MATH 189Z Homework 2, Topic Modeling of COVID-19 Tweets Over Time (and PCA)

April 13, 2020

Name: Eve Kazarian

In this homework, you will be using topic modeling to identify topics in tweets about COVID-19 collected from February through April. You will use PCA to visualize the results of your topic modeling as well as investigate the semantic meaning of the topics you have found and the prevalence of these topics over time. To complete this homework assignment, run each block of code, filling in code as needed.

[366]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

0.1 Loading Tweet Data

[367]:
```
tweets = pd.read_csv('april_data.csv')
```

[368]:
```
tweets.head()
```

[368]:

	Unnamed: 0	date	text
0	0	2020-04-05 23:59:34	PROMISING NEWS! We asked a local doctor about …
1	1	2020-04-05 23:59:11	ToP 10 #LightsOverLockdown Videos That could A…
2	2	2020-04-05 23:59:01	#MakeChinaPay China owes us all. #Covid_19 #Ch…
3	3	2020-04-05 23:58:34	#Hydroxycloroquine shows promising results in …
4	4	2020-04-05 23:58:29	Yes and their numbers are going down!

1 Topic Modeling

1.1 Building Our Model

To do our topic modeling, we will be using Latent Dirichlet Allocation to do topic modeling, which you learned about in lecture. Don't worry, we will not be implementing Latent Dirichlet Allocation, instead we will be using the tools already available to us (sklearn) in order to do topic modeling. For a refresher on this method, you can check out this blog post. To start we will build up some understanding of how to use sklearn's LDA functionality.

[369]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction import text
```

We will start out by adding to our stop words. These are words that may be repeated often in the tweets, but do not add any topic information. These are words like 'the', 'a', 'that', 'almost', 'every', etc. You can check out the existing stop words using text.ENGLISH_STOP_WORDS if you want to know what's included. What you need to do is add words that are likely not in the existing stop word list that will not add any information to our topics. Because all the tweets are already about COVID-19, words used to identify the virus are likely not helpful, nor are things like 'https' and 'com' from links. Think about what words might come up a lot but do not differentiate tweets and add them to the stop list. After you run the topic modeling and see the words in the topics, continue to add unimportant words to the stop list to refine your topics (we added about 15 additional stop words but you can add as many as you think are appropriate).

1.2 TASK 1: add stop words

[370]:
```
my_stops = text.ENGLISH_STOP_WORDS.union(

    ['https', 'com', 'COVID-19', 'covid', 'coronavirus', 'SARS', 'corona',␣
↪'sick', '#COVID-19', '#Covid',
    'Covid', 'Hey', '#coronavirus', 'Corona virus', 'Corona', '19', 'hey',␣
↪'sars', 'virus', 'covid_19', 'pic', 'twitter',
    'just', 'day', 'today', 'new', 'got', 'pandemic', 'death', 'crisis',␣
↪'covid19', 'china', 'chinese', 'China','Chinese',
    'www', '2020', 'like', 'people', 'http']

)
```

Next, we will vectorize the tweets. This makes it so each tweet is represented by an w-dimensional vector, where w is the total number number of unique words in all the tweets (not including the stop words).

[371]:
```
vectorizer = CountVectorizer(min_df=10, stop_words=my_stops)
counts = vectorizer.fit_transform(tweets.text)
```

Now we will run Latent Dirichlet Allocation to find our topics. Check out the documentation in order to understand all of the parameters.

[372]:
```
as_topics = LatentDirichletAllocation(n_components=8, random_state=0, n_jobs=6).
↪fit(counts)
```

[373]:
```
def print_topics(topics,vectorizer):
    """
    Prints top 12 most important words for each topic in descending importance
    """
    topic_dists = (topics.components_.T / topics.components_.sum(axis=1)).T
    for comp in range(len(topic_dists)):
        top_i = np.argsort(topic_dists[comp])[-12:][::-1]
        print()
        print([key for key, value in vectorizer.vocabulary_.items() if value in␣
↪top_i])
```

[374]:
```
print_topics(as_topics, vectorizer)
```

```
['news', 'hydroxychloroquine', 'drug', 'trump', 'says', 'watch', 'negative',
'dr', '04', 'states', 'united', 'told']

['free', 'time', 'going', 'doing', 'right', 'don', 'house', 'lot', 'amp',
'queen', 'help', 'face']

['world', 'home', 'said', 'want', 'work', 'don', 'health', 'thank', 'great',
'stay', 'know', '5g']

