JSC270 H1 S LEC0101 **Runshi Yang** (1006734223)
Assignment 4 (*July 25, 2022*) **Jisu Qian** (1006699869)

Link to Github & Link to Google Colab Notebook (Click to open)

Work Breakdown:

All the work was done jointly.

Part I: Sentiment Analysis with a Twitter Dataset

A) The balance between the three classes of the training data is:

Sentiment	Proportion
0	0.374159
1	0.187407
2	0.438434

B) Tokenize the tweets. Finished in Google Colab Notebook.

C) Remove all URL tokens from each of the observations. Finished in Google Colab Notebook.

D) Remove all punctuation and special characters. Also, convert all tokens to lowercase only. Finished in Google Colab Notebook.

We may want to keep punctuation such as the exclamation mark (!) when we not only want to distinguish the tweet is positive, negative, or neutral but also want analyze the intensity of the sentiment of those tweets. This is because he exclamation mark is used to indicate strong feelings or to show emphasis. For example, there is a big difference between "I'm so happy." and "I'm so happy!!!".

E) We use the Porter stemmer to stem tokens. Finished in Google Colab Notebook.

F) Remove stopwords. Finished in Google Colab Notebook.

G) Convert lists of words into vectors of word counts. Finished in Google Colab Notebook.

We set the max_feature parameter of the `CountVectorizer()` function to be 3500, which means the CountVectorizer will select the 3500 most common words in the data. So the length of our vocabulary is 3500.

At position [113] of our Google Colab Notebook, we found that the total number of vocabulary in all tweets is 78963. But we choose 3500 as the vocabulary length since we tried to set it to 1500, 2000, 3500, 4000 and 4500, and found that the

accuracy of the test is highest when the max_feature parameter is set to 3500. (which means the model might overfit when the length of our vocabulary is greater than 3500)

H) Fit a Naive Bayes model to data (using count vectors). Finished in Google Colab Notebook.

$$\text{training accuracy} = 0.7277 \qquad \text{test accuracy} = 0.6919$$

The 5 most probable words in each class, along with their counts:

Sentiment	Word	Count
0	coronaviru	6703
	covid19	4862
	price	4332
	food	3622
	thi	3206
1	coronaviru	3792
	covid19	2751
	store	1581
	supermarket	1435
	price	1361
2	coronaviru	7466
	covid19	6001
	store	3895
	thi	3770
	price	3322

I) Based on what we learned in class, we can not fit an ROC curve in this scenario. To fit the curve, we need to define what are "positive" and "negative", however now we have 3 classes instead of 2. But we can draw a "one class versus the other two classes ROC curve" for each class, only this class is defined as "positive" and the other two are defined as "neigative".

J) Redo parts G-H using TF-IDF vectors instead of count vectors. Finished in Google Colab Notebook.

$$\text{training accuracy} = 0.7016 \qquad \text{test accuracy} = 0.6769$$

We found that the training and testing accuracy for TF-IDF vectors were slightly lower than the accuracy using count vectors.

K) Redo parts E-H using TF-IDF vectors instead of count vectors. This time use lemmatization instead of stemming. Finished in Google Colab Notebook.

$$\text{training accuracy} = 0.7193 \qquad \text{test accuracy} = 0.6759$$

The training accuracy using TF-IDF vectors and lemmatization were higher than the training accuracy using TF-IDF vectors and stemming. But the testing accuracy is a little lower than the testing accuracy using TF-IDF vectors and stemming. (Notice that to maximize the accuracy, the `max_features` we use in these three methods are not the same.)

Bonus Naive bayes is a Generative model since it does not try to draw boundaries in the data space, instead, it tries to model how data is placed throughout the space. In other words, it learns the joint probability distribution $p(x, y)$, while a discriminative model learns the conditional probability distribution $p(y|x)$.

