Textual Clustering Of COVID-19 Tweets

IST 664

Group 8:
Ryan Ondocin (rjondoci@syr.edu)
Nikhil Gohil (nrgohil@syr.edu)
Kashish Shah (kshah11@syr.edu)
Renjie Yin (ryin04@syr.edu)

Table of Contents

Due Friday, May 8 at 11:59 pm

Abstract:

Given the vast influx of information surrounding the COVID-19 pandemic and its exponential transmission via various social networking platforms, it is difficult for users to coherently process all of the surrounding buzz. Social media sites such as Facebook and Twitter generate ad-revenue based on user engagement, which allows for rumors and misinformation to spread like wildfire. Due to the broad scope and lack of relevantly labeled data, we will not focus on identifying misinformation. Instead, we will attempt to give the user a new point-of-reference for digesting social media.

By topically clustering tweets via the use of a variety of NLP and ML techniques, we will offer a new lens into the COVID-19 pandemic that can help users digest information in a much simpler manner. The interactive bokeh plot we create will aid in understanding the information dynamics between Twitter users and the coronavirus. ***Disclaimer***: Our modeling process was inspired by a publication on Covid-19 Literature Clustering. [3] Many of our ideas were molded from their works on health-care literature made available in the COVID-19 Research Dataset Challenge on Kaggle.

For this project, we will cover the following

- Web scraping and Data
- Preprocessing and Feature Engineering
- Sentiment Analysis TextBlob (brief insight)
- SpaCy tokenization
- Vectorization (TF-IDF/BOW) and Dimensionality reduction via PCA
- K-means clustering model (validation and visualization)
- Bokeh plot

We hope you enjoy this exploration.

Introduction:

Since the end of January, New Coronary Pneumonia (COVID-19) has become the epicenter of discussion at home and abroad. The virus has claimed over a quarter million lives (as of May 7, 2020, [2]) devastated world economies and raised questions between trusted authorities (WHO, CDC) and the people affected by their policies, thus changing the dynamic in how people interact both on social media and in-person. Terabytes of data are generated each day by Twitter, Facebook and Instagram. Such platforms have become a valuable source of insight into studying the information flow of these dire times. We will visually cluster some of the tweets being shared on Twitter regarding COVID via K-Means. This approach can be seen as a means of noise reduction amidst the deafening social buzz of the virus. Relying on metadata (retweets/mentions/location/favorites) could deduct from the uniquely text-based approach we are attempting but it could serve in validating our findings. This philosophy allows us to fully leverage NLP techniques that can be integrated into an unsupervised learning algorithm to visualize tweets in a nuanced way. This is a refreshing vantage point for viewing COVID data to help visualize social media in a time of crisis.

Twitter Data vs. Regular Text

Tweets are highly distinguished from regular textual data given their 140 character-limit and ubiquitous use of slang, hashtags, and emojis. Furthermore, the use of embedded metadata such as hyper-links, GIFS, and images enables users to devour a higher volume of content much faster than ever before. Twitter's platform has effectively restructured the way in which we process data: rewarding users with the currency of retweets, favorites, and verification. This change calls for a few fundamentally different approaches to traditional NLP techniques to help us understand information transmission. We will elaborate on this idea throughout the course of this work.

Web scraping and Data Quality:

Twitter's API was accessed via tweepy. This library allows for the retrieval of 1500 tweets every 15 minutes. Thus, every time the search command was executed the data frame was changed. Data was simply filtered by the 'coronavirus' search word. This API searches against a sampling of tweets published within the last week. The selected time frame is arbitrary and can be manipulated but will play a key role in the analysis of our unsupervised learning algorithm.

The standard search API focuses on relevance and not completeness. Hence, some tweets have missing information. Complete web crawls can be obtained using a premium version of the search API which has a pay per use framework. For this reason, we are going to focus solely on the free version. **Figure 1** below shows our initial data frame after web-crawling twitter. Notice how all retweets (RT) end with "...". Once exported to CSV the "..." is replaced with an ellipsis character: Ä¶. The full version of the text can be accessed using search.api, however only 100 search results appear at a time. This is one of the *major* limitations of our study and although we compromise information loss, we elected to work with truncated tweets due to sheer volume.

	text	user	location
0	RT @mattyglesias: How do you keep people sane?\n\nOutside activity with reasonable precautions looks very safe. Open up the parks and beaches…	CeCe_23Spalding	Brooklyn, NY from Queens
1	RT WAXIEbuzz "RT GreenSeal: Our friends WAXIEbuzz have a #COVID19 prevention guide and info about their disinfectants. Check it out. \n\nhttps://t.co/fri6QXKU18"	HurleySupply	
2	'An Anvil Sitting on My Chest': What It's Like to Have Covid-19 https://t.co/84eslDocVG	Elle2Tha	Los Angeles, CA
3	RT @CNN: An elementary school teacher in Connecticut says she is taking care of a newborn baby boy after getting a phone call from a stude…	Pelleg1Gabriell	Tokyo and Nagoya, Japan
4	The magnitude of human loss from #COVID19 is massively visualized 👇\nU.S. Mortality: Death Certificates Listing Pneumonia, Influenza, and COVID-19 I CDC\nhttps://t.co/Wd56eeimY4 https://t.co/v0HVP9aSMB	mzkhalil	Cairo - Egypt
...	...	...	...
1495	RT @migov: Stay Home. Stay Safe. Save Lives. The state of Michigan (@migov) & @MichiganHHS report today, May 6, 2020, 657 new COVID-19 case…	MizJeniJonze	
1496	RT @V2019N: A genetic study of samples from more than 7,500 people infected with COVID-19 suggests the new coronavirus spread quickly aroun…	staywoketravel	Los Angeles, CA
1497	RT @Johnrashton47: 🦠 Remember that it's about twice this. If you want the truth check the Financial Times. Follow the money. 🦠 Coronavirus de…	chris_traynor	
1498	Had my Covid-19 antibody test today. Thanks @thesolutioniv for a great easy, quick, and clean experience. Looking forward to getting my results!!! #swissqualitysmile #covid19 #coronavirus #antibodytest… https://t.co/23a4uDHjwL	swissqsmile	Los Angeles, CA
1499	RT @BrianDeLay: In 1847, impoverished Choctaws scraped together sent $170 to send to victims of the Irish potato famine. Some Irish remembe…	ImpeachBDevos	Tell the truth and you shame Trump. - Shakespeare paraphrased

