Topic Modeling of Coronavirus Tweets

Capstone Project 2

Alex Chung

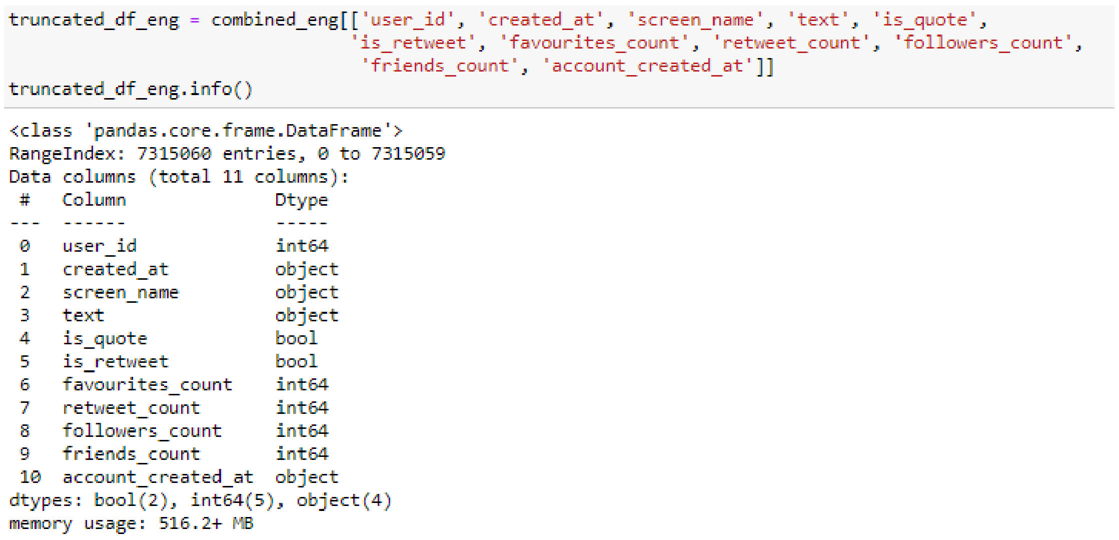
Capstone 2 Project Final Report: NLP of Worldwide Lockdown Tweets

We started to **explore twitter feeds scraped** from March 30th to April 30th, 2020 to see if we can gain some insight into the effects of the worldwide self-seclusion because of the Covid19 pandemic. We did initial exploratory data analysis leading us to look at the tweets located in the "text" column. As we saw in our first milestone report we have about 4 gigs of data in 30 csv files we combined into a single dataframe with 7.16 million observations.

Our workflow:

- **Prep** our data thru exploratory data analysis and data wrangling
- **Clean**, lemmatize, tokenize our text
- **Vectorize** our processed text to start the natural learning process using scikit-learn and spaCy.
- Use KMeans Clustering to **discover** topical **clusters**
- Finally, we will **extract** important topical **keywords** using Latent Dirichlet Allocation

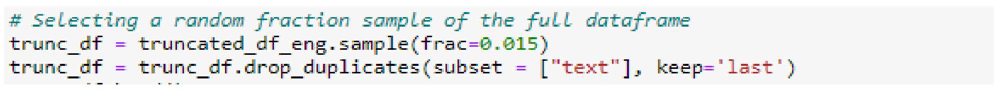
Reading in and prepping our Data:



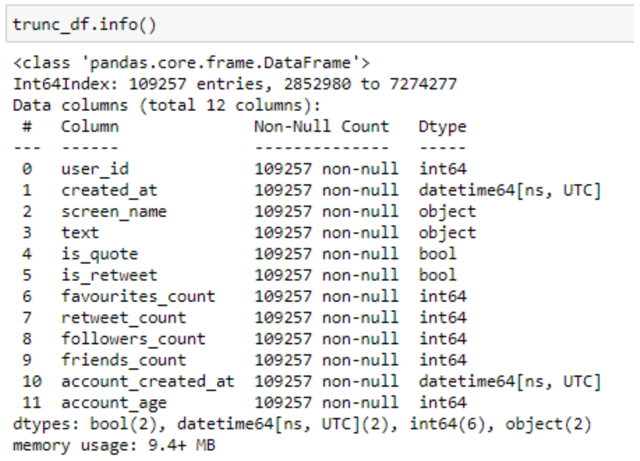
```
truncated_df_eng = combined_eng[['user_id', 'created_at', 'screen_name', 'text', 'is_quote',
                                 'is_retweet', 'favourites_count', 'retweet_count', 'followers_count',
                                 'friends_count', 'account_created_at']]
truncated_df_eng.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7315060 entries, 0 to 7315059
Data columns (total 11 columns):
 #   Column              Dtype
---  ------              -----
 0   user_id             int64
 1   created_at          object
 2   screen_name         object
 3   text                object
 4   is_quote            bool
 5   is_retweet          bool
 6   favourites_count    int64
 7   retweet_count       int64
 8   followers_count     int64
 9   friends_count       int64
 10  account_created_at  object
dtypes: bool(2), int64(5), object(4)
memory usage: 516.2+ MB
```

We have about 7.3 million rows of data. We reduce our data to about 100,000 rows by picking a random sample in order to work with our available resources.



```
# Selecting a random fraction sample of the full dataframe
trunc_df = truncated_df_eng.sample(frac=0.015)
trunc_df = trunc_df.drop_duplicates(subset = ["text"], keep='last')
```



```
trunc_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 109257 entries, 2852980 to 7274277
Data columns (total 12 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   user_id             109257 non-null  int64
 1   created_at          109257 non-null  datetime64[ns, UTC]
 2   screen_name         109257 non-null  object
 3   text                109257 non-null  object
 4   is_quote            109257 non-null  bool
 5   is_retweet          109257 non-null  bool
 6   favourites_count    109257 non-null  int64
 7   retweet_count       109257 non-null  int64
 8   followers_count     109257 non-null  int64
 9   friends_count       109257 non-null  int64
 10  account_created_at  109257 non-null  datetime64[ns, UTC]
 11  account_age         109257 non-null  int64
dtypes: bool(2), datetime64[ns, UTC](2), int64(6), object(2)
memory usage: 9.4+ MB
```