Part II: Having fun with NLP using the Twitter API

(I) Problem description and motivation

In this assignment, we explored a new model for NLP – Latent Dirichlet Allocation. We are curious about whether we can use unsupervised learning methods to discover some natural topics given a large quantity of tweets. We choose this topic since in text mining, we often have collections of documents, but we might not sure what we're looking for (there is no predefined groups or topics), so we'd like to divide them into natural groups so that we can understand them separately. And we think it would also be a useful feature for twitter if, after a user enters a keyword with a broad meaning, potential topics associated with it can be presented to him for further narrowing the search.

(II) Data description

We used tweepy to extract tweet data and used "oil" as our search word. We filtered out all retweets to avoid duplicates, since we are afraid that some very popular tweets would make up one topic simply because of hundreds or thousands of retweets. We set a parameter of tweepy.Cursor function, `tweet_mode`, to "extended" and extract the full text so that the tweet text returned would not be truncated to 140 characters. Thus we could understand the whole idea of tweet better in this way. The number of observations is 6360 and we only made use of one feature–the "full_text" attribute of tweets. We saw other topic modelling researches using twitter data and most of them also use LDA since this is the standard approach. But most of them does not analyze the performance

of their model rigorously.

One of the limitations of our data is tweets are very casual since twitter is a social networking site, people often use shortcuts and self-created words and there could be fewer consistent language patterns. This makes it more difficult for models to detect latent similarities. Another potential limitation of our data is that the users of Twitter are mostly from English-speaking countries. If we want to get a sense of what people in the world are talking about, the data could be biased. One strength of our data is that it is up-to-date and we can get an understanding of hot topics that people care about very recently. Another strength is that our data could be obtained in big amount in a relatively quick and cost-effective way so that we can repeat this process relatively easy and use the discovery to make a difference.

(III) Exploratory data analysis

In the preprocessing step, we tokenized the tweets, removed URLs, punctuations, special characters and stopwords. All tokens are changed into lower-case. Lemmatization is applied to the tokens. After all these preprocessing steps, we obtained a vocabulary of size 22266.

We carefully looked into the tokens column and discovered some weird patterns. There are many single letter tokens such as "s" and "t". We traced back to original tweets and find some explanations for this phenomenon. Most of the 's's are produced by possessive case "'s" and it is not in the stop_word so they are not removed. It was also verified that occurrences of "s" is too high among all tokens when we displayed the top words later and we don't think that this "s" makes sense, so we added it to our stop_word list. Many "u" are produced

by "U.S." and U.S. is mentioned a lot in worldwide issues. It got separated during the process of tokenization. "U" is used as an abbreviation for you as well. That is why they occur a lot.

