Multi-Input Modeling to Forecast YouTube Video Trend Time

Evan Phillips, Jaypal Bhatia, Noor Gill

Saturday April 9, 2022

1 Abstract

As YouTube becomes a heavily monetized service, it gains traction among internet influencers and visibility becomes increasingly important. By developing an end-to-end question answering system, we enable C-level content creators to estimate the amount of time it would take for their video to reach the trending page or even strategize video titles and publish timing for the future. With data for over 57,000 YouTube videos across the U.S., Great Britain, and Canada, we pass input variables like the title, publish date, number of likes and dislikes, comment count, view count, quarter, and whether or not comments and ratings were enabled through a multi-input CNN model with MLP and CNN sub-components. We generate an estimate of the amount of the remaining time it would take a user's video to trend. To measure success in our project we defined three indicators: (1) Our model would be abstracted into a function that prompts the user to input information about a video, (2) Our mixed model would minimize MAPE, (3) The predictions we obtained from our model seemed reasonable. To reflect, our model was abstracted well and guides the user through all steps to receive predictions, we did not minimize the MAPE because it was consistently above 65 for all attempted CNNs. Although the magnitude of predictions did not differ substantially for test videos intentionally created to trend slow and fast, our predictions generally were accurate relative to one another.
Keywords: Youtube, convolutional neural net, multilayer perceptron, social media optimization

2 Introduction

Since its creation in 2005, YouTube has cemented itself as a staple in digital media content. YouTube is a platform providing users with the symbiotic opportunity to generate revenue for Youtube and themselves. In 2021, Content creators continue to grow on YouTube each day, with 500 hours of content uploaded each minute on the platform. Many of YouTube's creators upload videos with one goal in mind: reach trending status.

The goal of our project is to help YouTube users, particularly "C-level" content creators, identify when their video will trend. In this context we are defining C-level content creators as video creators who often post videos with high metadata metrics (view count, like count, etc.) but don't always trend. We differentiate this Youtube demographic from a top tier creator who almost always posts a trending video due to sheer brand power. Our algorithm is tailored towards account managers of C-level content creators or the creators themselves. Inputting metadata of a video's like/dislike count, comment count, publish date, season, category, and comments/ratings enablement along with an analysis of the title using NLP, we are able to predict how long it will be until a video trends (in hours and days). If account managers or C-level content creators, teetering between trending and non-trending videos, have the ability to recognize how soon their account's most recent videos are likely to trend, they can recommend constructive tactics to effectively leverage the trending market. For instance, a video posted that is likely to trend in an amount of time a manager deems to be too high will cause the manager to recommend that the C-level content creator makes more videos to avoid losing the spotlight. Conversely, a video predicted to trend quickly might mean the content creator can relax for a while, earning the Youtuber's equivalent of PTO. Because a C-level content creator is going to be competing to enter the sphere of the most notorious creators, we are only training our model on Youtube videos that have trended.

The motivation to achieve this goal stems from a lack of predictive analysis currently available to content creators and their managers. YouTube provides its uploaders with a robust analytics dashboard, with access to nuanced information after a video has been uploaded. However, noticeably missing from this arsenal of analytics is any predictive applications. This in turn leads many YouTube creators and their management to rely on a "wait and see" approach when it comes to making major decisions such as uploading new content, generating advertising revenue and more. For example, whether a video will trend will have a profound impact on when a content creator will produce and upload more content. Therefore, rather than taking a wait and see approach, our model aims to provide more insights to content creators by helping them predict when a video will trend, and if it is realistic for a video to even do so. It is worth noting that there are also third-party YouTube analytics extensions such as TubeBuddy, which is used by companies such as Pepsi, CBS Sports, and more. Even these third-party extensions lack a predictive feature for when a video will trend. Our project aims to operate like an extension, where users can input details and features of their video to discover how long it would take for their video to trend.