Clean, lemmatize, tokenize our text:

Our text data is contained in the “text” column. Cleaning up the text data is necessary to highlight attributes that we want our machine learning system to pick up on. We will use spaCy to process our texts in the following sequence:

1. Break our texts into tokens
2. Remove capitals and lemmatize
3. Remove stopwords and punctuations

```
# Importing spaCy natural learning processing library
import spacy
import en_core_web_sm
```

```
# Tokenizing our text to start
from spacy.lang.en.stop_words import STOP_WORDS
punctuations = string.punctuation
stopwords = list(STOP_WORDS)

parser = en_core_web_sm.load(disable=["tagger", "ner"])
parser.max_length = 500000

def spacy_tokenizer(sentence):
    mytokens = parser(sentence)
    mytokens = [ word.lemma_.lower().strip() if word.lemma_ != "-PRON-" else word.lower_ for word in mytokens ]
    mytokens = [ word for word in mytokens if word not in stopwords and word not in punctuations ]
    mytokens = " ".join([i for i in mytokens])
    return mytokens
```

```
trunc_df["processed_text"] = trunc_df["text"].progress_apply(spacy_tokenizer)
```

Vectorize our processed text

Once our data has been cleaned and tokenized we will scikit-learn to vectorize our data for machine learning. We use TfidfVectorizer to get the Tf-idf or the *term frequency inverse document frequency* weight for each token represented by:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

This weight is a statistical importance of a word to a document in our data.

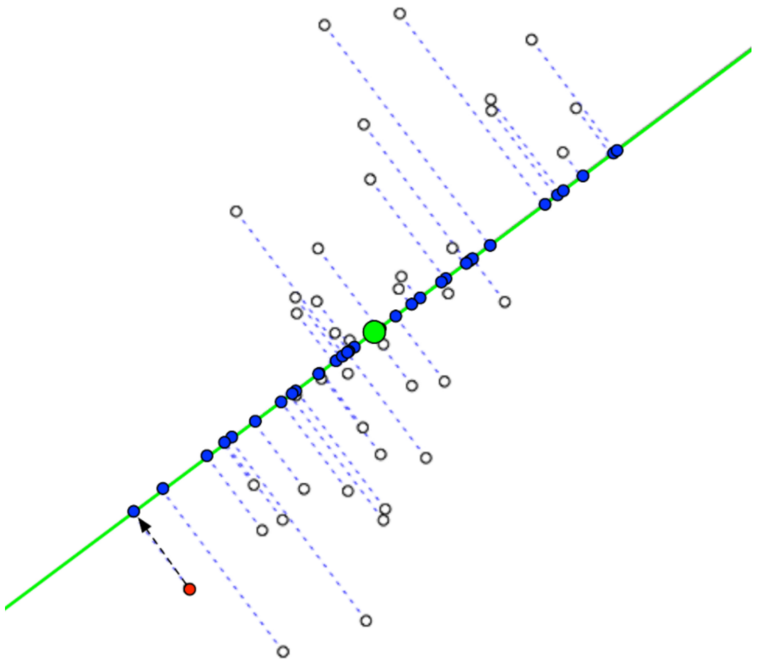
```
# Importing scikit-learn modules
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# Vectorizing text with scikit-learn TfidfVectorizer
def vectorize(text, features):

    vectorizer = TfidfVectorizer(max_features=features)
    X = vectorizer.fit_transform(text)
    return X
```

```
text = trunc_df['processed_text'].values
X = vectorize(text, 2 ** 11)
X.shape
```

```
(109257, 2048)
```

Once vectorized, our data takes the dimensions of 109,257 x 2,048. We next use scikit-learn to run PCA or principal component analysis to reduce the dimensions of our data. PCA calculates the most important variables and drops the least important variables by finding the direction of the line that maximizes the variance (the mean of squared distances from the projected points to the origin).



PCA reduces the dimensions down to 109,257 x 1,735.

```
from sklearn.decomposition import PCA

pca = PCA(n_components=0.95, random_state=42)
X_reduced= pca.fit_transform(X.toarray())
X_reduced.shape
```

```
(109257, 1735)
```

Use KMeans Clustering to discover topical clusters

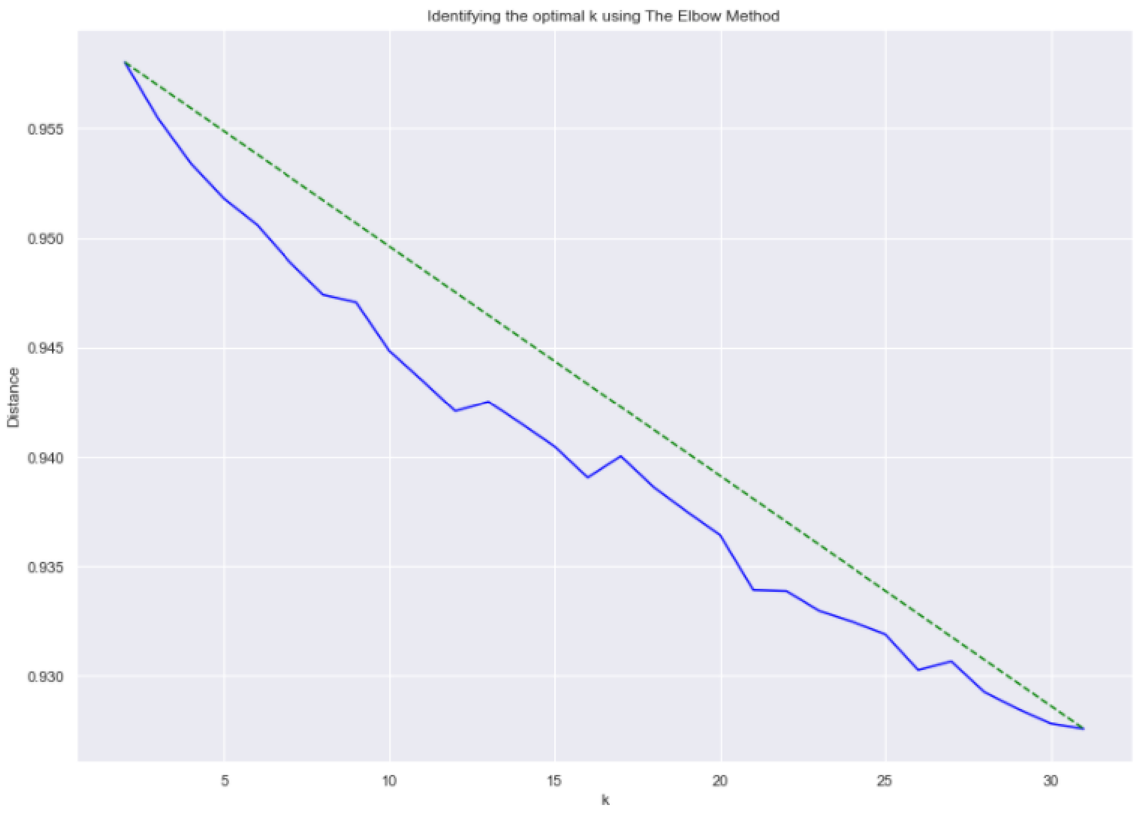
Since our data isn't labeled we use kmeans as an unsupervised learning method to discover clusters to discern topics. Kmeans uses an iterative algorithm to locate centroids among k number clusters.

```
from sklearn.cluster import KMeans
```