Figure 1: Original look at our data frame (emoticons/URLs/hashtags)

Instantly, we can see that tweets are rife with emojis, special characters, and hyperlinks. These will be removed in the preprocessing step. Also, note the prevalence of retweeted data. This speaks volumes to the ecosystem of COVID related data. At the click of a button, users can garner more attention than originally formatted tweets. Users tend to gravitate towards statistics regarding the virus and embedded metadata(videos/images/urls) which is logical during these times. **Figure 1** also suggests the sparsity of locational data (480 missing). Since locational data was collected from manually entered user bios, we saw a lot of variance. Here's an example of different variations of the same locations we obtained.

- (Liverpool, England, Liverpool UK | Scottsdale Arizona, Scottsdale AZ, USA | ¯_(ツ)_/¯ NYC | NY, NY USA)

Preprocessing and Feature Engineering:

We are not reinventing the wheel in terms of preprocessing Twitter data. Numerous NLP publications regarding Twitter preprocessing refer to boilerplate helper functions for cleaning tweets.[4] Combining the use of regular expressions to address the formatting styles of Covid-19 tweets, we have developed the following:

```
# clean tweet
def clean_tweet(tweet):
    # Remove hyperlinks
    tweet = re.sub("https?://[A-Za-z0-9./]*","",tweet)
    # Remove hashtags
    tweet = re.sub(r'#\w*', '', tweet)
    # Remove tickers
    tweet = re.sub(r'\$\w*', '', tweet)
    #@user -> at_user
    tweet = re.sub("@[\w]*","",tweet)
    # To lowercase
    tweet = tweet.lower()
    # Remove Punctuation and split 's, 't, 've with a space for filter
    tweet = re.sub(r'[' + punctuation.replace('@', '') + ']+', ' ', tweet)
    # Remove words: I, a , am, me (2 or less letters)
    tweet = re.sub(r'\b\w{1,2}\b', '', tweet)
    # Remove whitespace (including new line characters)
    tweet = re.sub(r'\s\s+', ' ', tweet)
    # Remove single space remaining at the front of the tweet.
    tweet = tweet.lstrip(' ')
    # Remove emojis or other. special characters
    tweet = ''.join(c for c in tweet if c <= '\uFFFF')
    return tweet  $
```

Figure 2: Cleaning the tweets to remove hyperlinks, special chars, emojis, etc.

Hyperlinks, special characters, and emojis were removed. Due to the white-space generated, white spaces and newline characters also needed to be filtered out. The data was lower-cased as a means of standardizing our data to aid in initial NLP preprocessing.

A Cursory Glance at Tweet Sentiment via Text Blob:

Instead of training a classifier (Naive Bayes/Gradient Boosting Machine) to identify the sentiment of tweets on a granular scale (disgust, anxiety, positive ...) which would require vast amounts of labeled data, we decided to settle on TextBlob to scale tweets based on polarity and subjectivity between 0 and 1. This could help our investigation of clustered data in terms of gauging some high-level sentimental overview of tweets that garnered the most attention. Although this is not the primary focus of this project, it's extensive research has inclined us to approach this problem from a point of topic modeling. The user can then form their own opinions on this topic and choose if they would like to hear more about it. Sentiment analysis would be a wonderful thing to geolocationally map over time, however, we feel that work done by the CoMune lab [2] (populational emotional state, bot/human classification, news reliability) has been a phenomenal academic venture that has already been covered in depth.

Hashtags: Hashtags allow the author of a tweet to distill the main points they are trying to ascertain. Twitter uses hashtags to rank content by keywords. [5] Although many empty fields are found following the extraction, they could provide a manual way for our group to intuitively label the gist of each tweet, instantly. For example, if a tweet contains [#COVID,#conspiracy] vs [#PrayforOurHealthProfessionals, "#Coronavirus] then we have two starkly different viewpoints (most likely) that we can distinguish with #hashtags over the content of the tweet itself. This was an interesting idea to wrestle with throughout this project. Also, it was observed that there were many variants of COVID 19 hashtags like #COVID_19, #COVID, #covid19, #Covid_19, #COVID19, and others.

Data Analysis:

