YouTube Video View Prediction

Brett Bass[1] and Evan Ezell[1]

***Abstract*— YouTube has increasingly become a prime space for content creators to distribute their content. Work has been done in predicting views using meta-data associated with the video such as category, title, etc. We assess the use of meta-data for predicting Youtube views using a random forest model on a subset of videos from three categories of the Sports 1M Dataset [1]. We also provide the framework for using transfer learning with convolutional neural networks (CNN) to analyze the images in a Youtube video. We conclude that random forests provide a useful model for predicting Youtube video views.**

I. INTRODUCTION

Since it was founded in 2005, YouTube has increased in popularity and the quantity of views a creator obtain is becoming more and more lucrative. The exact value of a view is unknown due to the complex cost structure that YouTube utilizes to pay content creators (ad views, skipped ads, video topic, etc).

There are many factors that can be used to predict the success of a video, which we define as the number of views. Among these factors are the channel popularity, title, date published, category, etc. While these are important factors, they do not capture the actual content of the video.

We chose to perform our analysis on three categories of sports videos. These categories were bowling, table tennis, and American football. These videos were obtained from a subset of the Stanford and Google Sports 1M data set [1]. We chose these videos as they were all similar type and similar length.

We used a random forest model to predict the views of a video given its associated meta-data. We chose this model due to its robustness, ease of tuning hyper-parameters, generalization ability, and interpretability. The random forest model's default hyper-parameters often yield good results, and typically do not require a large amount of tuning to get good results. The random forest model provides variable importance based on the number of splits of each variable in the model making it easy to interpret which variables most influenced the predictions.

When just using meta-data to predict views, it is not clear whether the number of views is driven by the actual content of the video. Thus, in addition to using a random forest model, we provide the framework for using CNNs to capture features of the videos.

CNNs in recent years have taken the lead in image and video processing because of their ability to capture features in an image. CNNs are used in such cases as facial recognition, video analysis and natural language processing.

University of Tennessee, Knoxville

[1]Bass and Ezell are graduate research assistants in the Bredesen Center, University of Tennessee, Knoxville, TN 37902 USA

II. DATA

Since the video content of YouTube varies so vastly, we are restricting this study to specific subsets of YouTube sports videos as mentioned above. These videos vary vastly in channel popularity and type of sports video within their category.

All YouTube videos have associated meta-data that will be considered for the analysis. This data includes author, category, description, likes, dislikes, length, keywords, title, and view count. The data will be scraped from YouTube using the python package 'pafy' as well as the YouTube API. !! One issue that was encountered was the reliability of pafy. Pafy would often provide a response but was not consistent and failed to pull views correctly. Because of this, we turned to the YouTube API for this data.

III. DATA DECISION

We had to decide what types of videos we wanted to pull in. We could have chosen particular channels or particular categories using the tags on Youtube videos. This might have taken us longer to decide which particular videos we wanted to pull in and given the limited time of this project we decided to use a curated list of Youtube videos that were already out there. We researched what available datasets were out there and what the community doing video research was using to decide on potential Youtube videos. We ended up choosing the Sports 1M Dataset which was used in a paper titled Large-scale Video Classification with Convolutional Neural Networks by Karpathy et al [1].

This dataset consists of 1,133,158 video URLs each of which have been tagged with 487 different classes. Given the size of this many videos we had to narrow down the amount of videos we could actually use for our project. We decided to include a subset of videos from 3 different classes: bowling, table tennis, and american football. The decision to use these classes was completely arbitrary. We decided to sample 1000 videos from each category.

IV. DATA COLLECTION

A. Getting the video links

The video links were provided in two text files from the curators of the dataset. The dataset was partitioned into training and testing sets. We aggregated both of the files into a single file containing all of the individual links with their accompanying tags. We split the 1000 URLs from each category into 3 separate text files corresponding to their category.