3 Background

As YouTube has grown, so has the research and information behind what causes a video to trend. In the past, many studies have aimed to apply machine learning framework to YouTube trending videos but focused extensively on predicting how long a video would remain popular. For example, one study aimed to use machine learning to predict how long a video would remain trending once it had reached trending status (Rangaswamy et al.). Such a project would benefit those who have already reached trending status with their videos. However, this differs from our project conceptually because we are estimating how long it will take a video to reach trending status. As a result, our project targets very different audiences than previous works in this domain. C-level content creators would stand to benefit from a project more like ours, which does not assume the video has already reached trending status.

Additionally, there has been one other work that also tries to predict how long it would take a YouTube video to reach trending (Niture). However, this project is vastly different in the methodology it employs to reach this goal. For example, the exploratory study uses Random Forests and Naïve Bayes to reach its goal to predict how long it may take a video to reach trending. Meanwhile, our model architecture includes a Keras Functional API. Our model's two branches, the Multi-layer Perceptron (MLP) and the Convolutional Neural Network (CNN), provide a vastly different technical approach to a similar problem. This difference is explored more in depth in the methodologies section of the paper.

Lastly, a vast majority of studies aim to extract metadata from videos and conduct analysis on this data. For example, one study that visualizes this trend sought to mine data extracted from videos that trended (Choe et al.). The study undoubtedly displays useful information for a C-level content producer, but provides generalized outcomes and takeaways based on the data. Our project differs because it aims to provide a predictive analysis for how long it will take a video to reach trending.

4 Methodology

We use a data set that consists only of YouTube Trending Videos with each row corresponding to a single upload. This data, retrieved from Kaggle, is constantly refreshed on the site. We extract the information corresponding to USA, Great Britain, and Canada (57,000+ entries total) since these countries are predominantly English speaking and removing non-English words would not decrease the size or quality of the data set drastically. The given metadata consists of video title, channel title, publish time, trending time, tags, view count, like count, and category Id.

Regarding the model architecture, we take advantage of Keras' flexible deep learning framework to implement a multi-input model for handling mixed data to predict the expected time (in both hours and days) for a video to appear on the YouTube trending page. This model architecture with the Keras Functional API allows us to utilize numeric/continuous values (like count, dislike count, view count, comment count, and publish date) as well as categorical values (title, category, and quarter) as inputs. The model consists of 2 branches, MLP and CNN, that are concatenated to form the multi-input CNN model.

4.1 EDA and Data Pre-processing

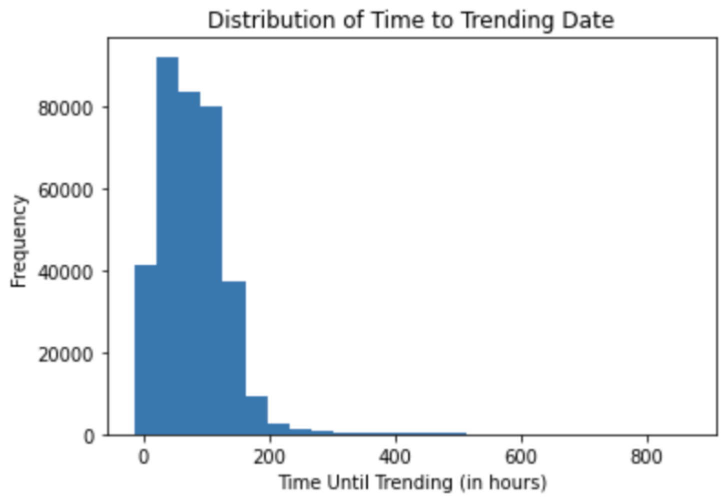

Figure 1: Distribution of Time to Trend

After initially performing some exploratory data analysis on some of the notable features that would be available during prediction time, we determined that most videos on the trending page appeared there within 8 days of publishing (Figure 1) and that like counts are more closely linked to view count and comment count compared to dislike counts, which makes sense since an individual is more likely to watch a video repeatedly or comment if they like it. In terms of data cleaning and pre-processing, we change the publish date to a continuous variable by calculating the number of seconds since the Unix Timestamp, remove non-English words from titles, create a category column that maps the number in the category Id column to the actual category, and validate all other attributes in addition to performing an 80-20 train-test split. Min-max scaling is applied to the continuous features via scikit-learn's MinMaxScaler and one-hot encoding is applied to categorical features via scikit-learn's LabelBinarizer.