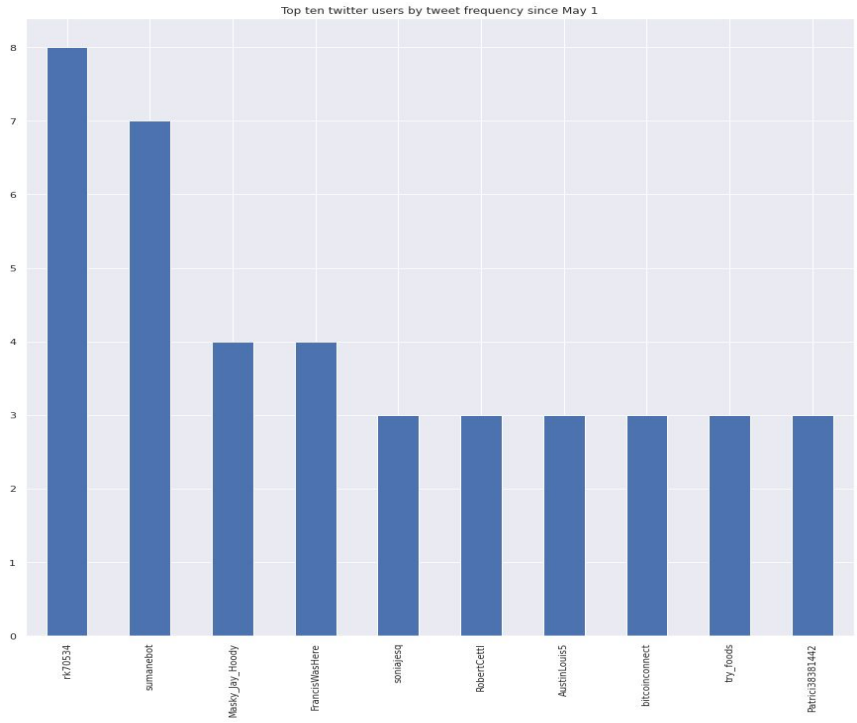

Figure 3: Top ten twitter users by frequency of tweets

Visualizing the top ten twitter users by frequency can help us in identifying potential social bots. If a user has generated 200 tweets within the past hour, then this would characterize possible bot activity. Websites such as Hoaxy have also created bot classifiers based on metadata such as number of followers, retweets, and activity coming from the account. According to the CoMune Lab, 60% of misinformation spread on Twitter's platform regarding the virus is due to the activity of social bots. [2] Although this is not the scope of this project, it is an idea we will keep in mind for understanding the platform.

Visualizing Unique Words in Our Data

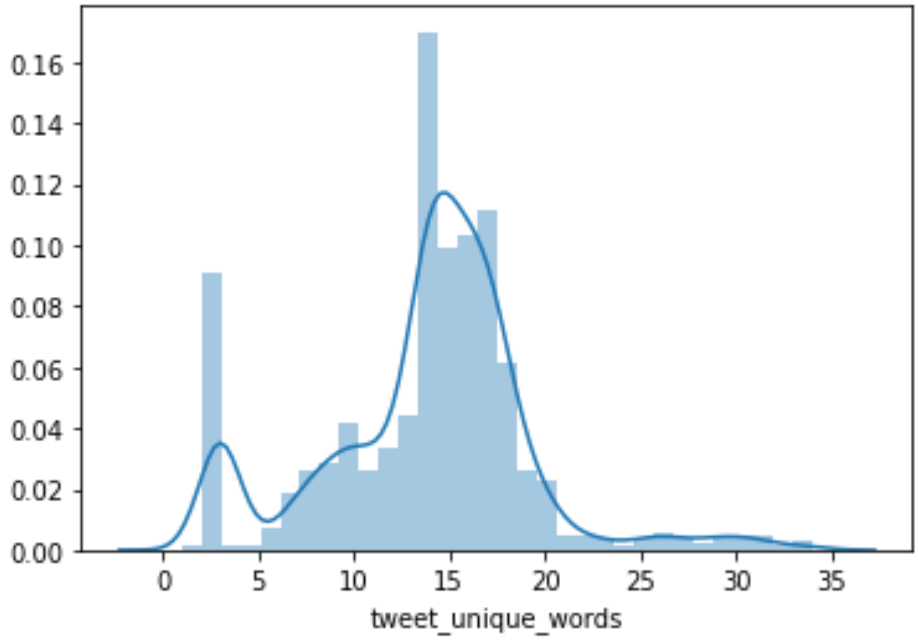

Figure 4: Plotting the number of unique words

Plotting the number of unique words in our data shows an approximate bimodal distribution split between 3 and 15 words per tweet. They are terse and to the point, generally. Following clustering, we will perhaps be able to identify the reason for this bimodal distribution. We suspect that it has been generated from a popular retweet that has been circulated in our time frame.

Word Frequency Distribution:

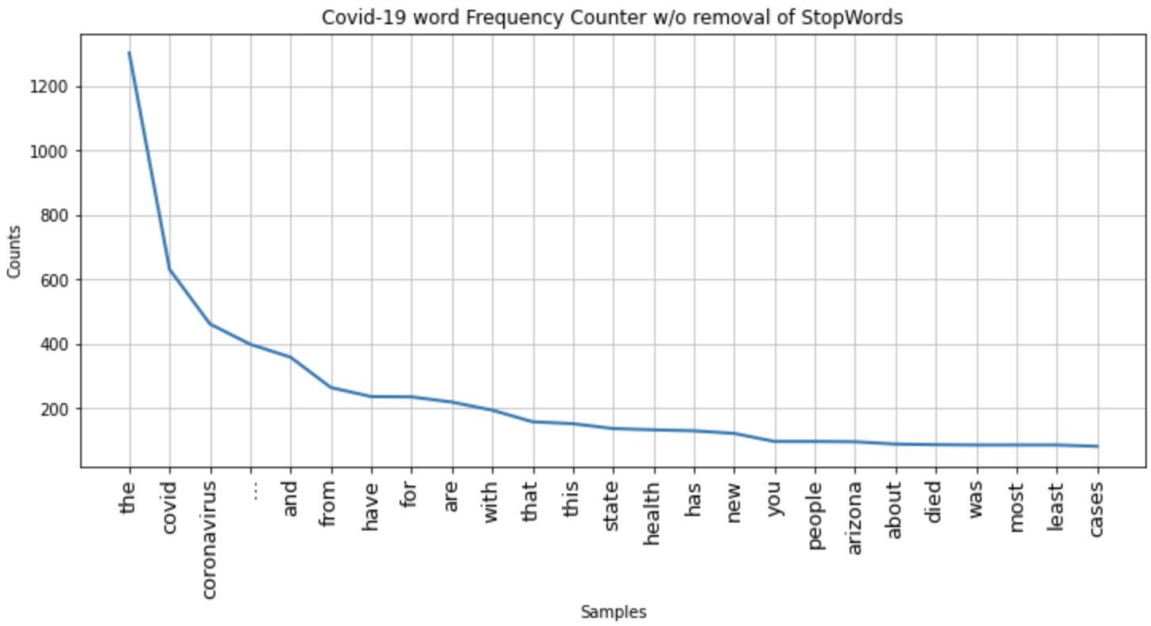

Figure 5: Word-frequency distribution

A Frequency distribution of the data shows a preponderance of stopwords and coronavirus variations that need to be removed. There are certain words we would like to remain in our analysis such as Arizona, died, and state. These are topic-oriented, contextual clues that could help with visualization. To finish our preprocessing, we will discuss spaCy for stopword removal and tokenization.

