University of California, Santa Barbara

Spark Machine Learning: Trending Study of YouTube Videos

By

Sam Leathers
Howard Deng
Jacqueline Zawada
Yuchen Zheng

Academic Advisor: Professor Adam Tashman

March 18, 2021

I. Abstract

In this project, we used YouTube data to predict which videos in the comedy and entertainment categories will be on the trending list. Therefore, the response variable in our models was the number of days that a video was listed on the YouTube trending page. The features we mainly considered in our models were the words in the title of the trending videos. We used a variety of the feature transformers from ML packages of Pyspark to preprocess our data and create our models, which included N Gram, Countvectorizer, Vector Assembler, Tokenizer, and Stop Words Remover etc. We constructed Linear Regression models with these different transformers to process the data and predict the output. Using the MSE and adjusted R square as verification standards, we found out that our N Gram model with feature expansion was the champion model since it had relatively better results. However, the overall results were not very satisfactory. Due to the time limit of the project, we would need to temporarily stop our further analysis. But we would continually explore the data for constructing better models in the future.

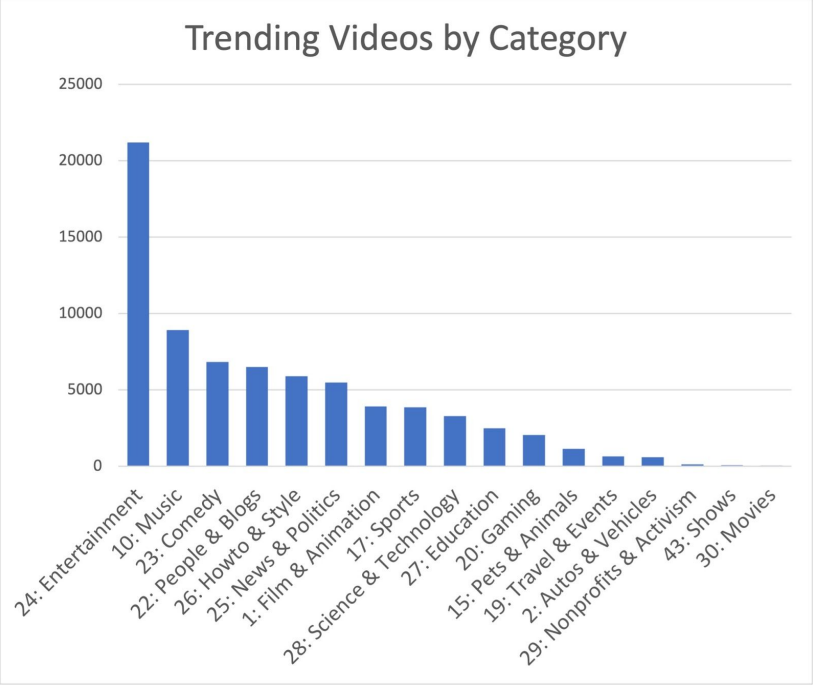
II. Data and Methods

Background of Data

We utilized the "Trending YouTube Video Statistics" found on Kaggle in order to predict the number of trending days a video shows up on the YouTube trending page. According to YouTube, the trending page is aimed to help viewers get an overview of what is happening on the platform. Trending is not personalized, so within each country, every individual's trending list will be the same. Due to the fact that trending videos differ between nations, the datasets are divided by geographic region, and our group chose to work with Canada and the United States. The data is a daily record of the trending YouTube videos, which are identified by a unique video ID. The combined US and Canada dataset has 81,854 rows and 16 columns which give information such as trending date, video title, category ID, and number of views, likes, and dislikes.

Data Cleaning

To begin cleaning, we first combined all of the data we downloaded from Kaggle. There were four files we used: The Canada trending data, the US trending data, and two files that gave the information for the category ID information of each country. We used 'textFile' instead of 'read.csv' to read the Youtube data file because we thought rdd was easier for us to organize and filter out. After combining this data into one dataframe, the next step was to check if all the observations were read in correctly. We found that 8,941 observations were incorrectly read into the dataframe. Since it is only a small portion of our entire dataset, we decided to remove them from the dataframe. After the removal, there are a total of 72,913 observations in our dataframe. Lastly we chose to work with only the videos listed under "Comedy and Entertainment" so the data was substed to select the unique videos within the category. In the final step of cleaning, there were 10,223 observations.



From this bar chart, we could see that the top 3 categories of trending were Entertainment, Music, and Comedy. Excluding music, we were interested in Entertainment and Comedy only in our analysis.

