Analyzing Conspiracy Content on Youtube

Noah Yann Lee & Will Duke

12/18/2020

Introduction

Youtube automatically recommends videos in large part based on engagement metrics that are agnostic to video quality. This practice has drawn criticism for its potential to expose users to conspiracy theories and other spurious claims.[1] In response, Youtube has employed content moderators and automated moderating techniques to remove the most egregious conspiracy content from their recommendations, some of which had drawn hundreds of thousands of views.[2] A longitudinal study tracking conspiracy videos on Youtube over a 16 month period found that the company's efforts to demote offending channels had been effective, though the prevalence of conspiracy videos on the site remained relatively high.[3]

The authors of the conspiracy study made some of their data publicly available, including the results of manual scoring for hundreds of videos on whether they contained conspiracy content.[4] For this project, we attempted to recreate their logistic regression model which classified videos by whether they contained significant conspiracy content. We applied unsupervised learning methods to identify patterns and characterize the major groupings in the data.

Background

Youtube

YouTube is a streaming site where billions of videos are watched per day.[5] It's popularity has been quite concerning to some, however. Sociologist Dr. Zeynep Tufekci recently wrote that "…YouTube may be one of the most powerful radicalizing instruments of the 21st century". There are many important metrics for YouTube, where content creators and YouTube itself receives money for advertisements through user engagement. The simplest factors for user engagement are video views, likes, comments. Most of this data is freely available online by web scraping or utilizing YouTube's query-limited API.

Preceding Work

Marc Faddoul, Guillaume Chaslot, and Hany Farid released a report titled "A longitudinal analysis of YouTube's promotion of conspiracy videos" which was picked up by the New York Times. Their goals were to record video recommendations provided by YouTube and measure the percentage of conspiratorial videos being promoted. The authors utilized natural language-processing techniques, Google API, and machine learning to apply text classification to video classification. They found discriminating words for conspiratorial videos and charted YouTube recommendations from 2018 to early 2020. This original paper used a manually curated and labeled set of around 200 conspiracy videos from a book about YouTube, about 200 "randomly scraped" videos, and listings from 4chan and reddit, totally around 1,100 videos with an

[1] Ovide, S. (2020, April 20). Take YouTube's Dangers Seriously. Retrieved December 17, 2020, from https://www.nytimes.com/2020/04/20/technology/youtube-conspiracy-theories.html

[2] Roose, K. (2020, October 15). YouTube Cracks Down on QAnon Conspiracy Theory, Citing Offline Violence. Retrieved December 17, 2020, from https://www.nytimes.com/2020/10/15/technology/youtube-bans-qanon-violence.html

[3] Faddoul, M., Chaslot, G., & Farid, H. (2020, March 06). A Longitudinal Analysis of YouTube's Promotion of Conspiracy Videos. Retrieved December 17, 2020, from https://arxiv.org/abs/2003.03318

[4] https://github.com/youtube-dataset/conspiracy

[5] https://www.omnicoreagency.com/youtube-statistics/

even split between conspiratorial videos and non-conspiratorial videos. Their model used "relative weights" for Logistic Regression by using features weight 52% for video comments, 22% for video snippets (title + description), 14% for the video caption (approximated transcripts) and 12% for the perspective score (Google API score of all three previous features). Rather than using just a binary classifier, they generated conspiracy likelihoods which accurately were intended to reflect the actual likelihood of a video being conspiratorial, for example, "70% of videos with a likelihood score of 0.7 will be conspiratorial." Per traditional metrics with a threshold at 0.5, their classifier had a precision of 78% and a recall of 86%. Importantly, the authors found that "words that characterize no conspiratorial content are more random, reflecting the fact that the negative training set is mostly not cohesive." As relevant later, we suspect that this sampling may be impacting the results. A better view of the general YouTube landscape would likely require a more accurate relative distribution of conspiracy videos within the training set, which of course skews heavily non-conspiratorial.

Modeling with PHATE

We have selected PHATE for our dimensionality reduction and visualization pipeline because it is adopts characteristics of diffusion maps and tSNE which we found relevant for the dataset. Notably, we suspected YouTube data to have an underlying manifold because of the complex relationships that often occur in natural language landscapes. Expectations about these nonlinear relationships might lead to using graph representations and walks to recapture distance metrics (geodesic distance), however we also suspect the YouTube data to contain a lot of noise and outliers. For example, videos on the same topic may have a drastically different approach to the video tags or comments section based on the channel manager. Likes and comments can also be disabled on videos, providing dramatic graph frequency signals. Therefore, we discounted using techniques like Isomap because we suspected some videos would lie off the important manifolds and break the shortest distance metrics. Additionally, we did not want to capture only global distances or variance within the videos using a technique like PCA because the topical nature of conspiracy videos has a lot of importance with local neighborhoods. With the smaller sample size (further shrunk by efforts to replicate the original dataset lacking invalid video IDs), there were additional concerns over the seed method used for the original data resulting in some areas to have an especially sparse sample rate. Therefore, we wouldn't want trajectories or groupings to shatter because of t-SNE. Thus, we chose PHATE as our main visualization technique.

Related Work

Conclusions from Faddoul et al.

In their work, "A longitudinal analysis of YouTube's promotion of conspiracy videos", Faddoul et al. concluded that YouTube's policy changes had impacted the percentage of recommendations for conspiracy videos and appeared to improve results on the aggregate. However, they state that this data can still "hide very different realities for individuals" and there is suspicion that personalized recommendations and channel subscriptions can still lead to these "conspiracy trajectories" which further amplify extreme effects of social media. We think our work provides the second of two parts for extending these speculations and providing some verification. A full follow-up would mimic users by processing those personalized recommendations and walking through the data using a YouTube account with recorded history. We have demonstrated a relevant pipeline on a public process which visualizes those suspected conspiracy trajectories and offers a tool for planning further analyses.

The OSoME Project

There are currently other researchers and institutions implementing ML and AI techniques to platforms similar to YouTube. OSoME, which stands for Observatory on Social Media, is a project from Indiana University studying and fighting spread of misinformation and manipulation of social media. Hoaxy is a visualization technique released under this project for twitter searches. Additionally, OSoME is using YouTube itself for other visualizations about trends on social media and graph data.[6]

[6]https://osome.iu.edu

Natural Language Processing

Substantial previous work has delved into various algorithms for processing text so that it can be optimally used by a classification model. Both `nltk` and `sklearn` provide powerful libraries for this task. Standard text normalization procedures often removes text features that are unlikely to be predictive such as URLs, punctuation, and numbers. To further reduce the feature space and improve model performance, text processing pipelines also often include lemmatization or stemming procedures, where words a truncated to either their most basic form or more bluntly to remove suffixes, respectively. This process collapses features that would occupy multiple columns in a traditional bag of words vectorization procedure (such as 'answered' and 'answers') into a single feature ('answer') that is likely to be more informative. Truncating the size of the feature space by taking the first `n` components from a singular value decomposition (SVD) can further decrease the feature space.[7]

In order to analyze text data, we had to choose a schema for embedding text into a vectorized form. We chose two approaches for this: the first, Doc2vec, provides an efficient method for learning vectors that represent documents. Doc2vec represents each document (in our case a video description or set of comments) as a dense vector from the latent space of a model that is trained to predict the words within the document.[8] This model has the advantage of preserving contextual information unlike a bag of words of approach as well as providing embeddings of equal size for documents of variable length. The second method, TF-IDF vectorization, normalizes word frequency counts by their inverse document frequency, thereby increasing the importance of words that occur frequently in only a subset of the documents. These are used in the somewhat simpler context of a bag-of-words approach, which does not preserve the order of words in the data. Though simpler, this method has the advantage of requiring no tuning, and works well with methods such as support vector machines for text classification.[9]

Methods & Theory

Data Collection

The publicly available data from Faddoul et al.'s work identifying conspiracy videos contained video IDs, titles, descriptions, comment counts, and conspiracy likelihood scores. A separate set of data contained a smaller number of videos with manually scored labels indicating whether the video contained conspiracy content.

We initially sought to supplement this data with a distance matrix developed by scoring each video in the original dataset on whether and where they occured in the recommended videos of each of the other videos in our data. In this way, we could develop a distance matrix by computing `1 / log(rank + 1)` for each video pair, where `rank` refers to the position of the recommendation in the list on each video page. We would than symmetrize the matrix such that it would satisfy the requirements of a distance metric. First, we attempted to download the related videos for each video in our data using Google's Youtube v3 API, which exposes a convenient method for just this kind of search. Unfortunately, Google recently reduced the number of queries allowed of this kind such that we would have only been able to request information on less than 100 videos each day.[10]

With the API no longer an option, we developed a method using `selenium` to automatically visit the page of each Youtube video, parse the page's HTML and CSS to identify links to related videos, and save their video IDs. Though much slower (the relevant code takes more than 24 hours to collect all of the data and will

[7]Bengfort, B. (2018). Applied Text Analysis with Python: Enabling Language Aware Data Products with Machine Learning. O'Reilly Media, Incorporated.

[8]Le, Q., & Mikolov, T. (2014, May 22). Distributed Representations of Sentences and Documents. Retrieved December 18, 2020, from https://arxiv.org/abs/1405.4053

[9]Robertson, S. (2004, October 01). Understanding inverse document frequency: On theoretical arguments for IDF. Retrieved December 18, 2020, from https://www.emerald.com/insight/content/doi/10.1108/00220410410560582/full/html

[10]The `yt_api_query/relatedvidgraph.py` file provides documented methods for calling the Youtube API and parsing the response. At first glance, the query limit for free Google projects of 10,000 queries per day seems more than sufficient to gather information on 4200 videos. Unfortunately, calling the `list.search` method counts as 100 queries, which equates to only 100 actual requests. To make matters worse, each initial request can return multiple page, each of which incurs a 100 query penalty to view.

repeatedly open tabs in a browser) and likely not as complete as the API method, we were able to record between 50 and 120 related video IDs for most of the videos in our dataset. From these, we constructed a distance matrix as described above. However, this matrix was so sparse – almost half of the nodes were isolated – that we were unable to draw interesting conclusions from (or in some cases, even run) our analyses on these data.

Instead, we returned to the Youtube API, and extracted the top 100 comments for all of the videos in both sets of data from the original paper that were still available. We then combined these data with our existing video data for our ensuring analysis.

Data Preprocessing

To create a feature matrix from the available data, we constructed a data processing pipeline that processes and combines the text from the titles, descriptions, comments, and tags for each video. As a first pass, we developed a custom text normalization scheme that uses word lemmatization and tokenization methods provided in `nltk` as well as some additional regular expressions and simple screens to remove punctuation, URLs, and other text anomalies that are unlikely to be predictive. In this pipeline, we employed separate techniques for the comments and the other features. Reasoning that keywords would be better captured with TF-IDF vectorization, wherein word frequencies are computed and normalized by the frequency of those words throughout the document corpus, we applied this technique to vectorize the titles, descriptions, and tags for each video.

For the comments, we trained a document embedding model known as Doc2Vec provided by the `gensim` library to create vector representations of the comments section of each video. The TF-IDF vectors were concatenated with the Doc2Vec vectors within an `sklearn` pipeline to create our final feature space. We also added an option to reduce the dimensionality of the feature space by applying a truncated singular value decomposition procedure to the data.

To speed up model evaluation, the pipeline was fed pre-cleaned and tokenized training data which were then vectorized for each fold of the cross validation procedure. This prevents data leakage between the training and test sets, thereby ensuring that the testing error is not biased downward because the model had partial access to the test set during training.

PHATE in Pseudocode

For a broad outline, a summary of our process for creating the PHATE embeddings is as follows:

```
Load_data() # Fetch video ids, related video data
YouTube_data <- load_data()
Filtered_data <- call_YouTubeAPI(YouTube_data) # supplement data from API
PHATE_coords <- PHATE(processed_data) # create PHATE embedding

Graph <- make_Plotly_Graph(PHATE_coords, color=processed_data['conspiracy_score'])

Dash.start_server()

Dash_display_graph(graph)
```

We built a service that allows a user to interrogate the video data within the PHATE plot interactively. Methods for loading this can be found inside the Github repository for this project.[11]

Please see the file `conspiracy_phate_demo.gif` for a demo of the interactive map of Youtube videos.

[11] https://github.com/WillDuke/ConspiracyGraph

Regression Methods

In an attempt to replicate and improve on the classification models used in the original paper, we used a series of classification methods including logistic regression, feed forward net classification, linear support vector machines, and a stochastic gradient descent classifier to compare how they perform on predicting the manual labels from the training data based on the feature set described above. We were unable to improve on the paper's scores, in part – perhaps – because we had a smaller training set to work with since about 20% of the videos that were in the original data were no longer available for our analysis. In each case, we used the `sklearn` implementation of the relevant model and performed 5-fold cross-validation to find average precision, recall, accuracy, and f1 scores for each model. Each model was run on two versions of the data set: with truncated SVD (for latent semantic analysis) or without.

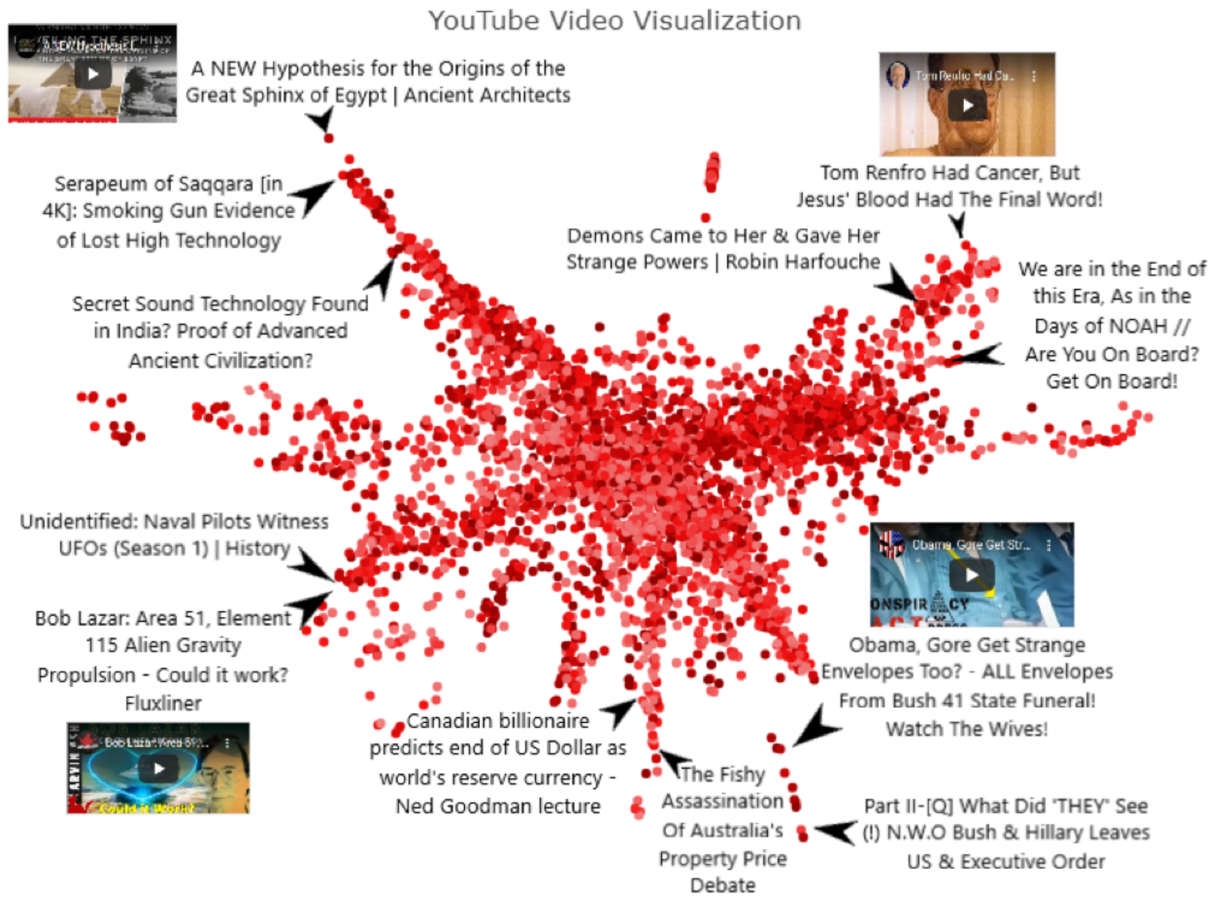
Empirical Results

Classification Modeling

X1	model	precision	recall	accuracy	f1
0	MLPClassifier()	0.7025678	0.6926567	0.6926567	0.6926567
1	SGDClassifier()	0.6525940	0.6392675	0.6392675	0.6397498
2	LogisticRegression()	0.6412991	0.6324271	0.6324271	0.6291293
3	MLPClassifier() with TrSVD	0.6746269	0.6424519	0.6424519	0.6259107
4	LinearSVC() with TrSVD	0.6201028	0.5902421	0.5902421	0.5905884
5	SGDClassifier() with TrSVD	0.6270218	0.5923712	0.5923712	0.5877664
6	LogisticRegression() with TrSVD	0.6208558	0.5846989	0.5846989	0.5800810
7	LinearSVC()	0.5527522	0.5311173	0.5311173	0.5247779

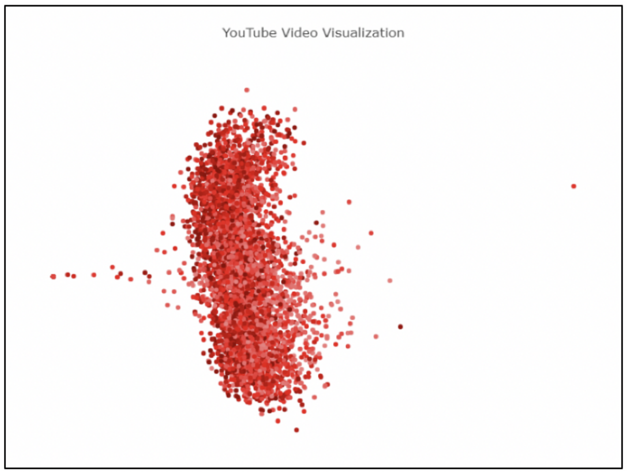
We compared the performance of a variety of models on the training data provided in the original paper. Here, we can see that of the four models tested on both versions of the feature space (created as described in the Methods), we achieved the best performance with `sklearn`'s `MLPClassifier` method applied to the original features without dimensionality reduction. We tried a series of iterations on both the data and the model hyperparameters, including introducing a bigram model into the TF-IDF vectorizer and learning embeddings for the descriptions as well as the comments. However, we found that these additions to the feature space actually decreased the accuracy of the models. It was interesting to note that the simpler seemed more powerful at times. This is a testament to the difficulty of finely tuning more complex embedding schemes, particularly when the sample size is small. In this case, the sparsity of our feature space increased the cost on performance of over-parametrizing the models. In future work, we might consider adding to our training set via a more thorough pull from the Youtube API for comment data, as well as performing data augmentation by adding confidently predicted videos to our data set. It is worth noting that the data used in this study did not represent a random sampling of the videos available on Youtube, so we we cannot use our model's performance on this set to estimate its performance on new data. Our next step with this data would be to establish methods for gathering and labeling videos so that they are more representative of the true data distribution.