SpaCy

SpaCy is a natural language processing toolkit, born in 2014 with industrial-level speed and versatility. Cython is widely used in spaCy to improve the performance of related modules. Contrasting to NLTK, spaCy is an objective-based language processing toolkit which is less time-consuming in dealing with short messages compared with NLTK. SpaCy was useful for quickly preparing the rest of our data for analysis. En_core_web_sm is a pre-trained spaCy

model that contains GloVe vectors trained on Common Crawl. [4] This was useful in quickly tokenizing and lemmatizing tweets to assign context-specific token vectors to the data. It can also be quite useful for assigning POS tags, dependency parsers, and named entities to datasets. SpaCy can make many of the time-consuming decisions for you that NLTK cannot afford in some cases, which is why it was chosen.

Custom Stop Word removal:

We filtered out all words that have to do with coronavirus in order to understand the discussion surrounding it. In doing so, we lose some of the contexts of word features, however, we consider the context in order to graphically represent our data. Following stop word removal we used SpaCy to lemmatize. Some final thoughts on the frequency distribution are bulleted below.

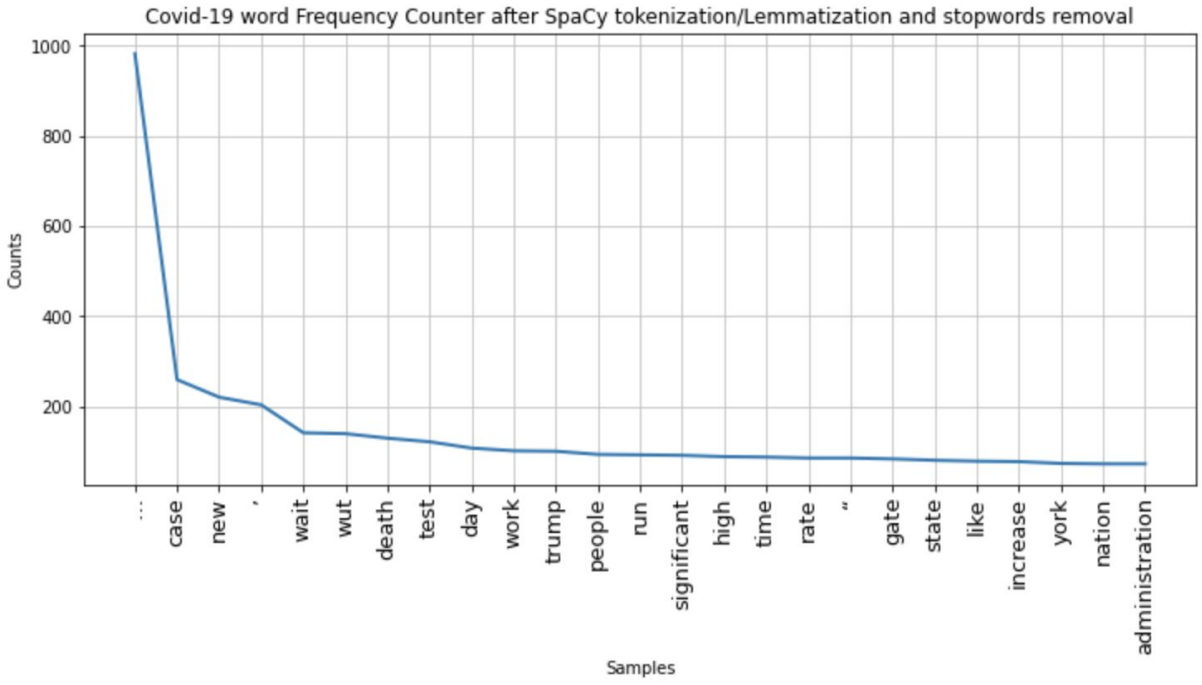

Figure 6: Word frequency after lemmatization/tokenization and stop word removal using Spacy

- Hasn't removed ellipsis "..."
- Death, cases, and tests are very common keywords
- " ' " certain special characters could use further cleaning but serve fine for this report
- Lots of government-related words that occurred frequently throughout tweets

Machine Learning Approaches:

After the preprocessing step we decided to move into machine learning approaches for our unsupervised learning algorithm to visualize the transmission of information. We begin by vectorizing the text data.

Vectorization (TF-IDF/BOW)

To find a low-dimensional representation of our tweets, we elected to use Tf-IDf in conjunction with t-SNE. Before diving in, let us take a look at a simpler vectorization approach. In general, text documents shouldn't be vectorized into a sparse matrix due to the possibility of high computational loss. Additionally, most linguistic information should remain after this process. For this, we could possibly implement a word embedding technique (word2vec, GloVe).

Bag of Words

If the same word in a sentence exists in a vocabulary, then a feature vector can be created. If that particular word is present in both places (vocab + sentence) then an entry of 1 is made in that place and a zero occurs if otherwise. For example, if we had a series of tweets that said:

1 "The coronavirus pandemic is bad"

2 "The coronavirus pandemic is horrible"

3 "The coronavirus pandemic is not a hoax"

Then we could create a vocabulary

[1] [2] [3] [4] [5] [6] [7] [8] [9]

The Coronavirus pandemic is bad horrible not a hoax

The tweet vectors could then be represented as

[1,1,1,1,1,0,0,0]

[1,1,1,1,0,1,0,0]

[1,1,1,0,0,0,1,1].

The disadvantage of this technique is that new vocabulary increases vector length. This results in a sparse matrix which, as previously mentioned, can lead to problems in low dimensional representations of our feature vector. Additionally, grammar is not accounted for which is why

we lose contextual information in our sentence (see tweets **2 and 3**). This is where Term-Frequency - Inverse Document Frequency can step in. [6]

TF-IDF

Another issue with the Bag Of Words approach is that each word has a similar significance considering its weight amongst all documents. To reckon with this, the TF-IDF approach tends to make sure that more weight should be given to words or terms which are more frequent throughout the document(where the frequency of *these* words or the term is comparatively lower). The term frequency can thus be calculated as:

$$Term\ Frequency\ (TF) = \frac{Number\ of\ times\ the\ word\ or\ the\ term\ appears\ in\ the\ document}{Total\ number\ of\ words\ or\ terms\ in\ the\ document}$$

The Inverse Document Frequency or IDF for a particular word will be the total number of documents, divided by only the documents in which that specific word is found. To reduce the effect of this division, we calculate the logarithm of its ratio. The Inverse Document Frequency or IDF can thus be calculated as:

$$Inverse\ Document\ Frequency\ (IDF) = \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ which\ have\ that\ word}\right)$$

In summation TF-IDF is a product of the two terms: TF and IDF.