One way to find out the optimal number for k clusters is to use the elbow method. We iterate through different k clusters the distance between the sum of squares to its centroid for each k. The distance decreases at the bend of the elbow - this highlights the optimal k value.

```
from sklearn import metrics
from scipy.spatial.distance import cdist

# run kmeans with many different k
distance = []
K = range(2, 32)
for k in K:
    k_means = KMeans(n_clusters=k, random_state=42)
    k_means.fit(X_reduced)
    distance.append(sum(np.min(cdist(X_reduced, k_means.cluster_centers_, 'euclidean'), axis=1)) / X.shape[0])
```

```
X_line = [K[0], K[-1]]
Y_line = [distance[0], distance[-1]]

plt.plot(K, distance, color='blue')
plt.plot(X_line, Y_line, color='green', linestyle='dashed')
plt.xlabel('k')
plt.ylabel('Distance')
plt.title('Identifying the optimal k using The Elbow Method')
plt.show()
```



We see from the plot above the bend of the elbow starts to diminish at between k = 12. This is a good place to start.

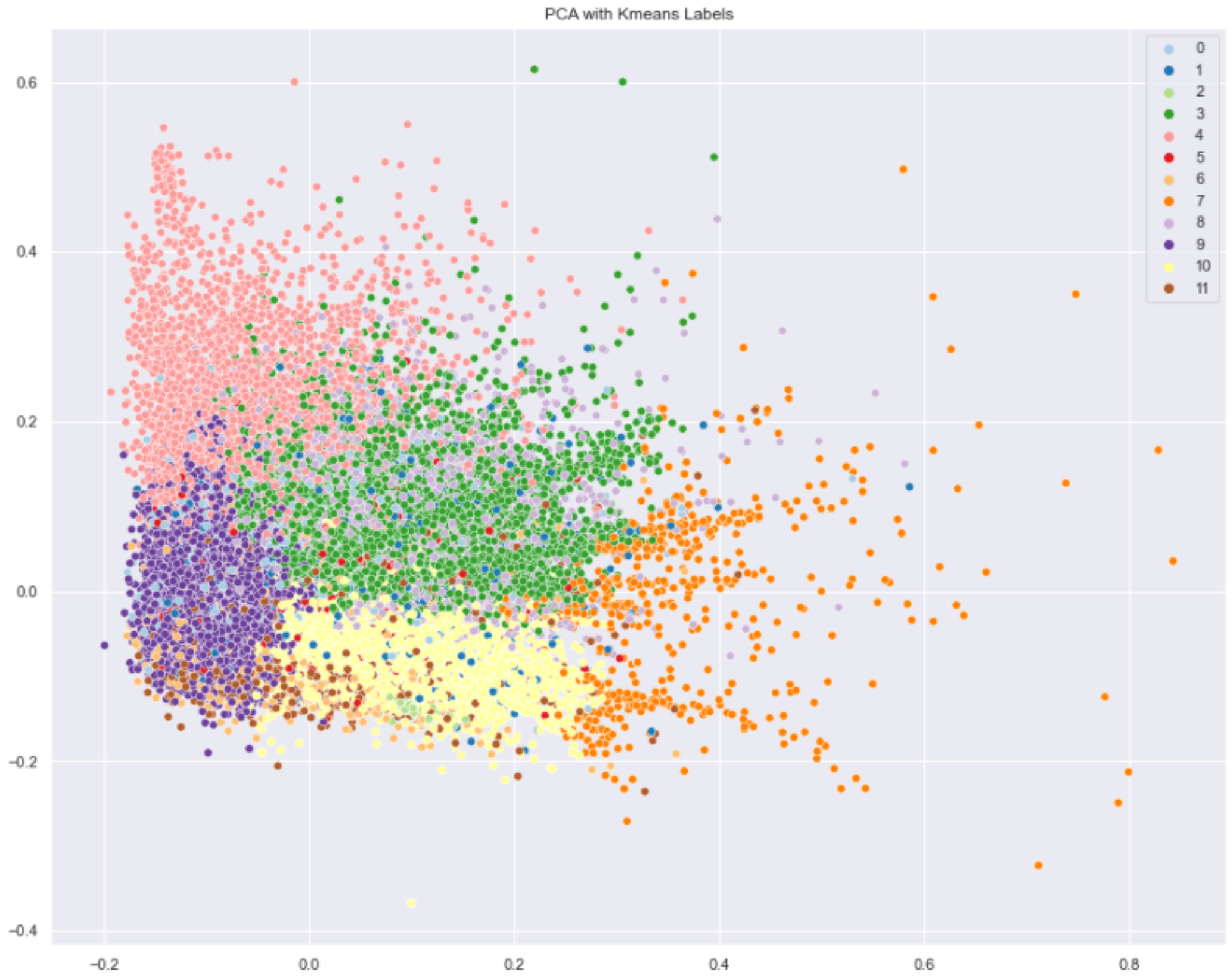
```
k = 12
kmeans = KMeans(n_clusters=k, random_state=42, n_jobs=-1)
y_pred = kmeans.fit_predict(X_reduced)
trunc_df['y'] = y_pred
```

In the following pages we will plot our topic clusters to see how they turn out.