B. Pafy

We researched various ways to pull in the data for each video and came across the Python package titled Pafy. This Python package is built to scrape Youtube content and metadata. It allows you to download the Youtube video as an mp4 file, which was necessary for using CNNs. Pafy allowed us to pull in the same metadata that is provided from the Youtube API, but in a much more intuitive interface. This worked quite well except for a couple of pitfalls.

We had trouble getting the Pafy package to properly retrieve the views for each video. It proved to be unreliable, working sometimes for a URL and not working others. Also, the Pafy package did not provide the ability to get additional metadata such as the number of subscribers of the videos channel. Pafy only allowed us to retrieve data specific to the URL provided. Thus, we had to find some way to fill in the missing holes of our data, specifically for Youtube views since this was our intended response variable. We turned to the Youtube API to solve these issues. Had we known the problems we were going to run into ahead of time, we would have probably opted to use only the Youtube API since it provided all of the necessary metadata and much more reliably than Pafy.

C. Youtube API

The Youtube API is slightly more confusing than using Pafy and requires navigating through the documentation to figure out how the resources works. The Youtube API has a lot of different resources so finding which one you need can be quite challenging. Nonetheless, the documentation is well written and the correct information can be retrieved with a little bit of effort.

The number of views video specific metadata which allows retrieval via the Youtube video id. We gathered the views using the correct API and added the returned information to our dataframe. The reason Pafy does not gather the subscriber count for an individual video is because it is not video specific data, rather it is channel specific data which Pafy does not interface with.

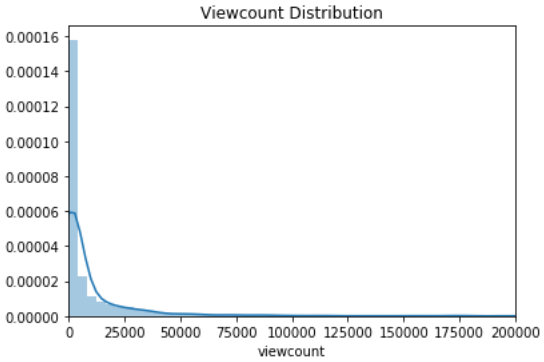
In order to scrape the subscriber data, we had to use the channel resource which only takes Youtube channel ids in order to retrieve the data. Thus, we first scraped the channel id from the Youtube video. We were then able to successfully grab the data associated with the Youtube channel. We added subscriber count to the data frame to use in our modeling process.

We were not able to collect all 1000 videos for each category because some of the videos have since been taken down from the creation of the dataset. Of the 3,000 videos we tried to collect, we were able to successfully retrieve the video and metadata for 2,730 videos.

V. Exploratory Data Analysis

The scraped data was congregated into a pandas dataframe. Some exploratory data analysis was done to determine how best to proceed with the analysis. The original features that were scraped were title, description, length, likes, dislikes, rating, date published, subscriber count, and view count.

Fig. 1. Viewcount Histogram



There were some missing values in the data that had to be either imputed or removed. There were 34 missing values in the likes attribute and 34 in the dislikes attribute (from same videos). We determined that the best way to impute these values was with the ratio of views to likes and dislikes in the rest of the data.

As there were not too many features, we were able to look at the distributions each variable as well as the target variable (view count). Several of the distributions such as view count, likes, and dislikes had very high skew as you would expect with most videos having few views and some popular videos having many. The view count distribution can be seen by the histogram in Figure 1.

For this analysis, we believed that view count outliers may disparage the results so this led us to remove outliers for views that were outside of three standard deviations from the mean. This filter removed 28 videos. These videos were clearly outliers with the average number of views for the videos removed being above 550,000 with a maximum above 3.2 million.

Video length also had a very highly skewed distribution with some videos lasting more than seven hours. Outliers were also removed according to this feature that were outside of the 95th percentile. This value was chosen based on the videos that were at this border (and with future work in developing CNNs in mind). This filter ultimately removed 135 observations and changed the shape of the distribution as shown in Figure 2.

