Classifying YouTube videos into Unbiased, Informative and False Rumor Videos

Yusuf Elnady
yelnady@vt.edu
Department of Computer Science, Virginia Tech, USA

Abstract

People in the current world has become dependent more than mainly on search engines, and hardly a day passes without people searching for something that will benefit them or just searching to find out the latest news about the topics they like. The problem with a large number of websites on the Internet is that there is a lot of misleading and wrong information that has been published and believed by a group of people. In this paper, I dedicate five topics in which there is a frequent controversy between those who support the truth and those who encourage the spread of false or misleading information. Then I examine several algorithms that can be used along with explaining the pre-processing and the best solutions that I came up with to detect such misinformative videos. Then I extract different patterns, common words, and ready-to-use models that will help later the stakeholders to extract data and will facilitate in training other models to work on different topics other than the five I chose to do in this project. In this project, I focus specifically on retrieving YouTube videos of these topics, then retrieving the comments of these videos, then trying to determine and predict the stance of the video using the user comments on the video itself. I am also explaining which features were specifically used to increase the prediction accuracy of these models.

CCS Concepts: • **Text Classification**; • **Word Embedding**; • **Machine Learning**; • **Data Mining**; • **Supervised Learning**; • **Latent Semantic Analysis**; • **Data Preprocessing**; • **Natural Language Processing**;

Keywords: datasets, Naïve Bayes, SGD Classifier, decision trees, grid search, stemming, text labeling. YouTube API, nearest neighbors ,logistic regression, misinformation , information retrieval , word2vec ,nltk, model evaluation;

Data Analytics Course, December 2020, Blacksburg, VA, USA

ACM Reference Format:
Yusuf Elnady. 2020. Classifying YouTube videos into Unbiased, Informative and False Rumor Videos. In *Proceedings of Virginia Tech (Data Analytics Course).*
, 7 pages.

1 Introduction

Watching videos has become a very important matter for many things that we need in our lives. Videos made the world connected to each other more easily, and anyone anywhere can watch others move in front of them. Hence, the transmission of information became spread very quickly, without guaranteeing the reliability of the information. Search engines are a major aspect that governs our ability to participate meaningfully in public life [8]. The second-largest search engine on the internet is YouTube and for long years, YouTube has been blamed for surfacing misinformative videos [6]. YouTube currently has more than one billion videos watched per day [4], so it is more important than before to start investigating the stance of the videos on YouTube to help to spread the right information and stop the false rumors and biased content from appearing to the people. That is possible by having models trained for every topic that is considered a hot topic that has a lot much controversy about it. By indicating which of these videos are misinformative, they can be then tagged and less appeared in the search results.

In this paper, I present the idea of classifying videos based on the comments of each video of each topic. Although people think that comments are far from the main topic of the video, as some people comment any irrelevant words that are unrelated to the content, but in reality, after trying many machine learning algorithms and a lot of adjustments, I was successfully able to know the stance of the videos just from analyzing and mining the users' comments. I reached an accuracy of 81.9 percent, which is the highest accuracy I was able to get. All of that guides us to the fact that the relationship between the video and the comments may seem trivial, but it has proven highly efficient in predicting whether this video is supporting a particular point of view or not and most importantly indicating its point of view. My project is a multi-classification, as the content and viewpoint of the video may be neutral or based on scientific reasons, or it may be rumors and misleading information. Also, my results showed that when removing stop words, the classification accuracy and F1-score decreases, which can be explained by

the fact that some of the deleted words contain is critical to the meaning of the comment and for the classification issue, and after removing them the model becomes unable to perform well. In general, comments tend to take the form of revulsion and exclamatory questions and removing these words greatly affected the classification.

2 Data Collection and Description

The initial dataset consists of more than 2900 videos that has the following features: {'Topic_Name', 'Label', 'Description', 'Title', 'URL', 'Stance', 'Duration', 'No_of_Views', 'Popularity', 'No_of_Likes', 'No_of_Dislikes', 'No_of_Comments' }. These videos consist of different topics as shown in the Topic_Name feature. Each video is annotated by its stance as either +1, 0, -1 which are informative, unbiased, or false rumor videos. Then, for each video, I used the YouTube API to obtain the comments. The results were more than 2900 files; a JSON file for each video. The JSON files were too complicated to be used in the learning process, so I processed each of them and extracted the useful attributes for each comment on each video from each file, and saved them as CSV files. Every comments file has the following features: { 'comment', 'No_of_Likes', 'Topic', 'stance', 'ID }.