['news', 'world', 'deaths', 'live', 'dead', 'years', 'cases', 'need', 'america',
'food', 'lockdown', 'italy']

['hospital', 'home', 'symptoms', 'let', 'lives', 'save', 'really', 'amp',
'fight', 'stay', 'india', 'narendramodi']

['social', 'tiger', 'bronx', 'zoo', 'tests', 'positive', 'tested', 'test',
'help', 'west', 'ik_saviourofnation', 'team_mastanda']

['patients', 'hospital', 'deaths', 'love', 'tests', 'positive', 'away', 'stop',
'died', 'breaking', 'testing', 'black']

['president', 'trump', 'deaths', 'use', 'ago', '000', 'cases', 'response',
'amp', 'fuck', 'days', 'outbreak']
```

1.3 Evaluating our Model

This is an unsupervised learning technique meant to help us find structure in our data. The data will always be divided into topics, however that does not mean that the topics will have meaning. This makes the qualitative part of topic modeling very important because you need to be able to look at the topics generated above and understand whether the model seems to have captured true structure or just noise. Now we can think about ways to visualize what we have just created.

One way we can gain intuition about our model is by checking how strongly it is classifying tweets into one category or another. In other words, how sure the model is of the topic of a tweet. When predicting the topic for a tweet, the trained model returns a distribution over the possible topics, which is referred to as that tweet in *topic space*. For example, if we had three topics it would return a vector of length 3 with the 3 values summing to 1. The topic represented by the highest probability is the principal topic of that tweet. The higher that maximum value is compared to the other probabilities, the stronger the association of that tweet to its principal topic. If tweets tend to have strong principal topics, then there is a lot of structure in the topic space of the tweets and the topics are more likely to be useful.

By measuring this charactaristic of our model, we are making sure our model identified more general structure in the tweets and didn't train strictly on noise. While this doesn't mean that the topics have **meaning**, it is a check that there is clustering going on. To visualize whether this clustering is happening we can plot our points and color them based on their given topic. However, our data is more than 3-dimensions, making it difficult to visualize. That's where dimensionality reduction techniques come in. There are many techniques out there, but we are going to focus on PCA as it is ubiquitous, easy to understand, and very useful.

If you want to explore other demonsionality reduction tools, tSNE is a widely used one as well.

1.4 TASK 2: Implement PCA

Before we perform PCA to visualize our topic modeling, check out this blog post on covariance matrices and this blog post on PCA if you are unfamiliar with either topic. To really solidify your understanding of PCA, you are going to **implement it yourself**.

To do this, please fill in the provided helper functions and PCA wrapper function. You **may not** use any built in PCA functions but you can use anything else. The following example depends on your implementation of this code so make sure to implement it before running that.

[375]:
```
def get_covariance(data):
    """
    Given a matrix of data, returns covariance matrix

    Hint: Make sure to standardize your data and center it around zero.
    Try to do this in a vectorized manner, our solution was less than 5 lines.
    """
    data = np.array(data)

    # center and standardize matrix
    data = (data - data.mean())/(data.max())

    # assuming matrix has one observation per row (n)
    # variables represented by columns (m)
    # covariance matrix of dimensions m*m

    # Xtranspose (dim m*n) dot product w X (dim n*m)
    Xt_X = np.dot(data.T,data)
    # compute covariance matrix
```

```
    covariance = Xt_X/(len(data)-1)

    return covariance
```

[376]:
```
def pca(data):
    """
    Given a matrix of data, returns sorted eigenvalues and eigenvectors of␣
 ↪covariance matrix

    Hint: This function should use your get_covariance() function and you might␣
 ↪find np.linalg.eig()
            and np.argsort() helpful

    """
    # get eigenvectors and eigenvalues of covariance matrix
    eigen_vals_orig, eigen_vectors_orig = np.linalg.eig(get_covariance(data))

    # sort eigenvalues and then eigenvectors based on eigenvalue order
    indx = np.argsort(eigen_vals_orig)[::-1]
    eigen_vals = eigen_vals_orig[indx]
    eigen_vectors = eigen_vectors_orig[:,indx]

    return eigen_vals, eigen_vectors
```

[377]:
```
def pca_transform(data, num_dims = 2):
    """
    Given a matrix of data, returns data projected onto num_dims principal␣
 ↪components (use your pca() function)

    """
    # standardize matrix
    data = (data - np.mean(data, axis=0)) / np.std(data, axis=0)