Variables

We kept the **video_id** column to identify unique trending videos. The response variable in our model was **days_trending** which represents the number of days that each video was on the trending list from 12.01.2017 to 05.31.2018. We created this variable by counting the **trending_date** column of the dataset. We considered CA's and US's 'trending dates' were unique, which meant if a video trends in both countries on the same day, that was considered 2 days, introducing a bias towards videos that trend in both. We specifically looked at how the title and the contents of a video contribute to the number of trending days for a video. The variable **tags** give us information about the contents of a video. We also wanted to see if the time of the day a video being published would be a significant factor. Hence, the variables that we considered valuable for the analysis were:

- Days_trending
- Day_segment: 0-morning, 1-noon, 2-evening (feature variable that explained in the data preprocessing section)
- Tags
- Title

Exploratory Analysis

After exploring the dataset, we found out that the trending date of videos were collected from 205 days of trending (from 12.01.2017 to 05.31.2018). And there were 24,685 different videos on the trending list, but with a total count of 72,913, which meant that some of the videos remained on the trending list more than one day. Also, we found out that the unique count of titles was greater than the unique count of the video IDs. That was because some of the videos had different titles. In other words, the titles of the videos were changed while the videos were on the trending list.

We examined the time that the videos were posted during a day. About 48% of the videos on the trending list were posted from 11 am to 6 pm, and about 41% of the videos were posted in the evening (7pm - midnight) and early morning (midnight - 5am). We looked at which words that were most frequently used in the title. We saw that the top five most frequent words in the trending videos' titles were "Episode", "2018," "الحلقة", "Drama," and "TV," which ignoring the words, such as "The", "a", and symbols, such as "-" and "|" etc. The top 5 most used tags for videos in the category Entertainment and Comedy are "funny", "comedy", "talk show", "entertainment", and "humor".

Data Preprocessing

<u>Feature engineering: Time of day posted</u>

We split the variable **publish_time** into Morning (0400-1000) with index 0, noon (1100 - 1800) with index 1, and evening (1900-0300) with index 2. Given we had sliced only by HOUR, each 00 implied 00 - 59 above. And we called this new variable **day_segment**.
After that, we applied One Hot Encoder to map such categorical features into represented label indexes, which was our **day_segment2.**

Pipeline 1: Base Model Cleaning

This pipeline is for our first benchmark model, which we created a pipeline for 1 gram of words. We first used the tokenizer to split the title strings into word lists. After that, we removed the filler words, such as “i”, “the” etc., by using StopWordsRemover. Finally, we used CountVectorizer to convert the collection of words into vectors of token counts. The idea was to get a sense of a benchmark for projecting days trending.

Pipeline 2: N-Gram

Pipeline 2 was defined as a function called **build_ngrams** with parameters of input column and n tokens for some integer n. This pipeline also mainly applied to the variable **title**. Initially, we assigned n = 5 for our N Grams processing. Then we set the N-gram class to transform the words from the titles into a sequence of 1-5 tokens. After that, we used CountVectorizer to extract all the N Grams up to 5 into vectors of token counts. And lastly, we used VectorAssembler to transform all the token counts with different amounts of grams into a single feature vector.

Additional Pipelines: Feature Expansion

All other pipelines were created by adding extra features to our N-Gram pipeline with the same parameters. For instance, we ignored the terms that appear in less than 1% of the documents, and added in tokenized tags as a feature.

III. Models and Results

Before creating our models, we split our data into train and test sets. For the benchmark model we had 70% of our data allocated for training and 30% for testing. And for the rest of the models we had 60% for training and 40% for testing. We used the RandomSplit() function to perform this task. We constructed multiple models with the response variable **days_trending**, which provides the number of days that the video appeared on the trending page.

Model 1: Benchmark Model 1

For our first benchmark model, we used **title** as our only feature. The first step was to remove and filter out all the symbols in titles. Then we transformed the dataset by our Pipeline 1. Once we had our transformed data, we built the model with Linear Regression. Finally, our benchmark model had a **MSE of 40.768**, and a **R squared value of -2.9739**. The evaluation of the model showed us that this model was largely a useless model. In hindsight, this likely should have been our first sign that a new methodology was needed, however, we opted for feature engineering and NLP techniques in an attempt to improve our model.

Model 2: Benchmark model + day_segment

Based on the result we got from our benchmark model 1, we decided to see if there was an improvement by adding a new feature of **day_segment**. Accordingly, we used VectorAssembler to combine the day_segment feature with the title token count feature into a single vector of features. However, this model had an increased **MSE of 46.708**, and a decreased **R squared value of -3.36078**. Because of this, we decided to let go this model, and use N Grams as our new feature transformers for modeling.

Model 3: Benchmark Model 2 - N Grams

This was our second benchmark model, and we applied pipeline 2 to transform the features and build the model. The main differences were that we took out the StopWordsRemover from pipeline 1, and simply looked at the ngrams up to 5. However, the running time of the model needed a longer time, which was approximately 1 hour 28 minutes 12 seconds on our record. And the result of the model showed the **MSE of 10.062** and the **R Squared of 0.019198**. This was a rapid improvement compared to the previous models we had however still far from useful.