Following this, our vectorized tweets were ready to have PCA applied.

Dimensionality reduction via PCA

Principal Component Analysis was used to reduce the dimensions of our Tf-Idf vectorized features while maintaining 90% variance. This was lowered from the default value (95%) in order to obtain $\sim$ 290 features (opposed to 379). Our data originally contained 1195 features (very high), so PCA didn't compromise the loss of too much information, luckily. 290 features is still a relatively large number of dimensions for t-SNE to process however this method was still able to remove some noisy outliers from our vectorized tweets to make the clustering process more digestible for k-means. [7]

K-Means Clustering

This is an unsupervised learning algorithm that will use the elbow curve method to plot distortion vs. the number of clusters, K. This allows data to be naturally segregated solely based on our vectorized text. The algorithm is distance-based and tries to minimize the SSE (intra or within-cluster variation) to produce coherent clusters. For labeled-tweet distinction, k-means will be run on the vectorized text. [8]

Following clustering, we could measure things such as precision, recall, and accuracy of our model. K-Means is an unsupervised learning algorithm so the model will not distinguish between what the true validity of our class labels actually are. The whole point of this experiment is to work without labeled data to view information transmission of tweets purely on a textual basis. Therefore, instead of validating our model on things such as LDA key-word topic modeling, we created a Bokeh plot that allows us to visually assess the coherency of clusters based on content.

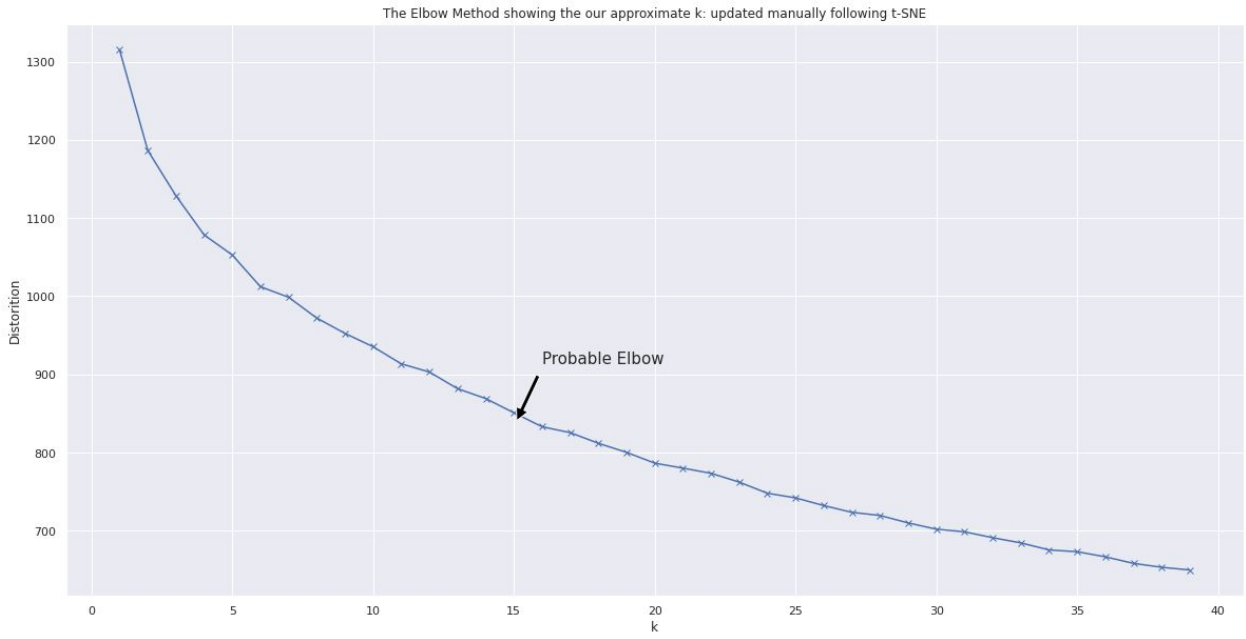

Figure 7: The Elbow Method showing our approximate k: updated manually following t-SNE

The elbow curve didn't yield desirable results. We don't see an extremely clear inflection point which is problematic. Instead of extensively grid searching our model, we decided to plot out the data using t-SNE to intuitively view the optimal number of clusters. This elbow curve is

jagged and suggests that we may want to train the model on a different vectorization technique in the future, such as Word2Vec.

The above figure proposes that K values are optimized between 15-20 clusters. Following 20 clusters, the decrease in distortion is not as significant. We will use 17 for this model. Given that this is an unsupervised algorithm, we may have multiple optimal clusters based on when the data was scraped from Twitter [May,1]. As mentioned, we proceeded with the process and decided to see how t-SNE was handling our optimal number of clusters.

t-Stochastic-Neighbor Embedding was then imported from the sklearn manifold package in order to visually map our 290 features onto a 2-dimensional plane. This is done by minimizing the KL divergence between joint probabilities of the low-dimensional (1500, 290) and high-dimensional embedding (1500,1195). t-SNE's cost function will keep most of the context-based word vectors that we want without being too large to process. t-SNE will use the original feature vector X(1500,1195) that was obtained via tf-idf on the preprocessed text. Following this, we have a visual representation of our tweets!

$$\mathrm{KL}\left(P \parallel Q\right) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Minimization of KL divergence with respect to our observations. It is optimized via gradient descent and the result of this is a map that shows similarities between our high-dimensional features. [9]