    # center matrix at origin
    Z = data - np.mean(data, 0)

    P_star = (pca(data)[1])[:, :num_dims]

    # ZP*
    transformed_data = np.dot(Z, P_star)

    return transformed_data
```

[378]:
```
def plot_pca(data, graph_title='Transformed Data'):
    """
    Given a matrix of data, runs PCA and plots transformed data onto first 2␣
 ↪principal components
```

```
    """
    labels = np.argmax(data, axis=1)
    reduced_data = pca_transform(data)

    scatter = plt.scatter(reduced_data[:,0], reduced_data[:,1], c = labels)
    plt.legend(*scatter.legend_elements())
    plt.title(graph_title)
    plt.show()
```

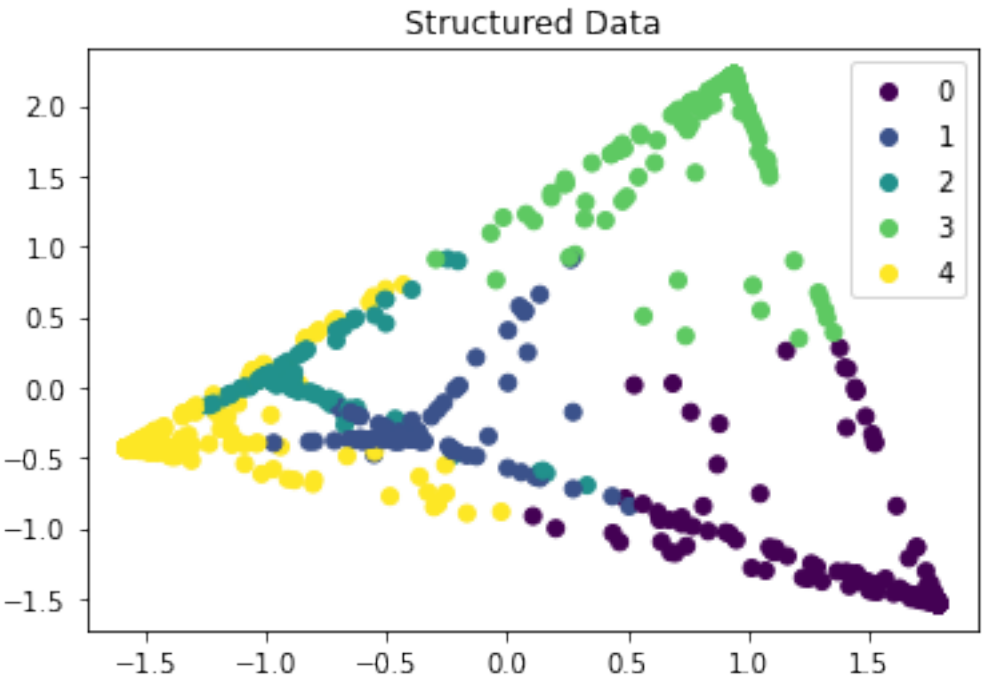
1.5 Using Principal Components Analysis to visualize structure

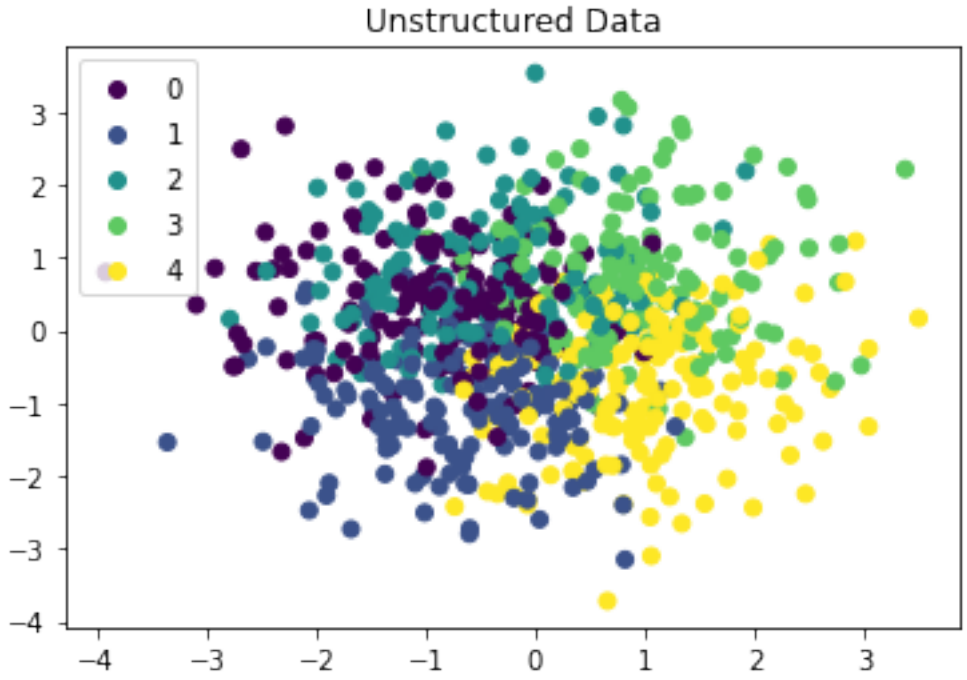
Now that we have our PCA up and running, we can gain some intuition on what our results might be showing us. From our visualization, we want to be able to determine if tweets are grouped tightly around a series of distinct topics (structured) or if tweets are not clearly defined by any one topic (unstructured).

When we do LDA topic modeling, we are assuming that tweets are the results of samples from an underlying dirichlet distribution. Therefore we can make data that "looks" structured or unstructured by sampling from a corresponding Dirichlet distribution. In the structured data visualization we are sampling from a distribution with a high likelihood of a (1.0,0,0,…,0) distribution, (or some permutation of that result), and in the unstructured data visualization we are sampling from a distribution with a high probability of a uniform vector.

NOTE: If you are getting an error with the plotting function, try updating matplotlib using the command `pip install --upgrade matplotlib` in your command line. Restart your notebook before trying again.

[379]:
```
structured_data = np.random.dirichlet(np.ones(5)*.05, 800)
unstructured_data = np.random.dirichlet(np.ones(5)*100, 800)

