Assignment 5: Tweet Clustering

Ryan Fischbach
Dr. Khuri

November 13th, 2020

1 The Dataset

The SNScrape library was used in a tweet scraping script to collect tweets. 1,000 tweets from 3 categories were compiled: 1,000 tweets with the keyword 'election', 1,000 with the keyword 'covid', and 1,000 with the keyword 'usopen'. All three were written to their own CSV files and then preprocessed with another python script. This yielded 3,022 tweets in total across the 3 topics, containing 3 attributes: 'id', 'date', and 'tweet'.

2 Preprocessing the Data

To enable the best clustering on the text data, several steps were taken.

1. Duplicates samples were removed because the tweets had an associated ID indicating if they were the same.
2. The 'id' and 'date' columns were removed to just have the 'tweet' and corresponding text data for each sample be clustered.
3. The DataFrames for each topic were combined and cast as a string to produce one larger corpus of text.
4. Punctuation and numbers were removed, text was made lowercase, the text was tokenized, stop words (common words) were removed because they don't provide meaning, stemming was performed via Porter Stemming, and tokenized words were rejoined to sentences.
5. Tfidf was used to encode the words into a matrix via ngrams size 1, yielding an array representation of the text of size (number of samples, number of attributes).

3 Text Clustering

SKLearn's implementation of KMeans and my own implementation of KMeans were used to cluster the tweets. Both use euclidean distance, so tfidf was the best choice to represent the text data. If a binary representation was created, euclidean distance would likely not be able to identify meaningful clusters from the distances provided from a sparse binary matrix.

3.1 SKLearn KMeans Clustering

To cluster the tweets using KMeans, k = 3 was used because the number of topics was known. This also facilitated the true labels being used to judge performance.

Principal Component Analysis was used to reduce the dimensionality of the dataset to allow for the clustering to be visualized in 2 dimensions.

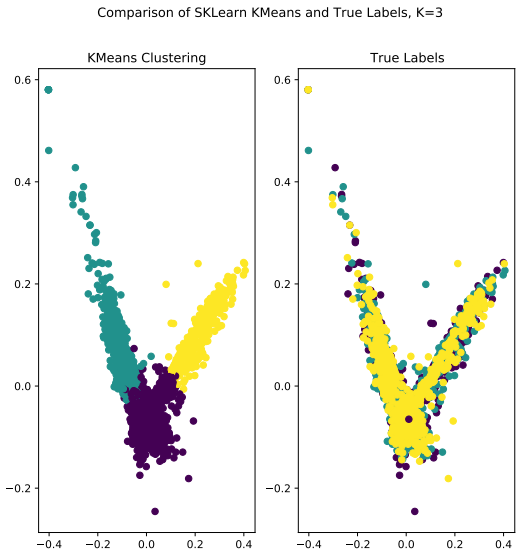

Figure 1: *KMean's clusters and the true labels were put side by side to determine the performance of clustering. SKLearn's KMeans clustering is on the left, with clear clusters being identified from the data. The true labels are on the right, with no clear pattern in the data. Classes are scattered together.*

The poor performance of KMeans is likely due to the encoding of tweets not containing enough information to be able to distinguish the topics from each other. ngrams = 1 was used to preprocess the data, so larger ngrams may provide better performance because the order of words is stored.

ngrams from size 1 to 5 was used next, creating 173,378 attributes compared to 11,456 with ngrams size 1. This encoded more information about the tweets but resulted in much slower clustering time.