4.2 Modeling, Hyper-tuning Parameters, and Validation

First, since we are analyzing titles as a feature, we employ a CNN model which takes in an optimizer and uses a Sequential Model with an embedding layer, Convolutional 1D layer, Global Max Pooling, and 2 dense layers. Second, with the Keras Sequential API, we build an MLP model with a fully connected (dense) input layer with ReLU activation and fully-connected hidden layer with ReLU activation, which allows us to handle mixed categorical and continuous data. After concatenating the MLP output and CNN output, we compile our model using the Adam optimizer. The batch size is set to 8 and there are 20 total epochs. We adapted our model from Adrian Rosenbrock (PhD) who initially used it to analyze images and metadata from real estate listings to predict prices (Figure 2). However, instead of using CNNs with image data, we created an embedding layer of our video title text data. Fitting the model requires the weights being tuned through back-propagation. In terms of hyper-tuning, we test various optimizer choices, as well as batch sizes and epoch numbers to derive optimally performing values for these parameters.

For model evaluation, we use the Mean Absolute Percentage Error (MAPE) as our loss function because we seek to minimize the mean absolute percentage difference between our predicted trend times and the actual trend times. Moreover, we use the root mean squared error as an accuracy metrics, which represent the standard deviation of the residuals. MAPE is used as the objective because we are predicting a continuous output while others doing similar work had categorical or binary outputs.

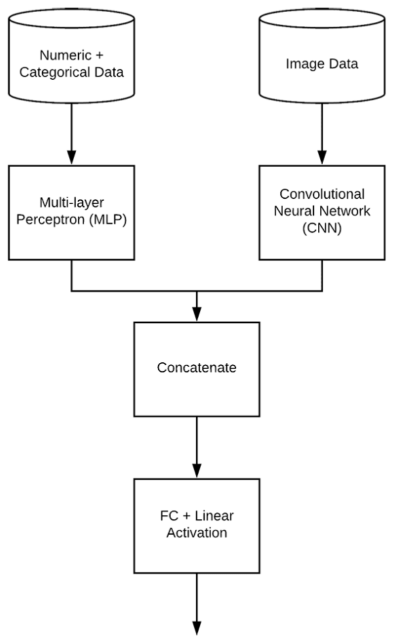
Numeric + Categorical Data
Image Data
Multi-layer Perceptron (MLP)
Convolutional Neural Network (CNN)
Concatenate
FC + Linear Activation

Figure 2: Model architecture overview (Rosebrock 2019)

5 Results

The loss patterns we saw were definitely not what we anticipated. For our concatenated CNN, we observed the test loss decreasing at a constant rate as expected, with the minimum percent error being in the late 40's range, but the validation loss was oscillating from high to low values in the range of 68-77 percent error instead of its expected quadratic shape (Figure 3). In general, the loss of our model is fairly high, hence the accuracy is not as high as we would like. We assume that the chosen input variables are a direct factor in whether a video trends and how fast it trends. Although some videos will trend or fail to trend no matter what combination or values they get for the chosen input features, our approach circumnavigates this assumption.

Our baseline model consists of using the same attributes and outcome variable to test various regressors and classifiers (Gaussian Naive Bayes, Multinomial Naive Bayes, and Decision Tree Classifier). Unfortunately, this yields an accuracy of about 7.6 percent at most with a Decision Tree Classifier. We decide to shift to a neural-network based model in order to deal with potential nonlinear dependencies between the variables and build on it to increase accuracy to at least 30 percent.

While constructing the CNN, we observed that modifying parameters such as learning rate, decay, and momentum in our initial SGD optimizer did not have a noticeable effect on the results which led us to use the Adam optimizer. Moreover, decreasing the batch size had an inverse relationship with the loss such that it increased the MAPE, but we were able to optimize the value at about 8. The number of epochs was dependent on the MAPE and MAE values based on each iteration, represented by the plot of training versus validation MAPE (Figure 3). We did not utilize PCA prior to constructing the model since the neural network model automatically performs dimensionality reduction with word2vec, which applies a shallow neural network to reduce inputs to a dimensional vector similar to the matrix factorization process.

In sharpening our understanding of which features have the most predictive power in estimating the time to trend, we noted that the title alone does better. Adding continuous and categorical variables via the MLP model increases the MAPE, which isn't surprising since a random combination of those variables is unlikely to tell the full story, leaving room for omitted variable bias. Using a CNN when the data set is not high-dimensional is prone to cause over-fitting, which may have occurred in this case. Another interesting observation is that the model performs very similarly with or without the view count feature, even though one would expect a strong positive association between the view count and the time it takes for a video to appear on the trending page.

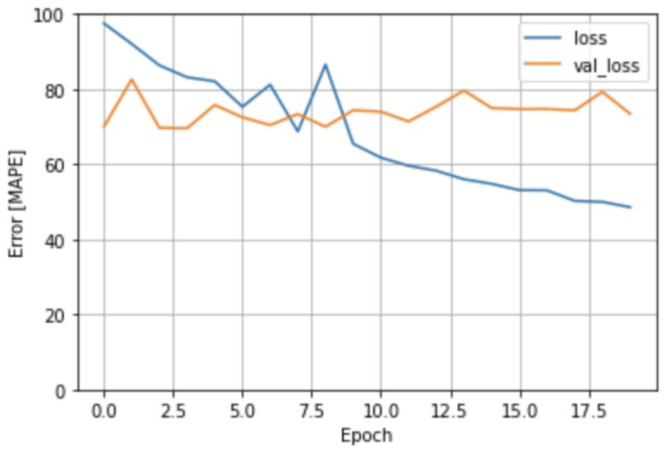

Figure 3: Plot of training versus validation MAPE

6 Conclusion

The mixed model technique we are applying to the trending video problem is unorthodox in nature. The majority of algorithms we've witnessed involved are SVM, Logistic Regression, Naive Bayes, and Random Forests. While maybe not the most common implementation, the mixed model approach was definitely the most intriguing. Concatenating a CNN utilizing an embedding layer of all of our titles with an MLP of all appropriate continuous and categorical metadata into a final CNN model is not something we had ever seen with text data before. Initially, we wanted to use LDA to narrow down the data to one estimated category based on the title of a video before running it through our mixed model as sub data since the category Id variable provided to us was uninterpretable. We didn't use LDA in the final version of the model because, after mapping the category Id variable to the corresponding category name, we no longer needed to cluster to infer category. Moving forward, we can escalate this project to support YouTube users looking for predictive analytics. For example, similar to how services such as TubeBuddy operate, C-level content creators looking to find out how long until their video will trend can make use of this project. In its most refined form, this project would operate as a third party add on to the current YouTube analytics page.

References

[1] Choe, Min, et al. ”How Long Will Your Videos Remain Popular? Empirical Study of the Impact of Video Features on YouTube Trending Using Deep Learning Methodologies.” Workshop on E-Business. Springer, Cham, 2018.

[2] Niture, Aakash Ashok. Predictive analysis of YouTube trending videos using Machine Learning. Diss. Dublin Business School, 2021.

[3] Rangaswamy, Shanta, et al. ”Metadata extraction and classification of YouTube videos using sentiment analysis.” 2016 IEEE International Carnahan Conference on Security Technology (ICCST). IEEE, 2016.