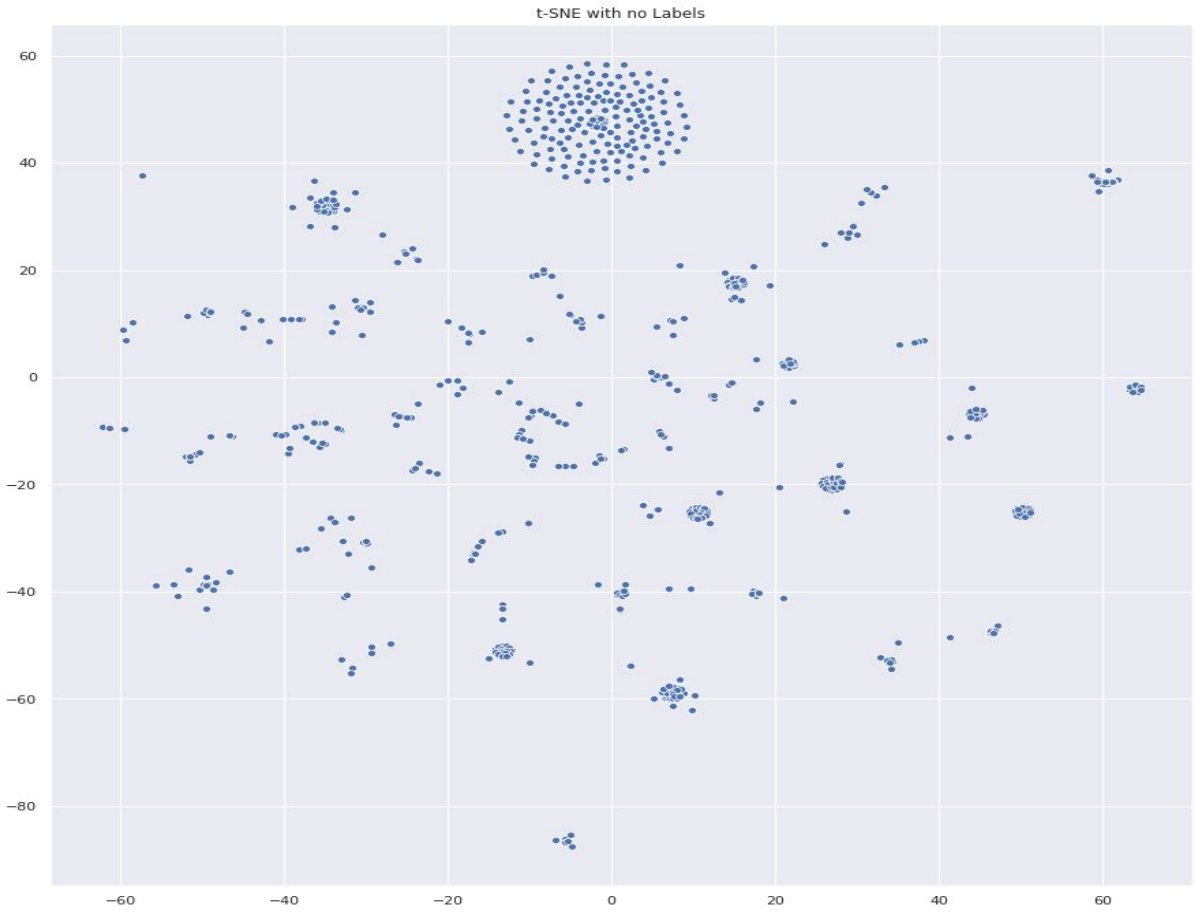

Figure 8: t-SNE with no labels

The goal of this technique is to reduce our high dimensional feature vector to 2 dimensions. We can plot the tweets themselves by using these dimensions as x, y coordinates. In other words, similar tweets will be closer together, and dissimilar ones farther apart. This algorithm has two hyperparameters that we considered for tuning.

Perplexity: float, optional (default: 30) (number of nearest neighbors: larger number of observations usually requires a higher value > 30)

Perplexity was iteratively tuned and reduced to 10 in order to account for the smaller number of observations in our dataset.

Earl_exaggeration: float, option (default 12):

Controls coherency of clusters in the space in which they are embedded. It also determines the amount of space between them. Higher values correspond to more space between clusters within the embedded area. We still wanted some leeway between t-SNE's representation and the coherency of clustered tweets, so early_exaggeration was slightly reduced from 12 to 10. This resulted in a more balanced model overall that allowed for the possibility of dissimilarity between tweets. In other words, RT information would be extremely close in space, but unique tweets would have the opportunity to be more spread out.