Model 4: Exploring NLP

In Model 4, we applied pipeline 2 with an additional setting of *minDF = 0.01* in CountVectorizer. The minDF setting was to ignore the terms that appear in less than 1% of the documents. We got a new **MSE of 9.585** and a new **R Squared of 0.0558613**. So we could see that Model 4 had improved from Model 3 while drastically reducing the number of features and run time to train the model.

Model 5: NLP Stopword Exploration

In Model 5, we applied pipeline 2 with another n-gram class to transform the column of filtered words with the original n-gram class for n-gram greater than 2. The setting of *minDF = 0.01* also applied on the model. The result of this model showed a new **MSE of 9.793** and a new **R Squared of 0.0353693**. The MSE of this model had slightly been increased, likely indicating that we removed useful information by removing the stopwords for ngrams less than 3.

Model 6: Feature Expansion

In Model 6, we applied pipeline 2, reverted back to including stopwords for all n, and included **day_segment2**. MSE = 9.610, R Squared = 0.0534279, again no improvement with the addition of **day_segment2** even when being treated as factors. Online ads can largely be affected by time posted, however it appears the nature of trending content is much more reliant on the content itself.

Model 4.1: Explore Regression Settings

This was an extension of Model 4. Given we had shown that adding in the day segment had not helped our model so much nor had using the filtered words, we tried to optimize the MSE with internal settings. In the result, we had a **MSE ridge of 9.558** with a **R Squared ridge of 0.0585086**, and a **MSE lasso of 9.893** with a **R Squared lasso of 0.0255109**.

Model 6.2

This was an extension of Model 6 with additional features of **day_segment2** and **tags_count**. In the result, we had a **MSE ridge of 9.479** and **R Squared of 0.0663276**.

Additional models

<u>Chi Square Selection</u>

Using the same data split as Model 6.2, we further subset our data using **ChiSqSelector** and then ran our ridge regression model to get **MSE ridge: 9.602, R Squared = 0.05419.** Noting there was a slight loss of information with irrelevant gains to the model run time we chose to continue using minDF as our main feature selector to avoid noise.

Random Forest Regression

In this model, we fit a random forest regression model with filtered title n-grams (up to 5), tokenized tags and we ignored terms that appear in less than 1% of the documents. This model had an Root Mean Squared Error (RMSE) on test data of 3.10032, which implied a **MSE of 9.612**, and an **R Squared of 0.0532232**.

Gradient boosted Tree Regression

In this model, we fit a gradient boosted tree regression model with the same features as the random forest regression model. We obtained a Root Mean Squared Error (RMSE) on test data of 3.18205, which implied a **MSE of 10.125**, and an **R Squared of 0.00265264**.

Model Comparison

Model	Description	MSE	R^2
1	Benchmark 1	40.768	-2.9739
2	Benchm1 + day_segment	46.708	-3.36078
3	Benchmark 2 - NGrams	10.062	0.019198
4	Exploring NLP	9.585	0.0558613
5	NLP Stopword expansion	9.793	0.0353693
6	Feature Expansion	9.610	0.0534279
4.1	Explore Regression Settings	9.558	0.0585086
6.2	Add features to model 6	9.479	0.0663276
Random Forest	Random Forest Regression	9.612	0.0532232
Boosted Tree	Gradient Boosted Tree Regression	10.125	0.00265264

From the table above, we compared the results and selected Model 6.2 as our Champion Model because it had the smallest MSE and largest R Square among the models.

IV. Conclusion

We set out to examine what features result in a video showing up on the YouTube trending page. It could be valuable information for YouTube to predict, or potentially an individual looking to start a channel. Anticipating the trending and viral videos on their platform would allow someone to organize marketing opportunities, like creating merchandise and preplanning advertisements. Thus, creating more hype for already popular videos and drawing more customers to their platform or channel.

Our best model is the Linear Regression Model with N-Grams in features of title, day_segment2, and tags_count. Although it had the smallest MSE and largest R Square among our models, we believed that there must be a better model since the dataset still had a lot of space for exploration and research. In general, it appeared that there was overlapping information from titles and tags. Moreover, there were abundant features that we could not have a chance to investigate due to the restriction of system configuration.

Looking forward, a regression model is likely not the path forward and would likely be outperformed by a finely tuned logistic regression seeking videos that trend more than some arbitrary number of days. Given the discrete nature of days and the distribution of the number of days trending, nearly 3 quarters trend for 1 day, a logistic model could be ideal for identifying videos that may trend for more than 5 days. Having a model which could identify potential ‘leads’ in terms of content, from the current list of trending videos, could be of some value for YouTube or a content creator. Ideally, should such a model contain an acceptable level of error, the model could be optimized into a streaming system which monitors trending videos daily, creates leads for review, and updates the model automatically through some form of built-in cross-validation to keep the model up to date with current trends.