Plotting our Results:

1) PCA

```python
# sns settings
sns.set(rc={'figure.figsize':(15, 12)})

# plot
sns.scatterplot(X_reduced[:,0], X_reduced[:,1], hue=y_pred, legend='full', palette='Paired')
sns.color_palette()
plt.title('PCA with Kmeans Labels')
plt.show()
```



We see there is some grouping of the clusters. The clusters do overlap alot but we do see some discernible grouping. Our legend shows 12 clusters.

Evaluating our model

Next we want to evaluate how well the model classified the clusters. Remember we used kmeans to create 12 unsupervised clusters in y_pred?

```
k = 12
kmeans = KMeans(n_clusters=k, random_state=42, n_jobs=-1)
y_pred = kmeans.fit_predict(X_reduced)
trunc_df['y'] = y_pred
```

We can now use these 'labels' to see how well it can fit unseen data. We split the data into 80/20 train/test sets of our data.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score, recall_score, mean_squared_error
import math
```

```
from sklearn.model_selection import train_test_split

# test set size of 20% of the data
X_train, X_test, y_train, y_test = train_test_split(X.toarray(),y_pred, test_size=0.2, random_state=42)
```

```
# Fit a basic Random Forest model
rf = RandomForestClassifier()
rf_model = rf.fit(X_train, y_train)
```

```
# Make predictions on the test set using the fit model
y_pred = rf_model.predict(X_test)
```

We get below precision and recall of 95% and accuracy of 98%.

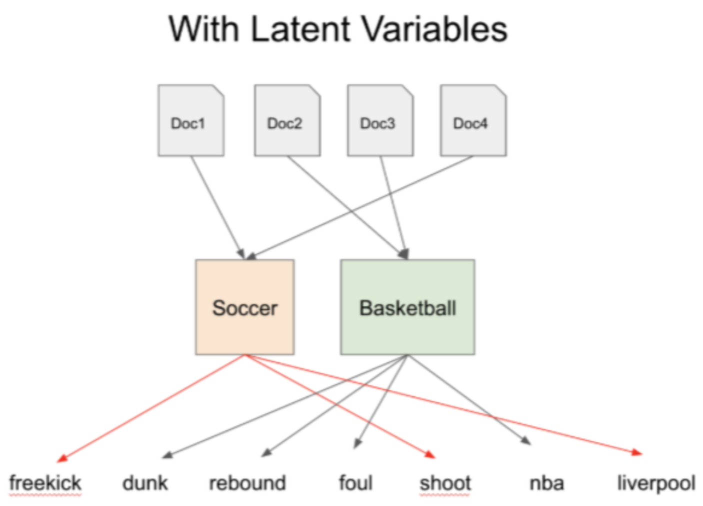
```
# Evalute model predictions using precision and recall
precision = precision_score(y_test, y_pred, average='micro')
recall = recall_score(y_test, y_pred, average='micro')
mse = mean_squared_error(y_test, y_pred)
accuracy = math.sqrt(mse)
print('Precision: {}'.format(round(precision, 3)))
print('Recall: {}'.format(round(recall, 3)))
print('Accuracy: {}'.format(round(max(0,accuracy), 3)))
```

```
Precision: 0.95
Recall: 0.95
Accuracy: 0.987
```

Finally, let's take a look at the keywords and topics created from all of this.

Finding Important Topic Keywords

Next we will look for important topic keywords in each cluster. We use Latent Dirichlet Allocation (LDA) for topic modelling. LDA works by creating a latent layer (unsupervised topics) bridging between documents and tokens. Through iterations it ranks the probability of each topic within each document and ranks each token within each topic.



```python
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction.text import CountVectorizer
```

As before, we choose 12 clusters from our determination of the optimal k above.

```python
vectorizers = []

for cluster in range(0, 12):
    # Creating a vectorizer
    vectorizers.append(CountVectorizer(min_df=10, max_df=0.90, ngram_range=(1, 4),
                                       stop_words='english', lowercase=True,
                                       token_pattern='[a-zA-Z\-][a-zA-Z\-]{2,}'))
```

We vectorize our data into these 12 clusters.

```python
vectorized_data = []

for current_cluster, cvec in enumerate(vectorizers):
    try:
        vectorized_data.append(cvec.fit_transform(trunc_df.loc[trunc_df['y'] == current_cluster, 'processed_text']))
    except Exception as e:
        print(f"Not enough in cluster: {current_cluster}")
        vectorized_data.append(None)
```

We instantiate the Latent Dirichlet Allocation (LDA) model.