Having many files is also not helpful for training a machine model, so I ended up combining the comments that belong to the different topic into one excel sheet file while adding another column feature that indicates the topic of this video. In a nutshell, you can refer to Figure 1. to understand the structure of the dataset. The final number of comments I got is 1,851,784 comments that cover the topics. You can access these dataset (More than 330 MB) using the following link: https://drive.google.com/drive/folders/1v0JOVtLR-x2hPfg1tro8-DgT6wZ34CdP?usp=sharing

3 Data Preprocessing

Most machine learning algorithms require data to be formatted in a very specific way, so datasets generally require some amount of preparation before they can yield useful insights [2]. In my project, I heavily needed to pre-process my comments as they come from the users, and for sure it will contain noisy data that affects the accuracy of the models. Also, you need to know that not all comments are written in English, and not all words can be analyzed using the existing corpus for the English language. So, there were a lot of steps to prepare the data such as stemming, removing comments in other languages, converting text into a text-encode format that allows having some characters that don't exist in ASCII code. Another issue is that some comments were NaN which was surprising to me. So, I will touch on these issues one by one.

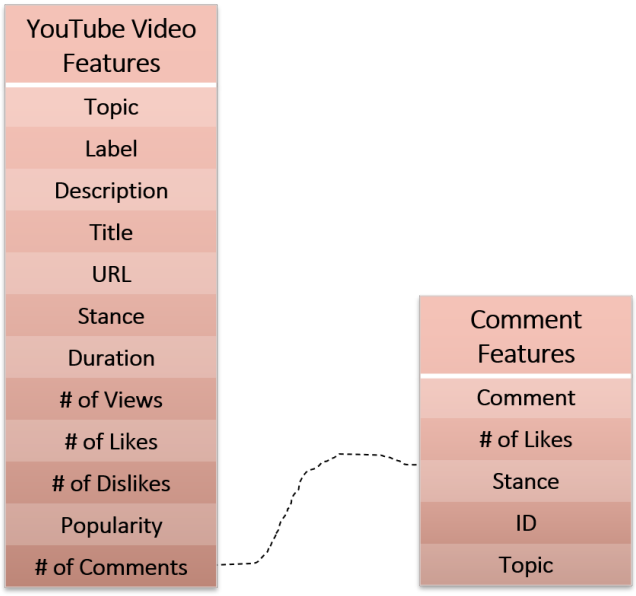

Figure 1. The structure of the dataset

3.1 Dealing with Missing Info

To get the YouTube comments you need to be connected to YouTube API and having a developer key, then you can query the videos with their YouTube URL to get the comments. The problem is when I started processing the data, there was an error that comments of float type cannot be processed, which was later found to be NaN comments. They are comments that are deleted by users so their content doesn't exist. The solution I followed is to completely drop any row that has a NaN comment.

3.2 Removing Unwanted words and characters

By exploring many comments, I found that some comments are very short and they just contain some symbols or special character that does not make any sense to remain in the data. There are other comments that are just having single letters as "w w x w w" without having a complete word that gives the comment a meaning. Other comments are written in languages that are not English which again will not be helpful in analyzing the comments because they are rare comments and considered noisy data. These variations in the comments are due to the variation in the users from all over the world accessing YouTube. So, I removed all of these words, special characters, and digits to make the comments clean.

3.3 Tokenization

Tokenization describes the general process of breaking down a text corpus into individual elements that serve as input for various natural language processing algorithms. A token is an instance of a sequence of characters that are grouped

together as a useful semantic unit for further processing. Figure 2 illustrates the process of tokenization on a comment from the dataset.

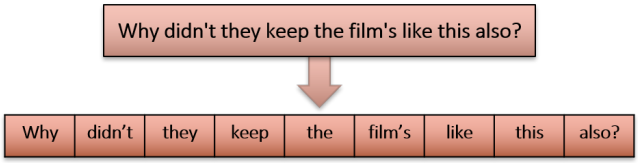

Figure 2. Tokenization

3.4 Stemming

Stemming [5] describes the process of transforming a word into its root form. The original stemming algorithm was developed by Martin F. Porter in 1979 and is hence known as Porter Stemmer. Stemming collapses provisionally linked terms. I stemmed the comments to increase the accuracy of predicting the YouTube position and acts as a word normalization method. Figure 3 shows an example of a comment after it has been stemmed.