plot_pca(structured_data, graph_title='Structured Data')
plot_pca(unstructured_data, graph_title='Unstructured Data')
```





We can see from the resulting graphs that PCA does a pretty good job revealing whether there is a lot of structure or if there is very little structure. Taking this information, we can compare our

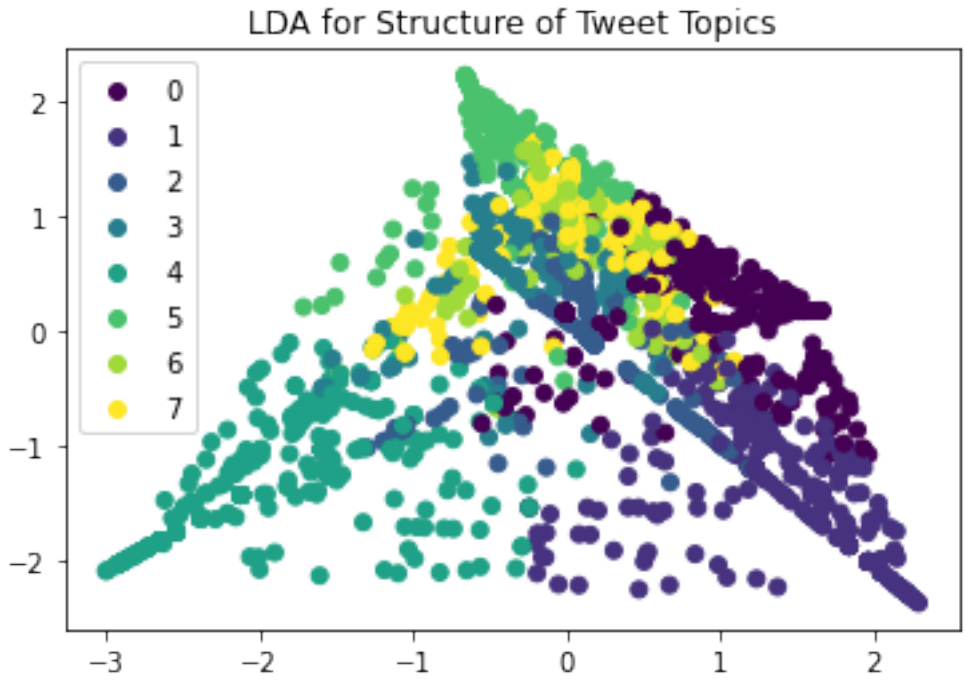
results from twitter to these graphs to get a better idea of the structure of our results.

1.6 TASK 3: Visualize Results of LDA with PCA

Now using the documentation for sklearn's LDA and the given `plot_pca()` function, visualize the structure of the topics of the tweets.

Hint: You need to `transform` your vectorized tweets into topic space.

[380]:
```
# transform vectorized tweets into topic space and plot
plot_pca(as_topics.transform(counts), "LDA for Structure of Tweet Topics")
```



2 Tweets Over Time

Now that we have an understanding of LDA, how the code works, and a way to evaluate it, we can look into ways to measure change in topics over time. We will look at tweets from February, March and April. These tweets were collected over a 6 day interval about 25 days apart.

To observe change over time, you will perform LDA on a data set with all three months of tweets. Then, for each topic, we can look at the frequency of tweets from each month to see if some topics were more popular at different points in time.

[381]:
```
# Loading tweets
feb_tweets = pd.read_csv('feb_data.csv')
march_tweets = pd.read_csv('march_data.csv')
april_tweets = pd.read_csv('april_data.csv')
```