```
# number of topics per cluster
TOPICS_PER_CLUSTER = 30

lda_models = []
for cluster in range(0, 12):
    # Latent Dirichlet Allocation Model
    lda = LatentDirichletAllocation(n_components=TOPICS_PER_CLUSTER, max_iter=10, learning_method='online',
                                    verbose=False, random_state=42)
    lda_models.append(lda)
```

We fit_transform our vectorized data in each cluster

```
clusters_lda_data = []

for current_cluster, lda in enumerate(lda_models):
    # print("Current Cluster: " + str(current_cluster))

    if vectorized_data[current_cluster] != None:
        clusters_lda_data.append((lda.fit_transform(vectorized_data[current_cluster])))
```

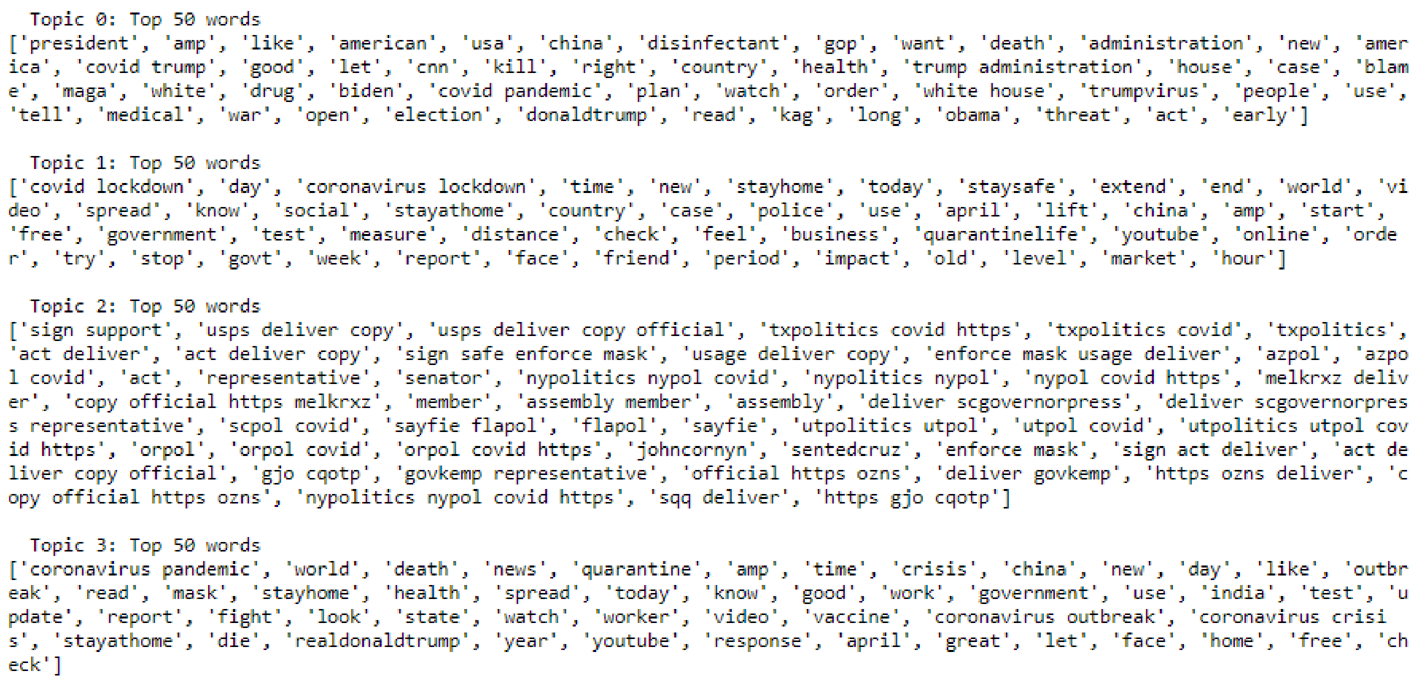
Next we extract our keywords, eliminating a few words adds no new information to the topic.

```
def selected_topics(model, vectorizer, top_n=3):
    current_words = []
    keywords = []
    elim_words = ['http', 'coronavirus covid https','coronavirus covid','covid coronavirus', 'coronavirus https', 'covid https',
                  'coronavirusupdate', 'coronaviruslockdown', 'coronavirusupdates', 'coronavirususa', 'covid', 'pandemic',
                  'virus', 'corona', 'coronaviruspandemic', 'stayathomeandstaysafe', 'live', 'coronavirus', 'https',
                  'covid-', 'coronaupdate', 'need', 'socialdistanacing', 'coronavirusoutbreak', 'coronavirus crisis https',
                  'covid covid']

    for idx, topic in enumerate(model.components_):
        words = [(vectorizer.get_feature_names()[i], topic[i]) for i in topic.argsort()[:-top_n - 1:-1]]
        for word in words:
            if word[0] not in current_words and word[0] not in elim_words:
                keywords.append(word)
                current_words.append(word[0])

    keywords.sort(key = lambda x: x[1])
    keywords.reverse()
    return_values = []
    for ii in keywords:
        return_values.append(ii[0])
    return return_values
```

```
all_keywords = []
for current_vectorizer, lda in enumerate(lda_models):
    # print("Current Cluster: " + str(current_vectorizer))

    if vectorized_data[current_vectorizer] != None:
        all_keywords.append(selected_topics(lda, vectorizers[current_vectorizer]))
```

Lastly, let's print out these keywords within our topics:

```
for topic in range(12):
    print(f'  Topic {topic}: Top 50 words')
    print(all_keywords[topic][:50])
    print(" ")
```



```
  Topic 0: Top 50 words
['president', 'amp', 'like', 'american', 'usa', 'china', 'disinfectant', 'gop', 'want', 'death', 'administration', 'new', 'amer
ica', 'covid trump', 'good', 'let', 'cnn', 'kill', 'right', 'country', 'health', 'trump administration', 'house', 'case', 'blam
e', 'maga', 'white', 'drug', 'biden', 'covid pandemic', 'plan', 'watch', 'order', 'white house', 'trumpvirus', 'people', 'use',
'tell', 'medical', 'war', 'open', 'election', 'donaldtrump', 'read', 'kag', 'long', 'obama', 'threat', 'act', 'early']

  Topic 1: Top 50 words
['covid lockdown', 'day', 'coronavirus lockdown', 'time', 'new', 'stayhome', 'today', 'staysafe', 'extend', 'end', 'world', 'vi
deo', 'spread', 'know', 'social', 'stayathome', 'country', 'case', 'police', 'use', 'april', 'lift', 'china', 'amp', 'start',
'free', 'government', 'test', 'measure', 'distance', 'check', 'feel', 'business', 'quarantinelife', 'youtube', 'online', 'orde
r', 'try', 'stop', 'govt', 'week', 'report', 'face', 'friend', 'period', 'impact', 'old', 'level', 'market', 'hour']