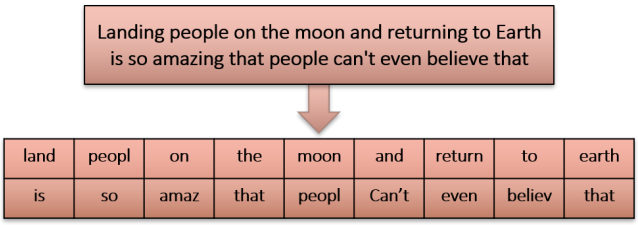

Figure 3. Stemming

3.5 Removing Stop Words

The last step I wanted to do as I find it useful and heavily used in many text classification is to remove the stop words. Stop words are words that are especially frequent in a corpus of text and therefore are considered somewhat uninformative (e.g., words such as so; and; or; and the). However, I ended up not removing stop words from the comments. Later in this paper, I report the metrics I got from running the different preprocessing on many data and showing that I obtained higher accuracy without removing the stop words; otherwise, that will alter the sentiment of the comment.

4 Data Exploration

To explore the dataset, I started by looking at the length of the 1,851,784 comments I have by plotting a histogram. It wasn't clear at the beginning the distribution of the length of them, because there were some outlier comments with a length of 70,000 characters so I removed all of them. The mode of the length of the comment is 28 words which is enough length to provide a piece of information or ask a question. In figure 4, you can find the shape of the distribution which is right-skewed.

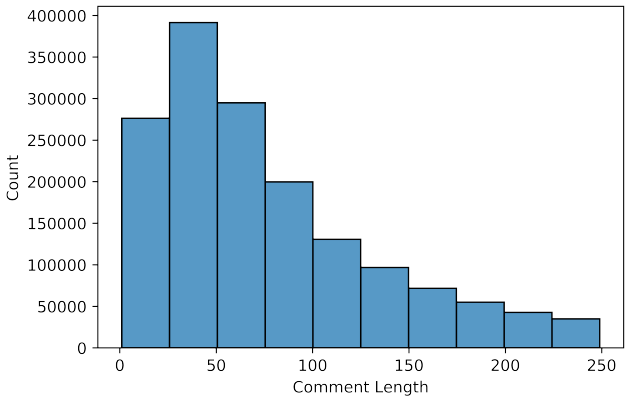

Figure 4. Distribution of Comments Length

To get more insights about the number of comments on each video on each topic, you can refer to Figure 5 to compare them. The most common misinformative content existed in the chem-trails and 9-11 attacks, while we see that the most scientific content is in the controversy between whether the Earth is flat or not. This distribution is showing the data, after the empty comments have been removed, so they are not counted.

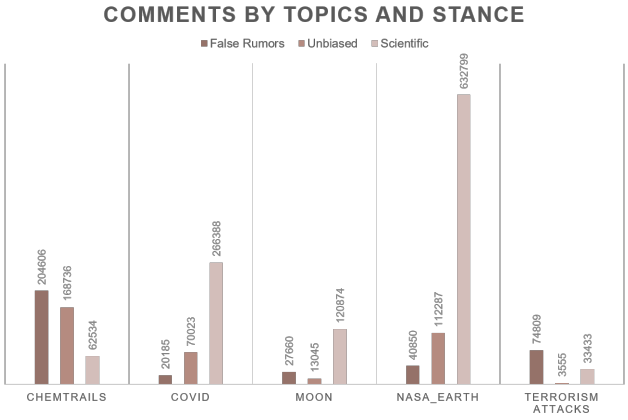

Figure 5. Number of Comments by Topic and by Content Validity

To get more information about the most frequent words in the comments. I state here an example of the words that exist in scientific and credible videos of my list. I used the WordCloud library to produce this beautiful combination of words in Figure 6 that eases the understanding of the data. For accessing the remaining of the word frequencies, I have attached my code to this paper and you can see similar figures for all videos at once, neutral videos, and scientific-based videos.