An additional column was added based on the date published which was time since published. The idea of this variable was to guide the model in the fact the the longer a video is out, the longer is has to gain views.

Title and description could not be used as strings and one-hot encoding would be cumbersome and ineffective for this case. For this reason, we used sentiment analysis on both the video title and description. We used a pre-trained sentiment analysis model called VADER (Valence Aware Dictionary and sEntiment Reasoner) [2].

This model was trained on various social media sources and is great for our use case can interpret strings that we

Fig. 2. Length Before/After Outlier Removal

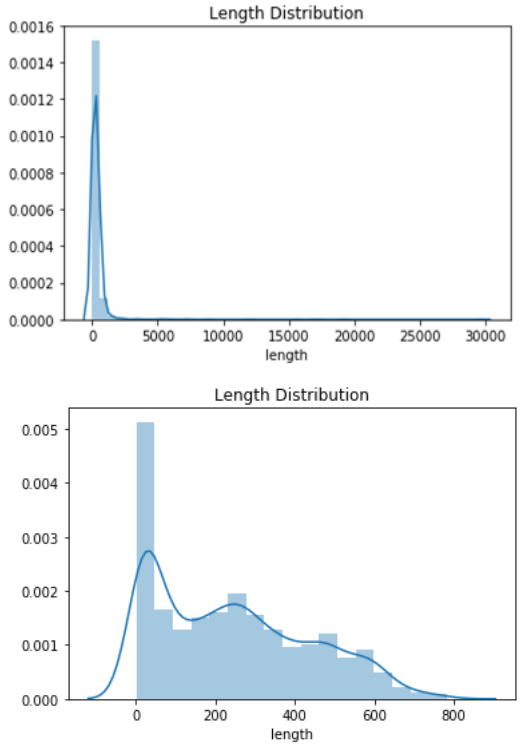
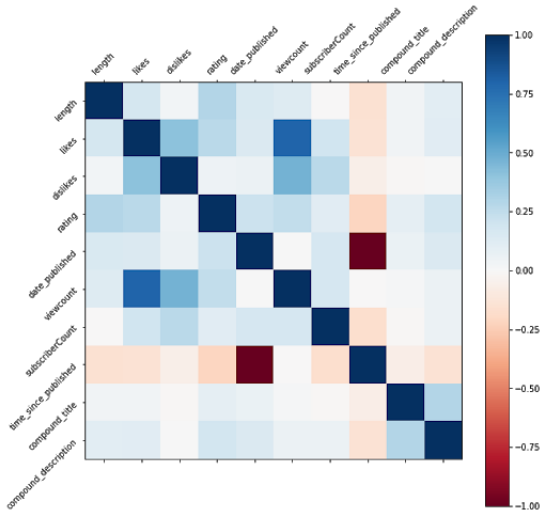


Fig. 3. Correlation Heat Map



may run into such as OMG or LOL as well as :). We ran this model on all of our titles and descriptions and obtained a polarity score between -1 and 1 for each item. -1 represented a fully negative title while 1 represents a fully positive title (0 being neutral). One issue we ran into was that some of the titles and descriptions were not in English. We attempted to solve this issue by using Googles Googletrans API but unfortunately they are experiencing a widespread problem with their service so we could not efficiently translate the languages to English. This resulted in a few of the sentiment values for these observations being equal to 0.

We were also able to look at correlations in our data to see if we had any features that were correlated to each other or to the target variable. We created a heat map visualization from the correlations which can be seen in Figure 3.

Looking at the correlations, most of the strong correlations make sense. There is a strong correlation between likes and views as you need views to have likes. As we said before, there is also a very strong correlation between date published and time since published as it is just a linear transformation.

Even with outliers removed, there was still skew in much of the data. For this reason, we elected to log transform our data before it was input into the model.