t-SNE with k-Means labels

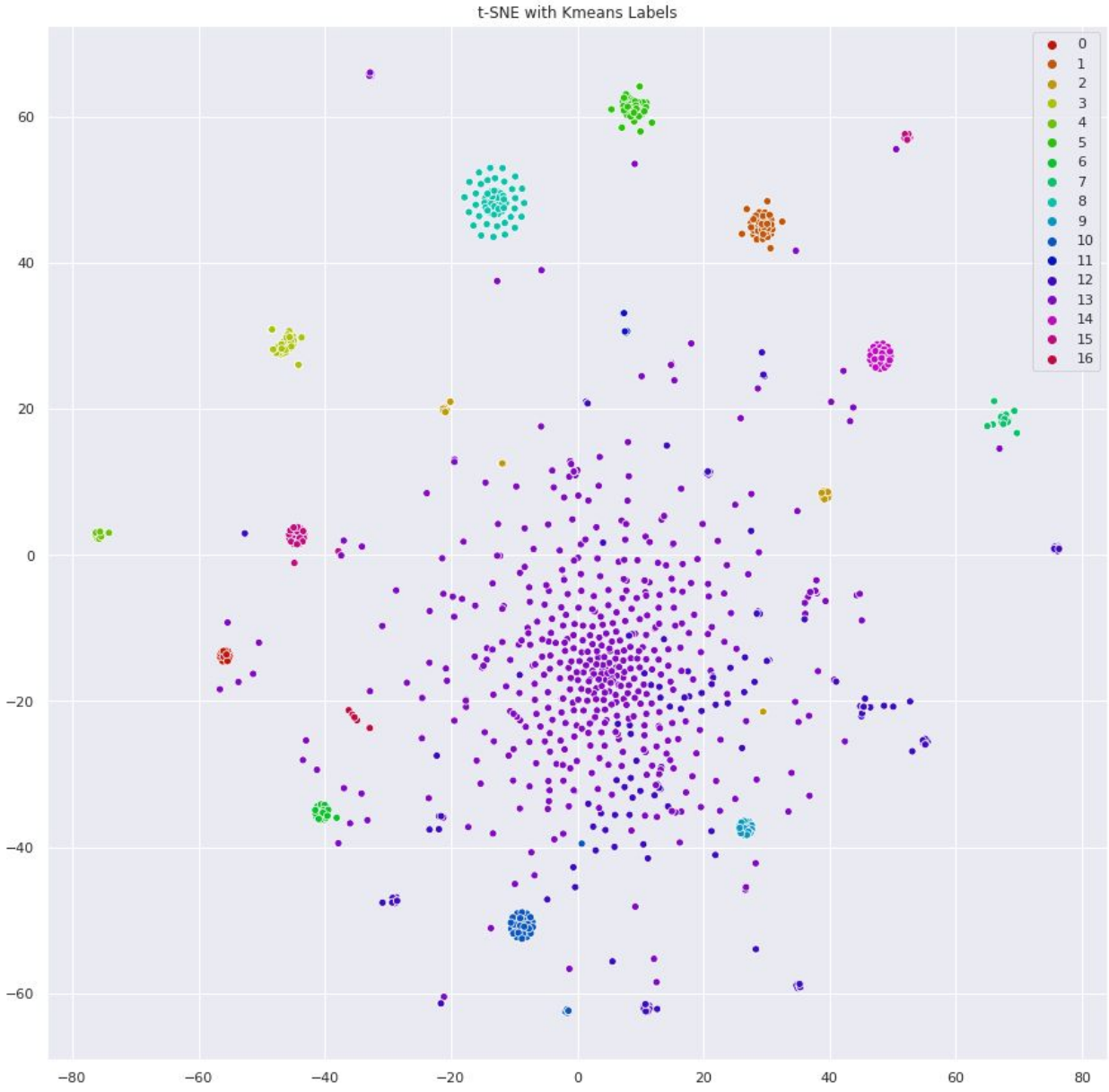

Figure 9: t-SNE with k-means labels

There are approximately 17 natural clusters that we can instantly recognize. This value was used to update our k-means labels to compensate for the inadequacy of our elbow-curve method. t-SNE did well in terms of dimensionality reduction, but labeled data generated by

k-means could help us in opening up the hood and examining our tweets through a content-based lens. Clusters developed by k-means aided in the generation of labels to help visually separate tweets containing different feature vectors. Validation, in this experiment, is then intuitive and manual. t-SNE clusters could be compared to the color of k-means labels to search for any discrepancies. For example, there are some deviations in the label generated by k-means and tweets that are spread out in **Figure 9** (see orange points w/ label = 2). This occurs because the labels and the points do not have proportionate resemblance between themselves and the higher dimensional data. This most likely signifies the computational loss we see from tf-Idf vectorization. This isn't all bad considering we simply want popular retweets to have a high concentration of coherently clustered and consistently labeled points. Points with more spread probably correspond, then, to posts that contain most of the same COVID-related keywords (death, cases, tests, trump) but fundamentally different content.

Discussion

By scraping a few thousand COVID-related tweets within a specified time range we hoped to give the reader an idea of information transmission (mostly retweets). Visualizing the spread of articles/rumors/stories/opinions during this time of crisis can serve the purpose of noise reduction which could ease some of the anxiety we get from social media. The modeling process was inspired by techniques used in the COVID-19 Literature Clustering publication, authored by Eren, E. Maksim. Solovyev, Nick. Nicholas, and Charles. Raff, Edward [3] The primary difference in their work is its application, which clusters academic literature by topic using LDA to help health professionals keep up on field-specific information related to the virus. Their idea being that by clustering similar research articles they could help simplify the search for related publications.

While some of the modeling and preprocessing techniques were similar to our approach, we have modified and repurposed this motivation to understand how Twitter data can be clustered within a given time frame. The clustering of tweets can be better represented in the form of a labeled plot. t-SNE was used to intuitively visualize the optimal number of clusters in our data, which is cheating the model to a certain extent, however, upon further review the dimensionality reduction steps seemed to sufficiently cluster together the information in a way

that was appropriate for visualization. Following PCA and dimensionality reduction, our data was mapped from 1195 to 290 to 2 features respectively. Although t-SNE had an extremely high number of features to map on the coordinate plane, it didn't seem to compromise too much information in our data.

To reiterate the limitations of our work, the free version of tweepy only allowed us to scrape 1500 tweets every 15 minutes or so, which is a very small number of observations for a robust machine-learning algorithm to be trained on. Furthermore, retweets were displayed in a truncated mode that was followed by an ellipsis character. Additionally, it was clear that this form of sharing was extremely popular in our data. This gave a disproportionate amount of weight to retweets given that they contained "..." even after preprocessing. In the future, more extensive data cleaning could be carried out on this work to give us a better idea of the variations and additional commentary people use for RTs. The use of tf-Idf for vectorization yielded a sparse matrix which is known to have a very high computational loss in the field of textual clustering. Perhaps sentiment analysis could've been investigated more in depth after dimensionality reduction took place but our plates were still full throughout this study. Finally, Twitter data is uniquely distinguished from regular text given its ubiquitous use of hashtags, slang, and emojis. Perhaps in the future, we could utilize a tool such as VADER as a means of processing these emojis to extract more meaningful information from the text. Even in light of all of these shortcomings, our exploration still yielded some pretty remarkable results that fascinated us time and time again. We will now open up the hood of our model and investigate our interactive Bokeh plot to make connections between the apparent clusters generated from t-SNE, the k-means labels and the tweets themselves.