Figure 6. A collection of the highest frequency words occurring in the scientific comments

I thought also of investigating the like count on each type of these comments, and perhaps the like count can be used as a way to weigh each data-point. But using the trial-and-error, I found that the like or dislike count is not an indicator for the stance of this video or even the comment. The reason behind that is people who tend to watch misinformative videos will get more similar videos that promote the false rumor in their recommended videos by YouTube according to its recommender systems. Therefore, they will tend to believe such information, and they will tend to like the comments that support their stance, even if these comments is not a scientific one. So, I ended up not using the like count at all in my models.

5 Model Building

We can not explicitly apply machine learning algorithms when dealing with high volume text data as it will complicate the model, introduce redundancy, and can lead to low variance and dependent attributes being learned. Thus, one of the most critical steps in any data analytics algorithm is feature selection and reduction. In this project, I test a lot of combinations on feature selection for acquiring the best accuracy on training the models. We test using Tf-IDF vectorizer, by varying the parameter of ngram_range to be unigrams, bigrams, or trigrams. Count vectorizer along with Latent Semantic Analysis has been provided using SVD. After feature selection, the models were trained using Multinomial Naïve Bayes and Stochastic Gradient Descent Classifier.

5.1 Tf-IDF Vectroizer

F-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is a very common algorithm to transform the text into a meaningful representation of numbers which is used to fit the machine learning algorithm for prediction [3]. In this project, we have the column "Comments" that has been preprocessed but it needs to be converted to numbers that can be useful in training. First, I used TF-IDF vectorization, then in other trials, I experimented with CountVectorizer and HashingVectorizer. In TF-IDF vectorizer, Sklearn provides a lot of parameters that must be hyper-tuned in order to let the algorithm works best for your solution. So, again using the grid search that is provided by Sklearn, I grid searched values for specific parameters of the TF-IDF to tune it and get the best numbers to use. These parameters are min_df and max_df which corresponds to the terms with a maximum frequency which therefore will appear in most of the comments and doesn't distinguish them from other comments. For example, max_df = 0.50 means: "ignore terms that appear in more than 50% of the comments". The best values I got are min_df = 0, max_df = 0.1.

Another Parameter that needed to be tuned in Tf-IDF vectorizer is the ngram_range. Unigram model considers each word appearing in the comment corpus as a feature just like a bag of words model [10]. I first produce a corpus out of all the terms that appear in all the comments to construct a feature vector for a comment. A matrix containing the word's frequency is then generated. Finally, with their TF-IDF Term Frequency and Inverse Document Frequency, I multiply these characteristics to give more weight-age to terms that appear less frequently in the reviews and serve as a good discriminator compared to very frequently occurring words that do not help to classify the class. In this case ngram_range = (1,1). The difference from the bag of words approach is when you use Unigram and Bigram, or Unigram, Bigram, and Trigram all at the same time with ngram_range = (1,3). One of the cons of only using unigram as features in NLP is the presence of word relations like "not good" in the reviews which are not captured by the unigram approach as it uses singular words in an isolated fashion. To cover such relations we extend the unigram approach by taking into consideration bigram features as well. The corpus is now appended with both unigram and bigram features with their respective TF-IDF scores matrix for feature selection. After experimenting with many variations for this parameter, I found that the best values are ngram_range = (1,2). The following, Figure 7, is an example of how Unigrams, Bigrams, Trigrams work by combining two/three words next to each other as one feature.

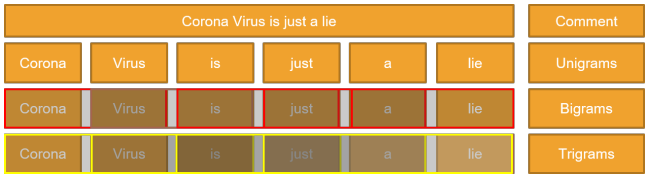

Figure 7. Example of how Unigrams, Bigrams, Trigrams work

5.2 Latent Semantic Analysis

To the sake of completeness, I tried also to do dimensionality reduction with LSA which is described to capture the latent structure and semantics of the comment. One of the major

problems in Natural Language processing is the presence of a large number of features extracted from the text which leads to complex models and high computation requirements, ultimately leading to low accuracy and poor results. [7] LSA match topics instead of exact words i.e. words occurring in a similar context. I made a matrix generated by unigrams and apply Singular Value Decomposition, SVD, to yield three matrices USV out of which V forms our final feature matrix, then you can construct a word-review matrix D, of dimensions of the unigram corpus. Finally, perform the Singular Value Decomposition on D. The problem I faced is that I was using Multinomial Naïve Bayes as my classifier, and the truncated SVD matrix returned some negative or NaN numbers that couldn't let the MultinomialNB work. Also, it doesn't make sense to use MultinomialNB with LSA, because in Naïve Bayes we assume the independence between the features columns. As, in the following subsection 5.4, you will see that I was also using SGD as a classifier, but I also encountered some problems even after removing the NaNs from the matrix. The solution I resorted to is to use Logistic Regression, but it was extensively taking time to grid search its best parameter combination, so the prediction accuracy wasn't so promising using LSA.

