no similarity are placed far away from each other. Hence, we can conclude that the K-means clusters the data based on similarity in the content of the TFIDF vectorizer output. For example, words like Medicare, Medicaid, Obamacare etc. will be clustered together as they represent same type of content (medical cover). Same can be said for words like doctor, patient and clinic, as all represent the medical sector.

Part 3:
Improving the clustering

3a. Try to improve the results by either changing the value of K, or by using an entirely different clustering algorithm (e.g., DBSCAN). Describe what you did.

One of the ways to improve clustering is to decide the optimal value of K (number of clusters). There are mainly three different ways in which we can find out the optimal value of the K namely, **Elbow Method, Average Silhouette Method and Gap Statistic method.** I decided to use the Elbow method to decide the clusters for improving the clustering. The way in which elbow methods decided optimal clusters is as follows:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, in our case the number of clusters is 16, so vary k from 1 to 16 clusters.
2. For each k, the method calculates the total within-cluster sum of square (WSS). It may use different types of distances like Manhattan, Euclidean etc. I decided to use Euclidean distance as it is popularly and generally used for calculation distance between vectors.
3. Plot the curve of WSS according to the number of clusters k. The Y-axis here represent the distortions (WSS distance within cluster). The x-axis is the Cluster number.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

```
X = reduced_data
distorsions = []
for k in range(1, 17):
    kmeans = KMeans(n_clusters=k).fit(X)
    kmeans.fit(X)
    distorsions.append(np.average(np.min(cdist(X, kmeans.cluster_centers_, 'euclidean'), axis=1)))
```

```
fig = plt.figure(dpi=100)
plt.plot(range(1, 17), distorsions)
plt.grid(True)
plt.title('Elbow curve')
plt.xlabel('Number of Clusters')
plt.ylabel('Distortion')
```

Fig 3a: Elbow Curve Method

Below is the Elbow graph for 16 clusters:

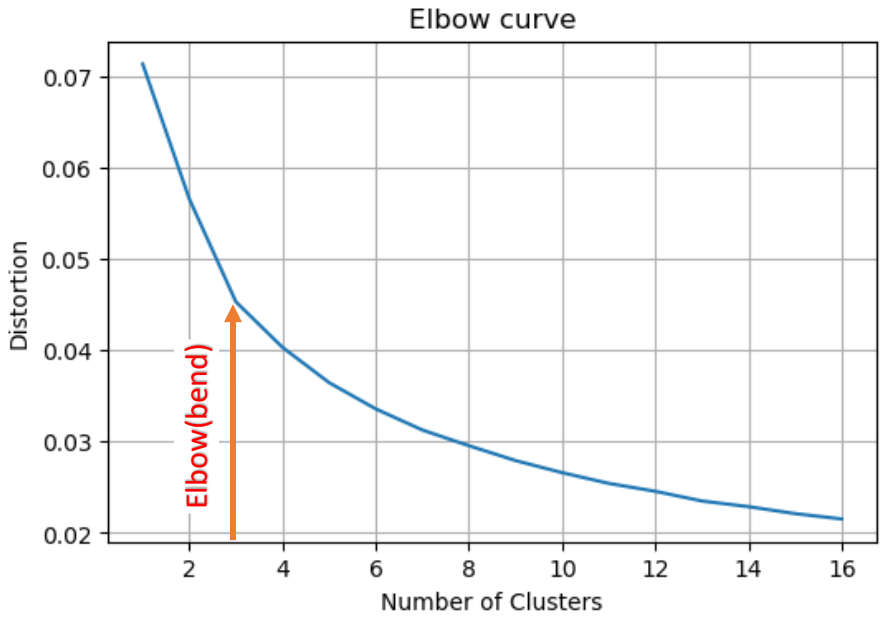


Fig 3b: Elbow Curve

From the graph, the elbow or bend in the curve is clearly visible at Cluster 3. So, I decided to use 3 clusters as the optimal number of clusters.

3b. Do you now see some patterns in the clusters? That is, do the clusters mean something?

After deciding the cluster, I fitted the data to K-means again and visualized the results again.

```
kmeans = KMeans(n_clusters=3)
model_kmeans = kmeans.fit(reduced_data)
model_predict = kmeans.predict(reduced_data)
```

```
plt.figure(dpi = 100)
plt.scatter(reduced_data[:,0], reduced_data[:,1], c= model_predict)
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.title('K-Means Clustering for 3 Clusters')
```

Fig 3c: K-means for 3 Clusters

Below is the visualization for 3 clusters:

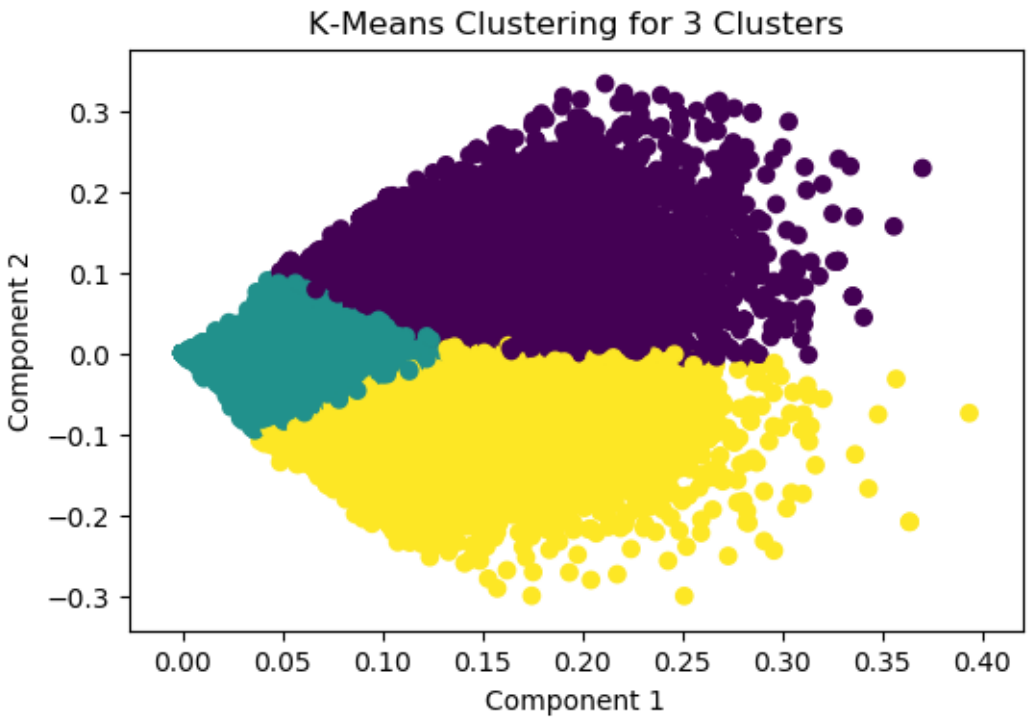


Fig 3d: K-means Clustering with 3 Clusters

The 3 clusters may be representative that the tweets are mostly about 3 most common health related terms and each cluster represents words related to a health-related term. It could represent that the tweets from similar countries are grouped together and as the tweets are from 3 countries majorly, each cluster could represent tweets from a single country. Also, there maybe be no pattern in the clustering, and the K-means algorithm might have just combined the clusters with close relation into one cluster and reduced the clusters to 3.