Bokeh plot

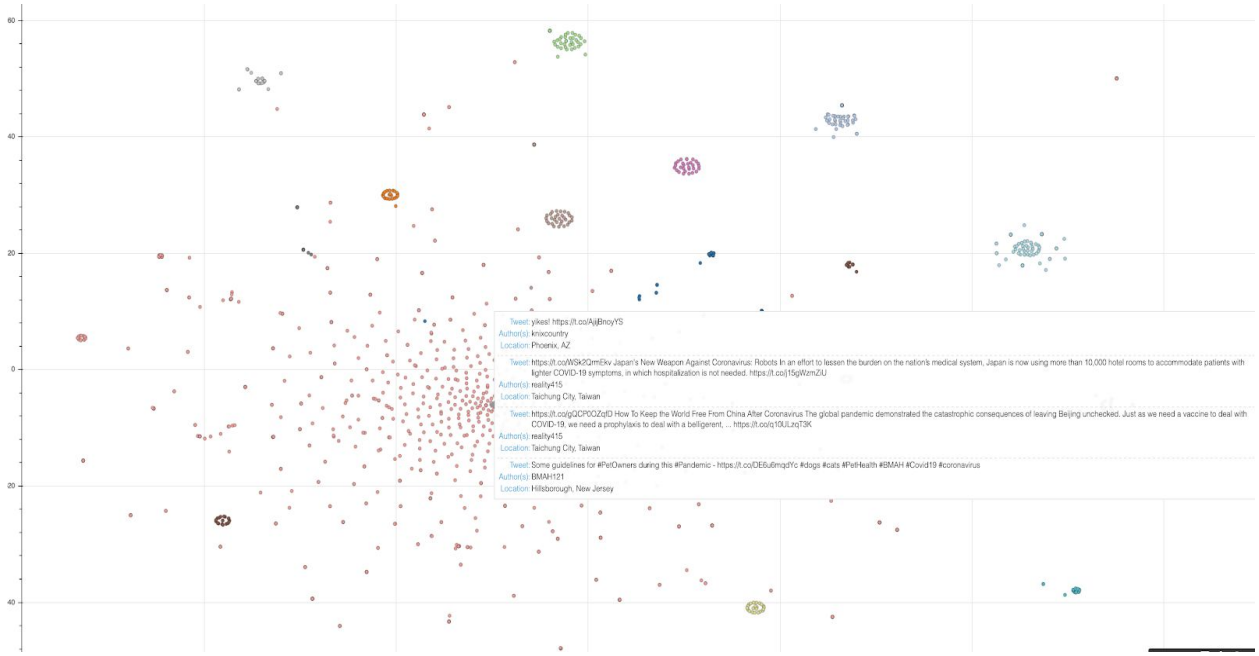

Figure 10: Bokeh app to show clusters

Following t-SNE, we generated an interactive Bokeh plot to investigate our data. Note that this process has been refined and modified from the Covid-19 Literature Clustering challenge. Only slight changes such as the removal of the slider (# of clusters) and keywords generated from LDA were made. [3]

In the cluster highlighted in **Figure 10**, there is a wide variety of similar tweets combined together. With this combination, we can easily determine and analyze the tweets which are

related to Coronavirus. This dashboard will remove the need to manually scroll through tweets and provide the user with an interactive template.

As observed above, there are various coherent and incoherent structures being formed by the tweets. An observation about the coherent clusters' formation is that they occur due to the retweet of the same twitter post without any additional content. While in the case of incoherent clusters' formation, there are retweets of the same twitter post but here the retweet is attached with some form of additional information such as text, images, or other embedded metadata.

Some tweets contain a larger length of text due to which some of the text data inside the retweet is ignored. If we had access to the full 'extended' form of tweets, we would expect clusters to be slightly less coherent or natural, overall. Our data contains tweets from May 1st to May 3rd, 2020. If we select a different date, we would procure a drastically different data frame that would yield a completely different visualization. That being said, after running the code 40-50 times, we still observed similar patterns in the transmission of information. Most of these tweets tend to be politically focused, which is consistent with our word frequency distribution seen in **Figure 6**. Trump has obviously dominated the social spotlight when it comes to COVID. When a tweet is posted by a major, verified source such as a celebrity or a politician, they have better reach and therefore have a larger cluster around them due to an increased number of retweets from their original post. The final dashboard application is saved in HTML format. This file can be accessed by opening the HTML in any supported browser on your device. The interactive bokeh plot will be hosted on Heroku and will soon be available on Github. We hope you enjoyed this exploration as much as we enjoyed making it.

References:

[1]: “Coronavirus Death Toll.” *Worldometer*, 2020, www.worldometers.info/coronavirus/coronavirus-death-toll/.

[2] R. Gallotti, N. Castaldo, F. Valle, P. Sacco and M. De Domenico, COVID19 Infodemics Observatory (2020). DOI: 10.17605/OSF.IO/N6UPX

[3] Eren, E. Maksim. Solovyev, Nick. Nicholas, Charles. Raff, Edward, COVID-19 Literature Clustering, 2020,April, location = University of Maryland Baltimore County (UMBC), Baltimore, MD, USA, https://github.com/MaksimEkin/COVID19-Literature-Clustering

[4] “All You Need to Know about Text Preprocessing for NLP and Machine Learning.” *KDnuggets*, May 2019, www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html.

[5] “How to Use Hashtags.” *Twitter*, Twitter, help.twitter.com/en/using-twitter/how-to-use-hashtags.

[6] Huilgol, Purva, et al. “Quick Introduction to Bag-of-Words (BoW) and TF-IDF for Creating Features from Text.” *Analytics Vidhya*, 29 Feb. 2020, www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/.

[7] “Sklearn.decomposition.PCA¶.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[8] “Clustering Text Documents Using k-Means¶.” *Scikit*, scikit-learn.org/stable/auto_examples/text/plot_document_clustering.html.

[9] “T-Distributed Stochastic Neighbor Embedding.” *Wikipedia*, Wikimedia Foundation, 16 Apr. 2020, en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding.