YouTube Video Category Classification

Terahn Harrison
Department of Computer Science
University of Victoria
Victoria, BC
V00842812

Joshua McIntosh
Faculty of Engineering
University of Victoria
Victoria, BC
V00875715

Jeremy Megyesi
Faculty of Engineering
University of Victoria
Victoria, BC
V00870363

Amanda Munro
Department of Computer Science
University of Victoria
Victoria, BC
V00802056

1 Introduction

YouTube has grown into a monolith in the world of video sharing and the internet as a whole. It is currently the most visited website in the world by an overwhelming margin, with over two billion unique monthly users [1]. It has become synonymous with video sharing and is currently one of the biggest sources of gatherable public data online.

Given the vast quantity of videos on Youtube, it becomes necessary to organize videos to allow ease of viewing. Videos are sorted on Youtube by assigning labels that indicate the associated characteristics of the video. Such labels include attributes such as video categories, age ratings, monetization status, etc. Video categories are particularly important as they allow viewers to easily view popular videos that are trending in their preferred categories. The assignment of these labels using machine learning algorithms has been a hot topic in the past, with many Youtubers complaining about the blackbox that is the "Youtube Algorithm". The machine learning algorithms used to sort and label videos can be largely responsible for how many views a given video will obtain as well as how much money the content creator is able to generate from the video. Given that increasingly more individuals are trying to use Youtube as a source of primary income, the accuracy of Youtube's video classification is a high priority.

2 Problem

In this paper, three unique classification models were investigated that can be used to help automatically classify the category of YouTube videos using the video data made available by the YouTube API. These three different classification models each use a separate element of the video data as input. These three elements are the video thumbnail, the video transcript, and the video statistics. There are 32 unique categories that YouTube uses to classify videos. For simplicity, this paper will focus on two of these 32 categories: Gaming and News & Politics. These specific categories were chosen as they are believed to contain noticeable differences in the three feature sets used in the classification models to achieve good results.

Every YouTube video requires a thumbnail image. The thumbnail can either be a separate image uploaded by the uploader, or can be a still frame chosen from the video. Either way, the thumbnail is meant to provide a pictorial representation or summary of the video. The features were extracted from the thumbnail in order to build a feature set that can be used to train a machine learning model to classify the category of a video. It is believed that there are distinguishable patterns in the thumbnails that a model could be trained to recognize.

The English transcripts were particularly useful for determining category classifications, as different video categories will tend to have their own vocabularies. The transcripts of the videos often provide the greatest insight into the content of the video, especially for videos such as VLogs (video logs) which rely heavily on dialogue. Videos can auto-generate transcripts through voice recognition (yet another Natural Language Processing machine learning algorithm), although this is not always completely accurate. It can sometimes be more reliable to look at more popular videos where subtitles/closed captions have been written by viewers. Viewer contributions are usually reliable (more so than auto-generated) as they require approval before being published. Viewers with a smaller audience are less likely to have decent captions (if any at all) given the number of people watching/contributing. Therefore, in some instances, it may be necessary to rely upon the auto-generated captions. It is worth noting that not all videos will have auto-generated captions unless enabled by the video creator.

Each YouTube video has an associated count of likes, dislikes, views, and favorites. These video statistics were used to try and determine the category of the video. Since all categories of videos have varying like/dislike ratios and view counts, this model was not expected to yield perfect results. However, some insight into if certain categories of videos are more or less recommended may be gained while analyzing viewer counts. As well, there may be insights into whether certain categories of videos get higher or lower like/dislike ratios and overall favorites.

3 Related Work

There has been some work done in the field of video classification in the past. One of the first papers to investigate video classification using YouTube videos specifically was from within Google itself in the 2010 paper that attempted to categorize YouTube videos by using a taxonomic classification system [2]. This paper is now 10 years old so it is unclear how YouTube currently categorizes its videos.

4 Dataset

Problems were encountered when finding an initial dataset for this project due to inconsistencies in YouTube data. Firstly, given that the initial dataset contained videos from 2017, many of the videos had since been removed. This made obtaining thumbnails and transcripts for these videos no longer possible. Finding suitable YouTube videos was also bottlenecked specifically by the transcript model. Not all videos have transcripts, so many of the videos found in existing datasets were unusable which significantly reduced the size of the dataset. Furthermore, only English videos were to be used, which further reduced the number of usable videos in the pre existing datasets. In order to use the same dataset on all three models, all models could not use any of these videos that did not meet the transcript requirements.

In light of these problems, a decision was made to build a custom dataset. This way it could be ensured that all of the videos in the dataset were usable by all three models. This was done by querying the YouTube API for the most popular videos in each category in each month from 2010-2019. Another additional query for each video would then have to be sent to request the video's transcript. Only once the video's transcript was returned and it was detected to be in the English language would the video be added to the dataset. This process resulted in a dataset containing 5454 unique videos with 2727 in each category.

5 Image Analysis

5.1 Feature Extraction

As thumbnails are an important factor in enticing people to a video, thumbnail analysis is an interesting topic by identifying differences in thumbnail creation amongst different genres. For this project, six features were extracted from each thumbnail: the mean colour value for each RGB channel, the mean gradient value, whether the image contained text, and whether the image contained a face. Each feature would hopefully aid in identifying the differences in and thus classifying News & Politics thumbnails verses Gaming thumbnails.

Each feature type provided unique challenges in order to extract them. Calculating the mean values of the colour channels and gradient was fairly straightforward. The numpy library has built-in functions for extracting these values and could be done in a single line without much effort [3]. Creating the gradient image was done manually through the convolution of the Sobel operator.

The algorithm for determining whether text was present in the thumbnail relied heavily on Zhou et al.'s EAST deep-learning detection algorithm and an OpenCV's implementation of it [4], [5]. Initially, an algorithm incorporating Google's Tesseract optical character recognition (OCR) engine was used, which would also allow the text to be "read" and returned, but the accuracy was extremely poor due to large variations in the fonts and font sizes of thumbnails [6], [7]. Tesseract was designed to read clean and uniform text, such as text from a book, and thus was deemed unsuitable for this application. The EAST algorithm provided excellent performance in detection of text and was suitable as only the presence of text was needed, not the text itself.

Face recognition was achieved using OpenCV's Cascade classifier, which uses Haar feature-based cascade classifiers to identify faces [8], [9]. Due to factors such as variations in image quality, face orientation, and obscurity, this algorithm achieved 66% accuracy in small-batch testing. Although not favourable results, this feature was left in as the quality of the face and whether it was detected could be a factor in assisting classification.

5.2 Results

Five different models were trained using the features extracted in the previous section. Each model's hyperparameters were tuned using k-fold cross validation. The results are shown in Table 1.

Table 1: The test accuracy of different machine learning models trained on the thumbnail features.

Model	Accuracy
Decision Tree Classifier	58.5%
Random Forest Classifier	66.2%
Neural Network	63.6%
Logistic Regression	63.6%
Linear SVM	52.7%

As shown in the table, the Random Forest classifier performed the best on the dataset with an average test accuracy of 66.2%. Overall, however, none of the models do a particularly good job of correctly classifying the dataset (a completely naive model can obtain 50% accuracy on a binary classification task). This is most likely due to the lack of quantity and quality of relevant image features to train with. The importance of each feature for a trained model can be viewed to obtain a better understanding of the quality of each feature. In Figure 1, the feature importance of each feature for the trained Random Forest classifier are shown.

As shown in Figure 1, the 3 colour means and the gradient means have rather equal importance in the trained model. The face and text detection features, however, do not hold much importance in the trained model. This could be because the actual detection is producing inaccurate results, or it could just be because these features genuinely are not important when determining the category of a video.

6 Transcript Analysis

Originally anticipated as a particularly fruitful medium to use with classification of Youtube videos, transcripts were selected as one of the three modules of the project. As previously discussed, Youtube videos sometimes provide the option for auto-generated captions while many (often an option for videos with more views) allow for viewer-submitted captions. Having both of these options available indicated that there would be a sufficient amount of transcripts available for analysis.

6.1 Feature Extraction

The first task in this area of the project was to find a feasible way of collecting the transcripts of thousands of videos from the given dataset of videos (transcripts were not provided in the initial

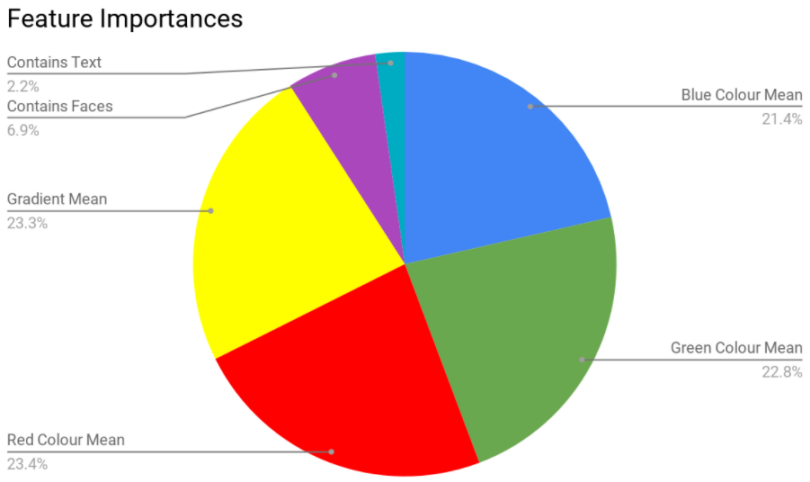

Figure 1: Feature importances for the Random Forest Classifier.

dataset itself). Luckily, a 3rd party (and open source) Python API was available to obtain video transcripts given their unique video ids [10]. The transcripts were converted into a numpy file and were then ready to be used for classification. It was much more efficient to process the transcripts this way since the largest bottleneck was in obtaining the transcripts from the API. The process of obtaining all 5454 transcripts took around 1 hr 20 mins.

As stated in the Dataset section, not all features had available transcripts and it was necessary to modify the dataset used for final classification. This was likely because some videos did not contain dialogue or auto-generated captions were not enabled.

6.2 Model Selection & Implementation

This component required some additional research on the topic of Natural Language Processing (NLP). It was well known that machine learning techniques had been used with text and language processing. This is evident in features such as predictive text features present in smartphones and has even gone as far as to determine whether a piece of writing was written by Shakespeare (there numerous articles on this subject, but here is one such example [11]). The biggest question was "how can such a thing actually be implemented to classify transcripts?" At first, this seemed not quite as clear-cut as analysis of numerical data. A more specific question that arose was "which model should be used to classify transcripts (documents)?" After researching these questions, it was determined that the two most effective models to use would be a Naive Bayes Model as well as a Linear Support Vector Machine (SVM). The Naive Bayes Model was created using the NLTK (Natural Language Toolkit) library while the SVM model was created using SKLearn's `SGDClassifier` (stochastic gradient descent) [12][13].

In order to classify documents, it was first determined that any given machine learning model would need to have some way of quantizing the data. For the Naive Bayes model, this meant constructing what is known as a TF-IDF (Term Frequency - Inverse Document Frequency). This was first implemented without the use of any libraries as an exercise to better understand exactly how words were being vectorized, although support for such a task was found in both libraries. NLTK had a function called FreqDist() (Frequency Distance) and SKLearn had a `TfidfVectorizer` object [14]. These were later used to prepare the data for the classifiers.

The term frequency essentially calculates how many documents each word is present in for each group. Of course, the higher number of documents that a word appears in, the more important the word is to that group. By using the libraries previously mentioned, filler words (such as "the", "and", "it", etc.) are accounted for, and words that appear much more in one class than the other are considered more important. The NLTK library has a useful function for displaying these "most informative features" as can be seen in Figure 2.

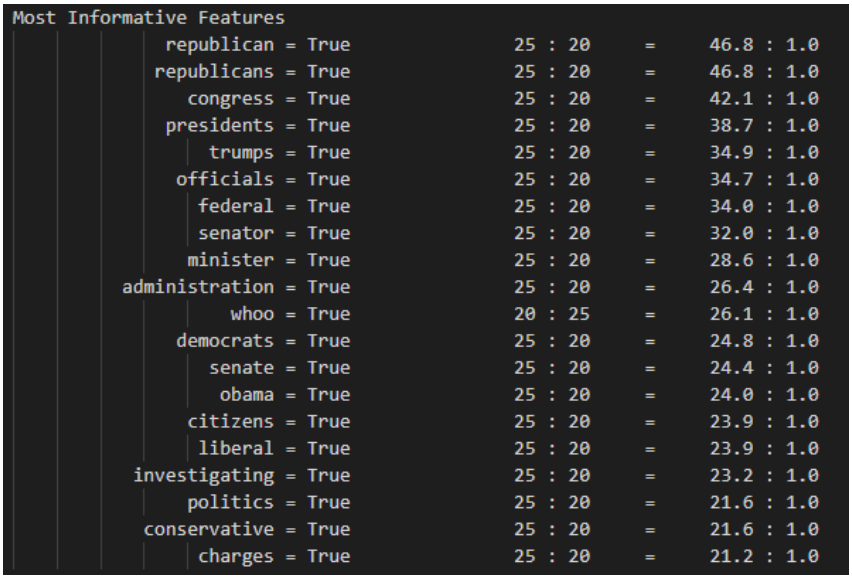

Figure 2: The most informative features for the trained model.

As can be seen, the majority of most informative features come from the News & Politics category. The top most informative feature in this case was the word "republican", which had 46.8 times the occurrences (note that this is using LaPlace smoothing). It was anticipated that each category would have their own distinct vocabulary that would be useful for classification, but based on results, it would seem that this is much more so the case for the News & Politics videos rather than for the Gaming videos.

While no parameter tuning was required for the Naive Bayes model, there were many parameters that could be adjusted for the SVM. However, it was found that many of the default values ended up being the most effective. For example, the alpha value (used to multiply l2 regularization term) default is 1e-4, but it was found that just about any alpha value smaller than 1e-3 yielded around the same accuracy.

6.3 Results

The accuracy was tested and averaged across 10 different sample runs. The data was shuffled before each sample. Each model was fitted using 60% of the dataset, leaving 40% for the test set. The accuracy on either model only changed at most around 2%. The averaged results can be seen in Table 2.

Table 2: The test accuracy of different machine learning models trained on the transcript data.

Model	Accuracy
Naive Bayes	90.7%
Linear SVM	75.8%

The SVM model proved to be more accurate and required less time to train and run; the SVM model completed the classification in around 2 seconds while the Naive Bayes model took closer to 30 seconds. According to an article from The Medium, Naive Bayes is often the best choice on smaller sized datasets, while SVMs can be much more effective on larger datasets [15]. It would appear that the size of the dataset was large enough for the SVM to be the more effective choice, even though both models performed very well.

7 Statistics Analysis

Due to the generic nature of likes, dislikes, comments, and views, the video statistics model was originally not thought to be a promising classifier of YouTube video category IDs. However, the initial results of classification with video statistics produced promising results. A table of initial results with no hyperparameter tuning can be seen in Table 3. It was theorized that this high accuracy could be due to either Gaming or News & Politics videos having a discrepancy in either likes, dislikes, comments or views.

Table 3: The training accuracy of different machine learning models trained on the statistics data.

Model	Accuracy
Decision Tree Classifier	79.7%
Random Forest Classifier	85.0%
Neural Network	84.1%
Logistic Regression	82.5%
Support Vector Machine	80.6%

After an analysis of video statistics was done, it was found that News & Politics videos had a much higher ratio of dislikes to likes than Gaming videos (see Figure 3). This is likely due to the controversial nature of some news stories in tandem with the divisive political climate seen within the last five years. This was determined to be the most important feature when classifying YouTube Category IDs.

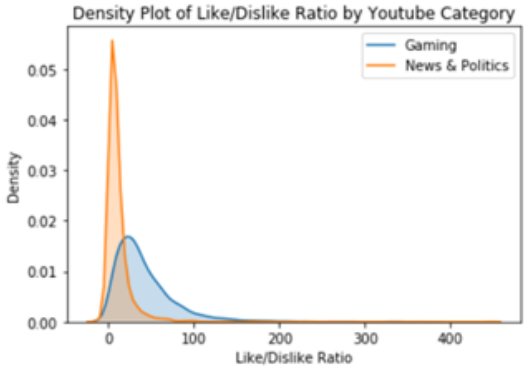

Figure 3: Density plot of like/dislike ratio by YouTube category.

The random forest classifier was determined to be the best model for video statistics classification and hyperparameter tuning was performed. The parameters which were varied were the complexity parameter for pruning, the split criterion, and the number of forests in each tree. A table of varying parameters and their associated mean scores after k-fold cross validation can be seen in Table 4. With the optimal parameters, the random forest classifier had a final test score of 85.2%.

8 Conclusion

The three models (video image, transcript, and statistic) were originally going to be joined into one distinct classifier; however, initial project timelines were shifted due to the loss of a team member, the complications surrounding the dataset, and the University of Victoria's transition away from face-to-face learning. These extenuating circumstances created a change in the problem goal from a combination to a comparison of the three models.

Each model was initially expected to achieve an accuracy of higher than 50% due to it being a binary classification problem. However, the image analysis model performed lower than expected (see Table 5). This is most likely due to an underdeveloped feature set, as both Gaming and News & Politics thumbnails are very easy to discern with the human eye. It can be seen from the table below that the transcript analysis performed the best with a final test accuracy of 95.7%.

Table 4: Test results for different Random Forest Classifier configurations.

Parameters				
Alpha	Criterion	N_Estimators	Mean Test Accuracy	Rank
0.01	entropy	50	83.9%	1
0.01	entropy	100	83.8%	2
0.01	entropy	10	83.7%	3
0.015	entropy	100	83.6%	4
0.015	entropy	50	83.5%	5
0.01	gini	10	83.4%	6
0	gini	50	83.3%	7
0.025	entropy	10	83.2%	8
0.01	gini	100	83.1%	9
0	entropy	100	83.1%	10
0	gini	100	83.1%	11
0.02	entropy	10	83.0%	12
0.01	gini	50	83.0%	13
0	entropy	50	83.0%	14
0.02	entropy	50	83.0%	15

Table 5: The best test accuracy achieved for each model.

Model	Accuracy
Image	66.2%
Transcript	95.7%
Statistic	85.2%

References

[1] K. Gordon, *Youtube*. [Online]. Available: https://www.statista.com/topics/2019/youtube/#dossierSummary__chapter2.

[2] Y. Song, M. Zhao, J. Yagnik, and X. Wu, "Taxonomic classification for web-based videos," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2010, pp. 871–878.

[3] *Numpy.mean¶*. [Online]. Available: https://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html.

[4] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, *East: An efficient and accurate scene text detector*, 2017. arXiv: 1704.03155 [cs.CV].

[5] A. Rosebrock, *Opencv text detection (east text detector)*, Aug. 2018. [Online]. Available: https://www.pyimagesearch.com/2018/08/20/opencv-text-detection-east-text-detector/.

[6] Tesseract-Ocr, *Tesseract-ocr/tesseract*, Mar. 2020. [Online]. Available: https://github.com/tesseract-ocr/tesseract.

[7] *Pytesseract*. [Online]. Available: https://pypi.org/project/pytesseract/.

[8] *Cascade classifier*. [Online]. Available: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

[9] S. Tiwari, *Face recognition with python, in under 25 lines of code*, Mar. 2019. [Online]. Available: https://realpython.com/face-recognition-with-python/.

[10] *Youtube-transcript-api*. [Online]. Available: https://pypi.org/project/youtube-transcript-api/.

[11] K. Kibbe, *An algorithm can tell us how much shakespeare was actually written by shakespeare*, Nov. 2019. [Online]. Available: https://www.insidehook.com/daily_brief/books/an-algorithm-can-tell-us-how-much-shakespeare-was-actually-written-by-shakespeare.

[12] *Source code for nltk.classify.naivebayes*. [Online]. Available: https://www.nltk.org/_modules/nltk/classify/naivebayes.html.

[13] *Sklearn.linear_model.sgdclassifier¶*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html.

[14] *Sklearn.feature_extraction.text.tfidfvectorizer¶*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

[15] A. Rai, *Text classification in nlp - naive bayes*, Feb. 2017. [Online]. Available: https://medium.com/@theflyingmantis/text-classification-in-nlp-naive-bayes-a606bf419f8c.