VI. Modeling

A. Random Forest

We chose to use a random forest to model the number of views with our features. A random forest works by building an ensemble of decision trees. Random forests use only a random subset of features to choose the best split at each node compared to decision trees where each split is tested to find the best one. Random forests can be used for either classification or regression but regression was used in this case.

Random Forests have some main advantages that led us to choose this algorithm. They are considered very easy to use as often the default hyper-parameters can produce good results as well as that there are relatively few hyper-parameters. As we had less than a semester to complete this project in its entirety, this benefit was clear. Another advantage of random forests is they are less prone to overfitting as the number of trees increases, so does the number of individual randomized models in the ensemble, which will never increase the generalization error. A final quality of random forests that is very useful is that relative feature importance is measured during training. The importance is found by averaging the drop in error for a variable at each split point. This is interesting to see which features have the most impact on the prediction.

B. Training and Tuning

To train and tune the model, we split the data into training and testing with 75% of the data being in the training set and 25% in the test set. We would ultimately use k-fold cross validation so the validation set was indexed into the 75% of the training set.

There are several hyper-parameters that need to be tuned for a random forest:

- N_estimators - Number of estimators (Trees)
- Max Depth - Max depth of trees
- Min_samples_split - Minimum samples required to split a node
- Min_samples_leaf - Minimum samples required to be at a leaf node
- Max_features - Maximum features to consider when looking for the best split

TABLE I

OPTIMAL HYPER-PARAMETERS

Hyper-parameter	Value
N_estimators	600
Max Depth	25
Min_samples_split	2
Min_samples_leaf	2
Max_features	4
Bootstrap	True

TABLE II

REGRESSION METRICS

R^2	0.8
MAE (Full-Dataset)	3795.33
MAE (0-1000 Views)	735.77
MAE (0-5000 Views)	313.45

- Bootstrap - Whether bootstrap samples are used when building trees

Two hyper-parameter tuning searches were done, both using 5-fold cross validation. The first was a random search over the hyper-parameter space to get a general idea of the optimal hyper-parameter values. The next was a grid search using a hyper-parameter grid based on the values obtained from the random search. The optimal values for the random forest are in Table I.

VII. RESULTS

Once the model was trained, we used the test set which was held out through the training process to assess model quality. The mean absolute error metric was scaled into smaller bins to better assess the performance for all values of the target variable.

The metrics for the model were satisfactory for the amount of time we had. There is plenty of room for improvement but we captured the general trend of the data with a good R-square value and the it can be seen in the predicted versus observed plot in Figure 4.

Fig. 4. Predicted Values vs. Actual Values

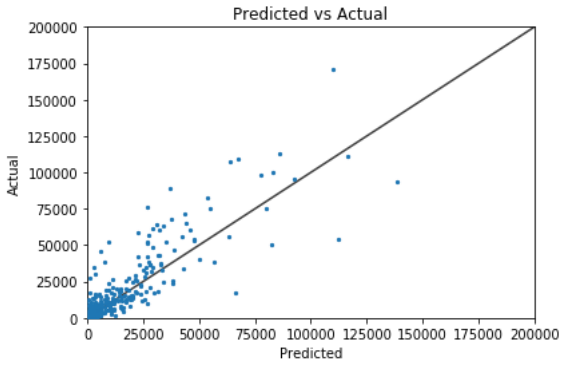


Fig. 5. Variable Importance

```
Variable: likes                Importance: 0.47
Variable: rating               Importance: 0.18
Variable: dislikes             Importance: 0.09
Variable: subscriberCount      Importance: 0.09
Variable: length               Importance: 0.05
Variable: date_published       Importance: 0.05
Variable: time_since_published Importance: 0.05
Variable: compound_description Importance: 0.01
Variable: compound_title       Importance: 0.0
```

TABLE III

ITERATIVE RESULTS

Change	**MAE**
Initial	9827.55
Removed outliers and added subscriber count	5203.4
Log transform (Final)	3795.33

The variable importance was obtained from the model based on the number of splits of each variable. The output of the variable importance can be shown in Figure 5.