  Topic 2: Top 50 words
['sign support', 'usps deliver copy', 'usps deliver copy official', 'txpolitics covid https', 'txpolitics covid', 'txpolitics',
'act deliver', 'act deliver copy', 'sign safe enforce mask', 'usage deliver copy', 'enforce mask usage deliver', 'azpol', 'azpo
l covid', 'act', 'representative', 'senator', 'nypolitics nypol covid', 'nypolitics nypol', 'nypol covid https', 'melkrxz deliv
er', 'copy official https melkrxz', 'member', 'assembly member', 'assembly', 'deliver scgovernorpress', 'deliver scgovernorpres
s representative', 'scpol covid', 'sayfie flapol', 'flapol', 'sayfie', 'utpolitics utpol', 'utpol covid', 'utpolitics utpol cov
id https', 'orpol', 'orpol covid', 'orpol covid https', 'johncornyn', 'sentedcruz', 'enforce mask', 'sign act deliver', 'act de
liver copy official', 'gjo cqotp', 'govkemp representative', 'official https ozns', 'deliver govkemp', 'https ozns deliver', 'c
opy official https ozns', 'nypolitics nypol covid https', 'sqq deliver', 'https gjo cqotp']

  Topic 3: Top 50 words
['coronavirus pandemic', 'world', 'death', 'news', 'quarantine', 'amp', 'time', 'crisis', 'china', 'new', 'day', 'like', 'outbr
eak', 'read', 'mask', 'stayhome', 'health', 'spread', 'today', 'know', 'good', 'work', 'government', 'use', 'india', 'test', 'u
pdate', 'report', 'fight', 'look', 'state', 'watch', 'worker', 'video', 'vaccine', 'coronavirus outbreak', 'coronavirus crisi
s', 'stayathome', 'die', 'realdonaldtrump', 'year', 'youtube', 'response', 'april', 'great', 'let', 'face', 'home', 'free', 'ch
eck']
```

Let's see how well the unsupervised learning model organized these keywords and if there is a discernible topic.

Topic 0: *america, usa, gop, country, administration, election, donaldtrump, obama, biden, china, maga, cnn*

We see the political wheels already turning even in April. The early seeds of the red & blue war that will continue until the beginning of 2021.

Topic 1: *lockdown, extend, stayhome, quarantinelife, stay safe, face, time, online, end, lift, market, business, government, world*

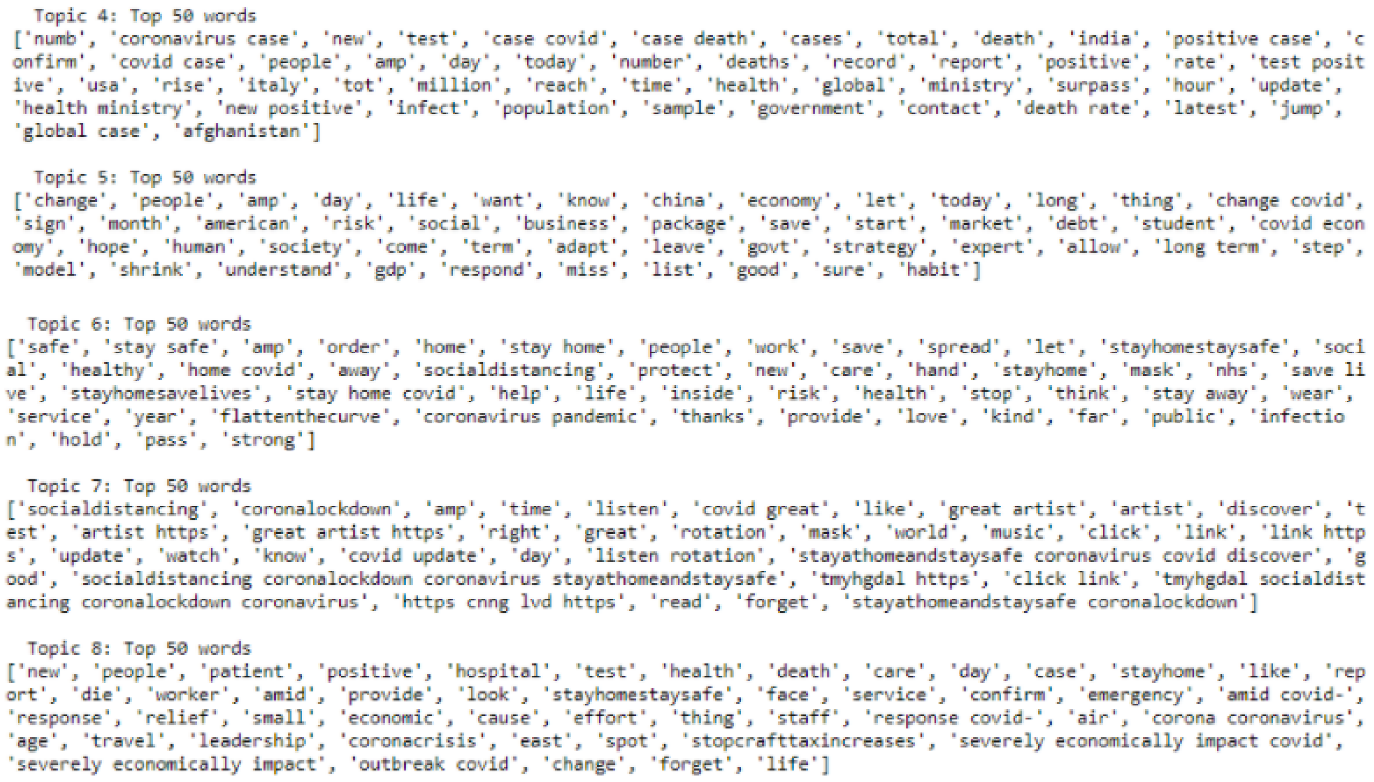
Mid-March was the beginning of the covid lockdown so this was a natural topic of debate around that time. How long will it extend to? How will it affect businesses and markets? How will it end?

Topic 2: *usps deliver copy,usps deliver copy official, sign support, act deliver, sign act deliver, enforce mask usage*

USPS was a big topic back then. It was in deep financial trouble and Trump refused to bailout the Postal Service. Delivery was also a big topic of discussion since stay home policies were mandated, everything relied on delivery - food, goods, medicine, masks and even ballots - many became a huge arena for debate.

Topic 3: *outbreak, spread, crisis, pandemic, quarantine, coronavirus outbreak, china, government, india*

This was an obvious topic - the word pandemic was avoided as long as possible in describing coronavirus - until early March when we saw the spread of virus reach crisis levels within a few short months.



```
  Topic 4: Top 50 words
['numb', 'coronavirus case', 'new', 'test', 'case covid', 'case death', 'cases', 'total', 'death', 'india', 'positive case', 'c
onfirm', 'covid case', 'people', 'amp', 'day', 'today', 'number', 'deaths', 'record', 'report', 'positive', 'rate', 'test posit
ive', 'usa', 'rise', 'italy', 'tot', 'million', 'reach', 'time', 'health', 'global', 'ministry', 'surpass', 'hour', 'update',
'health ministry', 'new positive', 'infect', 'population', 'sample', 'government', 'contact', 'death rate', 'latest', 'jump',
'global case', 'afghanistan']

  Topic 5: Top 50 words
['change', 'people', 'amp', 'day', 'life', 'want', 'know', 'china', 'economy', 'let', 'today', 'long', 'thing', 'change covid',
'sign', 'month', 'american', 'risk', 'social', 'business', 'package', 'save', 'start', 'market', 'debt', 'student', 'covid econ
omy', 'hope', 'human', 'society', 'come', 'term', 'adapt', 'leave', 'govt', 'strategy', 'expert', 'allow', 'long term', 'step',
'model', 'shrink', 'understand', 'gdp', 'respond', 'miss', 'list', 'good', 'sure', 'habit']

  Topic 6: Top 50 words
['safe', 'stay safe', 'amp', 'order', 'home', 'stay home', 'people', 'work', 'save', 'spread', 'let', 'stayhomestaysafe', 'soci
al', 'healthy', 'home covid', 'away', 'socialdistancing', 'protect', 'new', 'care', 'hand', 'stayhome', 'mask', 'nhs', 'save li
ve', 'stayhomesavelives', 'stay home covid', 'help', 'life', 'inside', 'risk', 'health', 'stop', 'think', 'stay away', 'wear',
'service', 'year', 'flattenthecurve', 'coronavirus pandemic', 'thanks', 'provide', 'love', 'kind', 'far', 'public', 'infectio
n', 'hold', 'pass', 'strong']

  Topic 7: Top 50 words
['socialdistancing', 'coronalockdown', 'amp', 'time', 'listen', 'covid great', 'like', 'great artist', 'artist', 'discover', 't
est', 'artist https', 'great artist https', 'right', 'great', 'rotation', 'mask', 'world', 'music', 'click', 'link', 'link http
s', 'update', 'watch', 'know', 'covid update', 'day', 'listen rotation', 'stayathomeandstaysafe coronavirus covid discover', 'g
ood', 'socialdistancing coronalockdown coronavirus stayathomeandstaysafe', 'tmyhgdal https', 'click link', 'tmyhgdal socialdist
ancing coronalockdown coronavirus', 'https cnng lvd https', 'read', 'forget', 'stayathomeandstaysafe coronalockdown']

  Topic 8: Top 50 words
['new', 'people', 'patient', 'positive', 'hospital', 'test', 'health', 'death', 'care', 'day', 'case', 'stayhome', 'like', 'rep
ort', 'die', 'worker', 'amid', 'provide', 'look', 'stayhomestaysafe', 'face', 'service', 'confirm', 'emergency', 'amid covid-',
'response', 'relief', 'small', 'economic', 'cause', 'effort', 'thing', 'staff', 'response covid-', 'air', 'corona coronavirus',
'age', 'travel', 'leadership', 'coronacrisis', 'east', 'spot', 'stopcrafttaxincreases', 'severely economically impact covid',
'severely economically impact', 'outbreak covid', 'change', 'forget', 'life']
```

Topic 4: *positive, rate, case, record, number, death, surpass, rise, test positive, total, test, infect, population, jump, million*

One of the hot topics is the escalation of covid cases and the rising numbers of infection and death. This was obviously an area of great concern.

Topic 5: *economy, business, market, debt, student, adapt, long term, shrink, gdp, risk, long term*

Another topic is the economy, the market, the shrinking gdp and concern over the long term economic effects

Topic 6: *stay safe, order, stay home, socialdistancing, mask, away, inside, stayhomesavlives, kind, care*

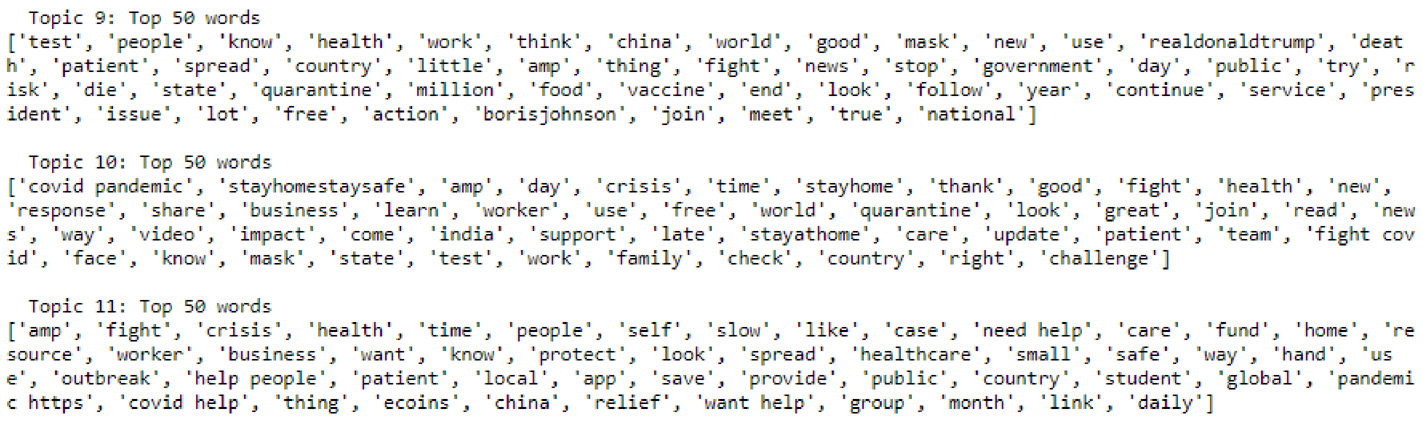
Social distancing and self isolation became the new normal. Along the same line - stay home, save lives, stay safe.

Topic 7: *listen, great artist, discover, read, music, coronalockdown, watch, listen rotation, click link, time*

With a lot of time spent indoors people turn to online entertainment and discovering new virtual hobbies

Topic 8: *patient, positive, hospital, test, health, death, care, case, report, confirm, emergency, staff, relief*

There's definitely a focus on front line and health care workers with hospital resources being taxed beyond the limit with the increase in covid patients and deaths.



```
  Topic 9: Top 50 words
['test', 'people', 'know', 'health', 'work', 'think', 'china', 'world', 'good', 'mask', 'new', 'use', 'realdonaldtrump', 'deat
h', 'patient', 'spread', 'country', 'little', 'amp', 'thing', 'fight', 'news', 'stop', 'government', 'day', 'public', 'try', 'r
isk', 'die', 'state', 'quarantine', 'million', 'food', 'vaccine', 'end', 'look', 'follow', 'year', 'continue', 'service', 'pres
ident', 'issue', 'lot', 'free', 'action', 'borisjohnson', 'join', 'meet', 'true', 'national']

  Topic 10: Top 50 words
['covid pandemic', 'stayhomestaysafe', 'amp', 'day', 'crisis', 'time', 'stayhome', 'thank', 'good', 'fight', 'health', 'new',
'response', 'share', 'business', 'learn', 'worker', 'use', 'free', 'world', 'quarantine', 'look', 'great', 'join', 'read', 'new
s', 'way', 'video', 'impact', 'come', 'india', 'support', 'late', 'stayathome', 'care', 'update', 'patient', 'team', 'fight cov
id', 'face', 'know', 'mask', 'state', 'test', 'work', 'family', 'check', 'country', 'right', 'challenge']

  Topic 11: Top 50 words
['amp', 'fight', 'crisis', 'health', 'time', 'people', 'self', 'slow', 'like', 'case', 'need help', 'care', 'fund', 'home', 're
source', 'worker', 'business', 'want', 'know', 'protect', 'look', 'spread', 'healthcare', 'small', 'safe', 'way', 'hand', 'us
e', 'outbreak', 'help people', 'patient', 'local', 'app', 'save', 'provide', 'public', 'country', 'student', 'global', 'pandemi
c https', 'covid help', 'thing', 'ecoins', 'china', 'relief', 'want help', 'group', 'month', 'link', 'daily']
```

Topic 9: *test, health, china, world, realdonaldtrump, spread, die, million, food, vaccine, mask, action, government, quarantine, borisjohnson, national*

This list of keywords is a more generalized list about the spread and rise of covid and covid deaths. It seems to center on government steps or missteps in meeting the crisis, respectively receiving praise or blame for it.

Topic 10: *stayhomestaysafe, crisis, thank, response, learn, free, join, read, share, look, great, team, care, good*

This topic seems to center on encouraging one another to continue our resolve to fight, to encourage, to express gratitude as each one is doing their part in beating this global crisis.

Topic 11: *crisis, fight, time, need help, fund, resource, business, help people, local, covid help, want help, group, save, protect, healthcare, worker, relief*

This topic centers around helping and providing resources or funds to people and businesses in need. There was definitely a focus on essential workers and first responders who risked their lives in order to provide services to the rest of the world who stayed indoors. There was a rise in financial needs for the unemployed and for struggling businesses who needed the relief funds to survive.

In Conclusion

We've **explored** a month's worth of data from twitter from the end of March to the end of April 2020. We used **unsupervised learning techniques** to perform **natural language processing** on a smaller portion of the entire set (there were originally 7.3 million rows of data, of which we randomly chose about 110,000 rows from). After we cleaned, tokenized and vectorized our text data, we decided on a few topic clustering methods to find out what important topics emerged. *Our original intent was to answer if we can discern what trends / sentiments / behaviors / challenges arose during our period of self-seclusion?* Which would in turn could provide insight and lead to future policy decisions. The above **12 topics** give a good snapshot of important topics of discussions that impacted the worldwide community during the lockdown.

We used scikit-learn's **Kmeans** and **Latent Dirichlet Allocation** to do unsupervised modelling of topics. The models were dependent on being fed the number of topics to group into - which we derived from doing the elbow method test over a range of k numbers to find the optimal k.

We used **PCA** to reduce the dimensionality of our vectorized data to drop less important features and noise. For future use, it may be better to use smaller test datasets to try out different parameters and settings before proceeding to the larger set.

Final Thoughts

We set out **to find out** *what happens when the world goes into self isolation for a long period of time* through **natural language processing** tweets. I think it's still a good area of study. The study was able to **uncover** some topic groupings. **One weakness** in this study is that the data only scraped tweets with the covid / coronavirus hashtag. This **limits** us to data around the subject of covid. A **better set** of data is all tweets during the lockdown whether or not it was directly meant to relate to covid. **We want to examine** what trends emerged out of the extended time in lockdown and I think we may be able to get topics such as "exercised more", "gained weight", "painted my walls", "where are my friends?", etc because they aren't necessarily covid related but lockdown related. **However**, even with the limits of the data, in the end, the nlp still produced really nice results with topic modelling the data we have.