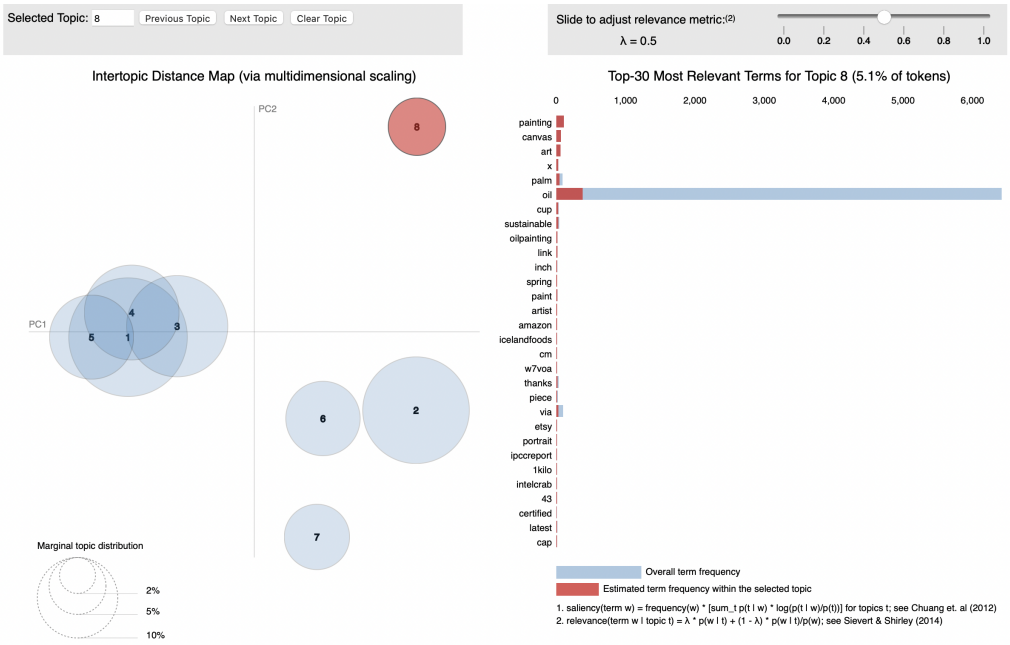
(IV) **Machine learning model**

We implemented Latent Dirichlet allocation model to help solve our problem. It is an unsupervised learning model. In general, it says in what percentage each document talks about each topic. There are two basic assumptions for LDA: Every document is a mixture of topics and Every topic is a mixture of words. So, it uses two Dirichlet distributions as priors and multinomial distribution's pmf as likelihood function and calculates the posteriors based on our data. It iteratively updates the priors and posteriors and gives more and more accurate results. The output of LDA is a probability distribution over all topics for each document and a probability distribution over all words for each topic. We use Scikit-learn's CountVectorizer to convert the lists of tokens into vectors of word counts and fed the data into LDA model. We assign each tweet the topic with the dominant probability.

One baseline model is K-means clustering. The difference is k-means clustering would directly assign each observation, here, each tweet, to one and only one cluster. LDA, however, gives the probability distribution of the document belonging to each topic, which could give more realistic results. That's why we choose LDA for our question. In fact, we also have a try with K-means clustering, the result is decent but not very interpretable for every topic. We would evaluate the model using coherence score, which measures the similarity of the top N words within each topic.

(V) Results and Conclusions

The result of our study is presented in an interactive way and the following is a screen shot of the visualization.



We choose 8 to be the number of topics for our model, so it splits all the tweets into 8 groups. Each of them is represented by one circle on the left. The larger the size of the circle, the more tweets are allocated in the topic. And the closer the two circles are, the more relevant the two topics can be. The blue bars on the right represents the number of times a token occurs in all the tweets we collected. And if we select a topic, the red bars will show the number of times a token occurs in the selected topic.

For group 1, 3, 4 and 5, we can see that the more representative tokens are "russian", "profit", "price"... so we believe that these are about the news like Russia-Ukraine conflict and oil price. Group 8 includes tokens like "painting",

"canvas", "art", ... so it's pretty clear this topic is about oil painting. And there are many tokens about food such as "olive", "stew", "tomato"... in group 6 and 2, so we can imply that they are about cooking oil. Finally, group 7 consists of many strange tokens, for example, "juststop_oil", "xrebellionuk", "talkradio" and we searched for the original tweets that produce them and noticed that most of them are official twitter accounts so they are often mentioned.

Since our model is unsupervised, we can not calculate the significance. But based on the fact that it is very easy to figure out what the topics can be, we believe that our model performs relatively well.

We plotted the coherence score against the number of topics. We could see that 3 has a relatively high coherence score of 0.49. So, it could be a good choice for the number of topics and it could be partly verified from our visualization. Despite that, we use 8 as the number of topics because it turns out to be the most human interpret-able. If we had more time, we wan to figure out why the number of topics suggested by coherence score is different from the number of topics that results in most understandable topics by ourselves.

References

[1] Shashank Kapadia: Evaluate Topic Models: Latent Dirichlet Allocation (LDA) https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0

[2] Himanshu Sharma: Topic Model Visualization using pyLDAvis https://towardsdatascience.com/topic-model-visualization-using-pyldavis-fecd7c18fbf6

[3] Ruma Sinha: Latent Dirichlet Allocation and Topic Modelling https://medium.com/analytics-vidhya/latent-dirichlet-allocation-and-topic-modelling-eea49dc3eea6