```
all_tweets = pd.concat([feb_tweets, march_tweets, april_tweets]).
 ↪reset_index(drop=True)
```

2.1 TASK 4: Perform LDA on All Tweets

The variables that you will need after performing LDA for our visualization are:

`all_vectorizer` - the vectorizer you create in order to get the counts for all the tweets

`all_counts` - the ndarray of vectorized tweets

`all_as_topics` - the trained LDA model. We used 20 topics but you can adjust the number as you see fit.

Don't forget to create a new vectorizer and that you can use `print_topics(as_topic, vectorizer)` to print the topics

[382]:
```
all_vectorizer = CountVectorizer(min_df=10, stop_words=my_stops)
all_counts = all_vectorizer.fit_transform(all_tweets.text)
all_as_topics = LatentDirichletAllocation(n_components=8, random_state=0,␣
 ↪n_jobs=6).fit(all_counts)

print_topics(all_as_topics, all_vectorizer)
```

```
['japan', 'world', 'outbreak', 'cases', 'total', 'cruise', 'ship', 'quarantine',
'health', 'passengers', 'spread', 'princess']

['time', 'fight', 'need', 'amp', 'help', 'outbreak', 'government', 'global',
'make', 'medical', 'support', 'lives']

['hospital', 'died', 'wuhan', 'outbreak', 'deaths', 'province', 'news', 'apple',
'really', 'doesn', 'director', 'dies']

['time', 'amp', 'outbreak', '000', 'cases', 'social', 'spread', 'reut', 'rs',
'home', 'caused', 'corruption']

['hospital', 'patients', 'infected', 'city', 'bit', 'ly', 'positive', 'health',
'breaking', 'tested', 'old', 'tests']

['time', 'need', 'work', 'world', 'stay', 'going', 'don', 've', 'know', 'stop',
'home', 'did']

['says', 'white', 'said', 'news', 'trump', 'negative', 'realdonaldtrump',
'house', 'tested', 'tests', 'test', 'president']

['outbreak', 'deaths', 'south', 'korea', 'cases', 'confirmed', 'state', 'masks',
'hong', 'kong', 'testing', 'march']
```

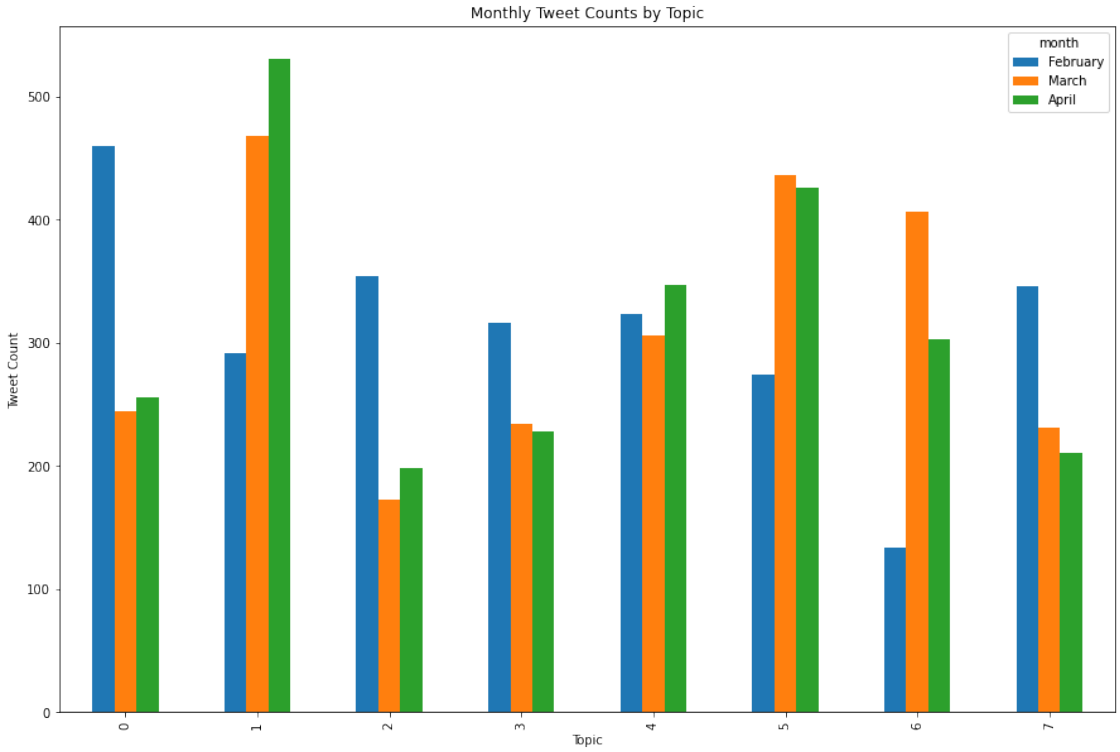
Now we can visualize how topics have changed over time.

[383]:
```
# Classify tweets into topics
tweet_dists = all_as_topics.transform(all_counts)
tweet_labels = np.argmax(tweet_dists, axis=1)
all_tweets = all_tweets.assign(tweet_labels= tweet_labels)