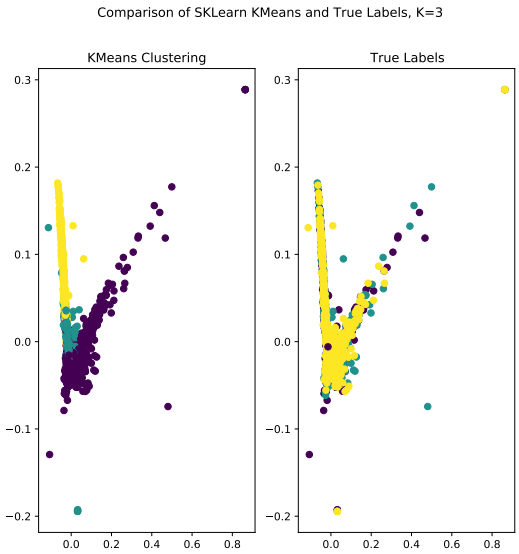

Figure 2: *KMean's clusters and the true labels were put side by side to determine the performance of clustering with a higher dimensional dataset via ngrams size 1 to 5. SKLearn's KMeans clustering is on the left, with clear clusters being identified from the data. The true labels are on the right, with no clear pattern in the data. Classes are scattered together.*

Once again, the true labels being visualized seem to imply that there is no clear patterns in the tweets for each topic. Thus, a better way to encode the data is needed to generate better performance.

3.2 From Scratch KMeans Clustering

A KMeans implementation from scratch was also used to cluster the data.

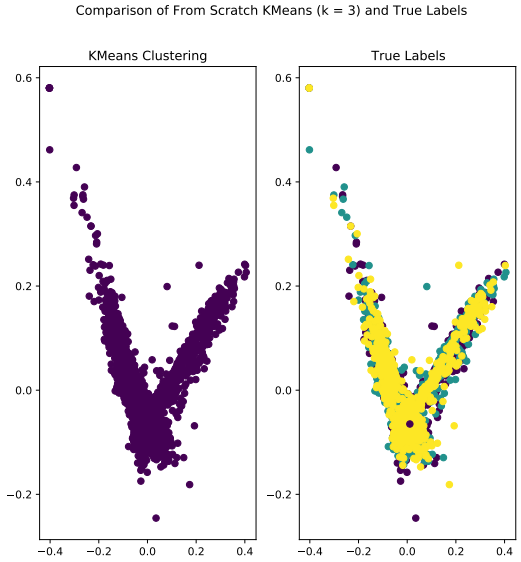

Figure 3: *KMean's clusters and the true labels were put side by side to determine the performance of clustering using ngrams size 1. From Scratch KMeans clustering is on the left, with all points having the same label. The true labels are on the right, with no clear pattern in the data. The From Scratch KMeans has an error in its implementation, so it provides no clear ability to cluster the data. Thus, it can't indicate any useful information to us.*

4 Using a Linear SVM Classifier to Classify Tweets

SKLearn's Linear SVM Classifier utilizing Stochastic Gradient Descent was tried next. The preprocessed tweets were split into a 60/40 training test split.

The classifer yielded a performance of 33.99 percent with the following performance:

class	precision	recall	f1-score	support
0	0.36	0.4	0.38	404
1	0.33	0.19	0.24	416
2	0.33	0.44	0.38	389

The Linear SVM's performance once again suggests that the data with ngram = 1 did not give the classifier enough information to be able to distinguish between the classes. 33 percent accuracy is the same accuracy we would expect from a naive classifier that assigns only 1 class label. Thus, this performance is very sub par.

ngrams size 1 to 5 was tried next with the classifier and yielded an accuracy of 33.88 percent and the following metrics:

class	precision	recall	f1-score	support
0	0.36	0.39	0.37	404
1	0.34	0.23	0.27	416
2	0.32	0.40	0.35	389

From these results, we can once again see that the classifier with more attributes per sample could not improve its performance. Thus, the tweets might not contain enough of a difference in words to be able to distinguish between topics. This could be due to crossover between the 'elections' and 'covid' topic, where they are intertwined in real life.

5 Citations

[1] SNScrape. 2020. https://github.com/JustAnotherArchivist/snscrape
[2] Tan et al. 2005. Introduction to Data Mining, 88 pages.
[3] Numpy. 2020. https://numpy.org
[4] Pandas. 2020. https://pandas.pydata.org
[5] Matplotlib. 2020. https://matplotlib.org
[6] Seaborn. 2020. https://seaborn.pydata.org
[7] Scikit learn. 2020. https://scikit-learn.org/stable/
[8] Natalia Khuri. 2020. lecture-7, 80 pages.
[9] NTLK. 2020. https://www.nltk.org