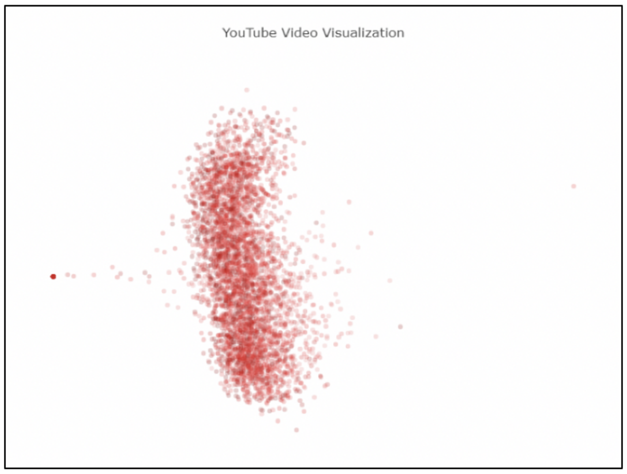
PHATE Observations



We found the PHATE visualization paired with an interactive plot allowed us to manually assess the quality of the dimensionality reduction quite well. As visible in the figure and the interactive plot demo provided in the GitHub, the topics along branches are cohesive and form a narrative which moves from recommended video to recommended video. Notably, capturing only recommended videos based on the YouTube API query did not yield as tight results, mostly due to the sparsity of being able to find a set video in the node set amongst all videos recommended by YouTube. Important to the original paper was the question of whether this analysis captures conspiracy scores. Using KNN on this new space might provide a way to quantitatively describe the effort, however the abundance of videos in the middle with a low score could be more evidence to a point made in the background: more random topics are usually not conspiracy videos. On the flip side, this also suggests that most videos are not conspiracy videos, which further complicates the generalizability of these analyses for YouTube as a whole.

Comparing to PCA





To validate our choices for the method, we intended to use PCA as a null comparison with no expectations about the figure other than an inferiority to PHATE. However, we were surprised to find a small correlation with the conspiracy scores captured by PCA as well as an interesting cluster of nodes which captured nodes missing data from comments. While the PCA provides distances between topics very poorly, it was interesting to see a somewhat cleaner representation of the scores than PHATE. According to YouTube's Chief Product Officer Neal Mohan, "it is not the case that extreme content drives a higher version of engagement" [Roose K. Youtube's Product Chief on Online Radicalization and Algorithmic Rabbit Holes. The New York Times 2019]. We suspect there is some type of differential engagement going on in these videos which are conspiracy videos, and we have captured that without the use of the Google API Perception analysis using PCA. These results would be worthwhile for future follow-up.

Conclusions

The original study on YouTube conspiracy videos emphasized longitudinal trends and aggregate results. Their classifier had an uphill task given the complex and subjective context, but it appeared to work well for following and characterizing trends over time. Our approach to visualization and user access to that visualization provides a key component to extending this work and refining the classification process. Our processing of YouTube video data and meta-data has successfully recaptured topical trends in the style of YouTube recommendations. Given more data, this would also provide a useful tool for characterizing subsets of conspiracy videos which is an important approach given the heterogeneous nature of these topics. While the general nature of conspiracy videos follow certain trends, the oftentimes isolated communities engaged with these topics may provide difficulties with assessing conspiracy videos in the "YouTube wild", whereby the original paper's estimates they represent around 2% of the top recommended videos on a monthly basis. Our approach to assessing trajectories of recommendations through topics on YouTube would likely provide focus and utility to projects trying to further characterize that diverse landscape.