The most important variable likes is what we expected as we saw a high correlation to views in the correlation plot. I find it interesting that rating has such a high importance while a variable like subscriber count is somewhat lower. I would expect the number of views to be highly correlated with the number of subscribers. I believe one of the reasons that subscriber count may not have as much of an influence on this model is that many of the videos had somewhat small number of views, meaning that they likely had few subscribers as well. I believe that if we used videos with views of 1 million or more, the results would be quite different. Unfortunately, the sentiment analysis had little impact. This may be because of the translation issue or that many titles of videos that dont have many views (non-viral) have neutral titles that are less of an attention grab.

We did not immediately arrive at our final results. This was an iterative process and our results changed as we made changes to the inputs. To illustrate the improved MAE, we include our most important changes along with their updated error in Table III.

There were obviously other changes made along the way but these were some changes that were impactful. Removing the outliers may have had the largest impact as some of the features that described these videos had different qualities compared to the more normal videos.

Our final model's error measures can be seen in Table II. We binned the results since there was such a large variance in video views.

VIII. LIMITATIONS

A. *Time*

Because this project was started from the ground up, we did not have as much time for the modeling phase of the project. We spent much of our time bringing in our actual dataset and preparing it for use. Also, time did not allow

us to run a CNN on our videos to see if they would help improve our random forest model.

B. Google Translate and VADER Sentiment Analysis

As mentioned above, the VADER package only allows sentiment analysis to be done in the English language. This should not have been an issue given that we could use a translation for the non-English observations of the text attributes in our sample. We attempted to use the Googletrans API to translate the title and description. However, Google is currently having a problem with its API so we were unable to successfully translate these variables.

C. Available Storage for Videos

CNNs require lots of training samples to produce successful working models. Thus, many videos are needed and storage for the videos. Had we proceeded with the CNN modeling, it is likely we would have to use transfer learning because our sample was so small.

It also may have been possible to use online learning techniques for CNNs, discarding each training sample after it was used to limit the amount of space required for the videos. However, this could limit the reproducibility of such a model by discarding the videos.

IX. Future Work and Recommendations

There are several future recommendations that would improve this project going forward. The biggest improvement that would add more depth and variance to the project would be to add more videos of all types. This would include videos of varying view counts, varying categories, channel popularity, etc. This would allow us to determine more broadly what influences the number of views a video will get. The main reason we limited our video selection for this project leads into the next recommendation.

The next recommendation was one of the original goals of the project which was to break down the actual content of the videos to determine if it had any effect on the number of views it would get over time or if this was based more on the meta-data pertaining to the video. The infrastructure for this analysis is set up with python code written to utilize PAFY to download the video, use OpenCV to break the video down into individual frames, then save these frames in labeled folders that were labeled with the binned number of views. The next step would be to use transfer learning with an established model to train the model to classify images into view count bins. We had planned to simply use this as an input to our original model to determine its variable importance and see if it had any predictive power. This was controlled by the time limitation but should be explored in future work.

Another future recommendation is to pull more attributes of the video from the YouTube API. There are several more factors that may have influence on the popularity of a video such as the comments or other data related to the content creator. These may or not have a positive impact on model performance.

One of the limitations was the Google Translate API bug which caused an issue for us if the title or description was in another language. A future recommendation would be to find a workaround for this issue whether that is looking for bug fixes online or finding another translate option.

An interesting aspect of view count prediction to explore when more videos of all types are used is to explore how the number of views grows over time. It would be intriguing to see what factors are at play in a video that goes viral whether this is determined by actual content, attention grabber titles, or by the person who posted the video.

A final recommendation is to attempt to improve model performance by researching and employing different models. Random forests were used as they are generally friendly models with good base performance but there are many other types of models that could be used. It would be very interesting to explore neural network architectures that incorporate the content of the video in addition to the meta-data.

References

[1] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[2] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *ICWSM*, 2014.