5.3 Multinomial Naïve Bayes

The Naïve Bayes Classifier assumes that given a class the conditional probability between any two features is independent of each other. [9] I calculate the posterior probability given a feature using this assumption and build the model. Then, when a new feature is encountered, we compute all the joint probability values of the class for that feature and the highest probability is the output as the final class label for this new feature. In this paper, we use multinomial Naïve Bayes Classifier, which assumes the Probability of some comment I given some video is a multinomial distribution of all comments in the database. Multinomial Naïve Bayes is generally used in document classification as it works well for data that can be converted into frequencies, like the count of the words in the text [9]. Again, I used the grid search algorithm provided by Sklearn for hyper-tuning my pipeline, and I got the best α, the additive smoothing parameter, to be 0.1. In section 6, Model Evaluation, I will compare the different combinations of algorithms I used and feature selection, along with the metrics that were the criteria for evaluation.

5.4 Stochastic Gradient Descent

[1] Stochastic gradient descent is a very popular and common algorithm used in various Machine Learning algorithms, most importantly forms the basis of Neural Networks. There are a few downsides to the gradient descent algorithm. For example, we need to take a closer look at the amount of computation we make for each iteration of the algorithm, or otherwise we will ran out of memory. [1] SGD solved this problem as is also common to sample a small number of data points instead of just one point at each step and that is called "mini-batch" gradient descent. Mini-batch tries to strike a balance between the goodness of gradient descent and the speed of SGD. Using grid search, the best parameters are loss = 'hinge', penalty = 'l2', alpha = 0.0001, max_iter = 1500.

6 Model Evaluation

To get an overall idea about the overall steps my code follows in order to classify the video by comments, you can refer to Figure 8. The pipeline is consisted of 12 steps each has its own impact on the algorithm and its effectiveness. For experiment purposes I split the review data as 80% training and 20% test data. The metrics I report are the accuracy, precision, recall, F-score, but the criteria I used to select the best algorithm and best setting was basically depending on the accuracy. All of the code is written in Python with the help of Scikit Learn, Pandas, nltk, seaborn, genism, and matplotlib Library. All the code is provided in the interactive Jupyter notebooks for result replication. **The highest accuracy I got was 83%**

For evaluating the models, and as you will see in the five trained classifiers I provide in my code, there are many settings that I have tried. Each settings of them results in a different accuracy. The final model I propose is the one with the highest accuracy. First, I started by MultinomialNB and SGD Classifier without removing stop words using unigrams. Then I evaluated the same settings using bigrams and trigrams. The combination of unigrams and bigrams outperformed the others. Then by comparing MultinomialNB and SGD Classifier, in most of the five topics classifiers, MultinomialNB was the best. For example, in Figure 9 and 10, I gave here the numbers for classifying videos that talk about whether the flat is earth or not. As I mentioned in section 5.2, I tried using Latent Semantic Analysis but it wasn't such success due to the lack of computer resources I needed in order to train SVM models, because Naive Bayes and SGD classifiers don't work on matrix that has negative numbers.

So far, I have been using specific trained model for each algorithm. So, I tried to make one model that can captures the stance of any video of any topic just by having the comments as input. After many trials, that are shown in the submitted code, the highest accuracy I managed to get is 0.77%, which is quite low comparing to the results obtained from the separate models for each topic. That's why the final solution has many classifiers not a single one.

I was also intending to try SVM/SVC, but due to its time complexity and the low resources I have, it was taking a long time to just fit some training data. That's why it is not included in my project, but for future work, I recommend to start exploring it on the same dataset.

7 Real-world Insights

In this section, I provide a set a list of actionable insights acquired from the project.

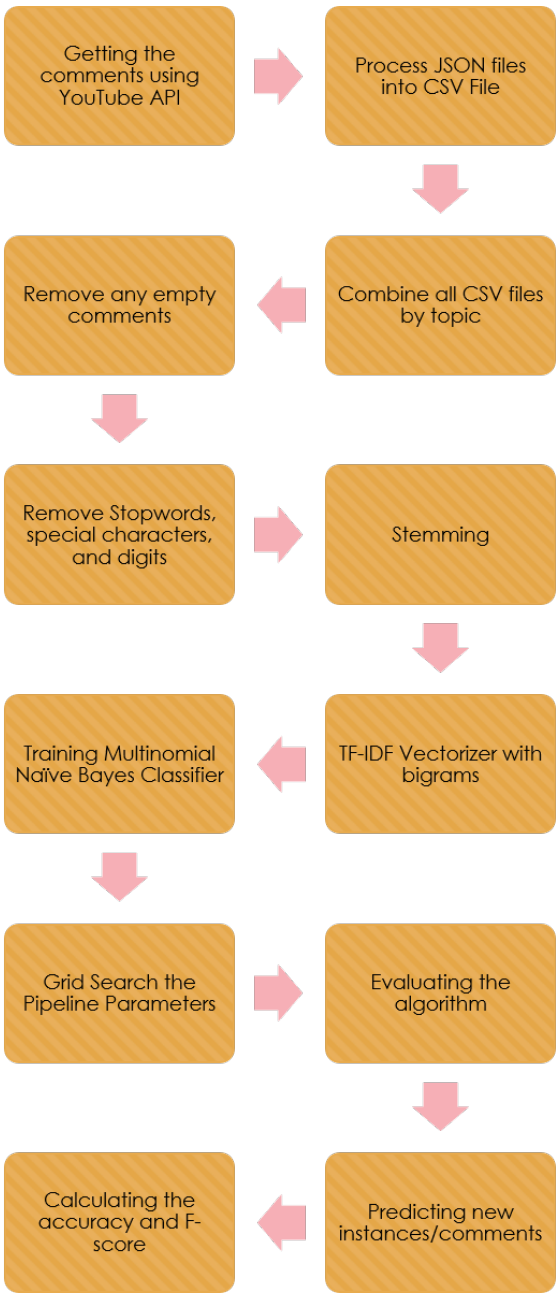

Figure 8. Overall summary of the pipeline model used in training and predicting the stance of YouTube Videos

ID	Settings and Classifier	Accuracy
1	MultinomialNB without removing stop words using unigrams	0.82
2	SGD Classifier without removing stop words using unigrams	0.82
3	MultinomialNB Classifier without removing stop words using Bigrams	0.83
4	SGD Classifier Classifier without removing stop words using Bigrams	0.81
5	MultinomialNB Classifier without removing stop words using Trigrams	0.82
6	SGD Classifier without removing stop words using Trigrams	0.81
7	MultinomialNB with removing stop words using bigrams	**0.83**
8	SGD Classifier with removing stop words using bigrams	0.81
9	MultinomialNB Classifier with removing stop words and using stemming with bigrams	0.82
10	Logistic Regression Classifier using Latent Semantic Indexing	0.81

Figure 9. Overall summary of the pipeline model used in training and predicting the stance of YouTube Videos

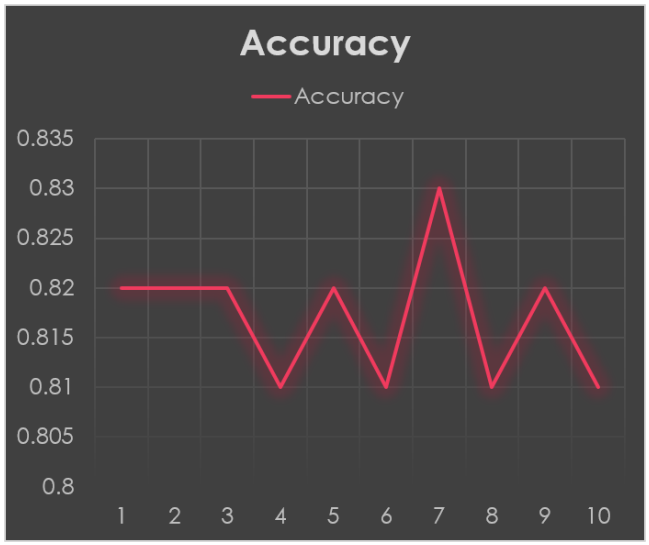

Figure 10. Overall summary of the pipeline model used in training and predicting the stance of YouTube Videos

- Comments are important in helping to understand the data and analyzing it.
- Choosing the correct Feature Selection technique is as important as choosing the correct learning technique.
- Traditional Machine Learning techniques perform well on the text provided that the features fed into them are robust and good.
- Like Count cannot be used as a feature to weigh the samples because it is varying from user to the other, and people who watch misinformative content will have more of these videos and they will intend to like these comments that support their perspective, while other may dislike it because it doesn't support their scientific beliefs.
- Also, my results showed that in some topics when removing stop words, the classification accuracy and F1-score decreases, which can be explained by the fact that some of the deleted words contain is critical to the meaning of the comment and for the classification issue, and after removing them the model becomes unable to perform well. In general, comments tend to take the form of revulsion and exclamatory questions and removing these words greatly affected the classification.
- Having one model for classifying any video into misinformative, unbiased, or scientific-based is not a good idea. Rather it's better to have models trained for specific models that have some controversies around them.

8 Lessons Learned

In this project I dived deeper into the field of NLP, data preprocessing, and text classification. I learned about using TF-IDF Vectorizer, Count Vectorizer, LSA, and I tried to dive into word2vec but I didn't have time to learn the deep learning and LSTM that it is depending on. I have also practiced a lot using machine learning libraries, such as sklearn, pandas, NumPy, matplotlib, nltk, genism, and seaborn. I learned some useful components that sklearn provides such as Pipeline, GridSearch, FeatureSelection, TruncatedSVD, NMF, and many others. I have also learned how to train models using my data and how to clean and preprocess it before feeding it into the classifier. I have tried many settings in order to get the highest accuracy and F1-score I can approach. Also, I learned using the YouTube API and Google Cloud Platform.

If I would have some opportunity again, what would I have done differently? After I almost finished working on my project, I started hearing about other advanced state-of-the-art techniques, such as RNNs, Word2Vec, LSTM and BERT. So, I would have started learning deep learning and neural networks and testing on them. I would also like to do more hyperparameter tuning to boost accuracy. I am also thinking of boosting and stacking. For future work, I am also thinking of using the features that are related to the video itself as auxiliary information to help further in predicting the stance of the video. Also, I would have used a GPU for training, so I don't waste a lot of time waiting for the code to complete and produce the results.

References

[1] 2019. *Stochastic Gradient Descent — Clearly Explained !!* Retrieved 2020 from https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31

[2] 2020. *Data Preparation.* Retrieved Dec 2020 from https://www.datarobot.com/wiki/data-preparation/

[3] 2020. *TF-IDF Vectorizer scikit-learn.* Retrieved Dec 2020 from https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a

[4] 2020. *YouTube for Press.* Retrieved Dec 2020 from https://blog.youtube/press/

[5] Pawan Tamta B. P. Pande and H. S. Dhami. [n.d.]. Ph.D. Dissertation.

[6] Nick Carne. 2019. *Conspiracies' dominate YouTube climate modification videos.* Retrieved Dec 2020 from https://cosmosmagazine.com/social-sciences/conspiracies-dominate-youtube-climate-modification-videos

[7] Prabhakar Raghavan Christos H. Papadimitriou, Hisao Tamaki and Santosh Vempala. 1998. Latent semantic indexing: A probabilistic analysis. *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (1998), 159–168.

[8] Tarleton Gillespie. 2014. *The relevance of algorithms. Media technologies: Essays on communication, materiality, and society 167.* Retrieved 2014 from https://www.microsoft.com/en-us/research/wp-content/uploads/2014/01/Gillespie_2014_TheRelevance-of-Algorithms.pdf

[9] Andrew Y. Ng and Michael I. Jordan. MIT Press, 2002. On discriminative vs. generative classifiers A comparison of logistic regression and naive bayes. *T. G. Dietterich, S. Becker and Z. Ghahramani, editors, Advances in Neural Information Processing Systems* (MIT Press, 2002), 841–848.

[10] Abir Messaoudi Riadh Bouslimi and Jalel Akaichi. [n.d.]. Using a bag of words for automatic medical image annotation with a latent semantic. *CoRR, abs/1306.0178, 2013* ([n. d.]).