# Label tweets by month
months = pd.Series(["February"] * feb_tweets.shape[0] + ['March'] *␣
↪march_tweets.shape[0] + ['April'] * april_tweets.shape[0])
all_tweets = all_tweets.assign(month=months)

# Get monthly counts for each topic
topic_counts = all_tweets.groupby(['tweet_labels','month']).count()
topic_counts.drop(['date','text'], axis=1, inplace=True)
topic_counts.columns =['monthly_count']

unstacked_topics = topic_counts.unstack()
unstacked_topics = unstacked_topics.monthly_count
unstacked_topics = unstacked_topics.reindex(columns=['February', 'March',␣
↪'April'])

unstacked_topics.head()
```

[383]:

month	February	March	April
tweet_labels			
0	460	245	256
1	292	468	531
2	354	173	198
3	316	234	228
4	324	306	347

[384]:
```
# Plot monthly counts for each topic
unstacked_topics.plot.bar(figsize=(15,10))
plt.title('Monthly Tweet Counts by Topic')
plt.xlabel('Topic')
plt.ylabel('Tweet Count')
plt.show()
```



2.2 TASK 5: Discuss Results

Now it is time to do qualitative analysis of your topic modeling. You final task is to look at the graph above and choose a topic with a majority from each month and a topic that seems to be present in all months (4 total topics) and assign a semantic meaning to each based on the words associated with that topic (when you print the topics using print_topics() they are printed in the same order as the graph). For example if the words were [‘test’, ‘positive’, ‘negative’...] you could say ‘Testing for COVID-19’. Comment on whether the results on the bar graph make sense after assigning the semantic meaning to the topic.

2.2.1 Discussion

- Topic 0: ‘Diamond Princess Japanese Cruise Incident’
  - This topic includes key words such as ‘passengers,’ ‘princess,’ ‘japan,’ and ‘cruise,’ indicating that it relates to when passengers disembarked the infected Diamond Princess Cruise in Japan.
  - It makes sense that this topic is prevalent in February since the incident occurred then.
- Topic 6: ‘Trump Gets Tested for COVID-19’
  - Some key words that indicate this topic are: ‘president,’ ‘negative,’ ‘realdonaldtrump,’ ‘white,’ and ‘house.’
  - It makes sense that this topic is prevalent in March since President Trump was tested then.
- Topic 1: ‘Fight Against Virus’

  - This topic likely includes tweets calling for support for relief efforts and any comments on the fight against the virus. Key words that indicate this topic are 'time,' 'fight,' 'need,' and 'support.'
  - It makes sense that most tweets relating to this topic occur in April and have been steadily increasing since February since more tweets calling for support and fighting against the virus occur as the virus progresses.
- Topic 4: 'Testing for COVID-19'
  - Key words indicating this topic include 'hospital,' 'patients,' 'positive,' 'infected,' 'tested,' and 'tests'.
  - This topic appears with relatively equal frequency across all three months, which makes sense since testing picked up in February and has been occurring at roughly the same frequency since.