
Predicting Popularity of YouTube Video Midway Report

Loc Do

Heinz College, Carnegie Mellon University
Pittsburgh PA, 15213
halocd@andrew.cmu.edu

Joseph Richardson

School of Computer Science, Carnegie Mellon University
Pittsburgh PA, 15213
jmrichar@andrew.cmu.edu

Forewords from the authors

As a result of our feedback on the proposal, as well as follow-up meetings with our TA mentor, Anthony, we propose some changes in the scope of our project. There are two main reasons: 1) The new goal can be qualitatively and quantitatively measured with existing metrics and 2) These changes are made to ensure our project would fit well to the course's theme.

Our previous plan was to generate a pandora-radio-like playlist based on a user input video, where the list was supposed to appeal to a viewer who enjoyed the seed video. However, this task poses several challenges in evaluation metrics, which are not trivial given our current available sources such as time, efforts, etc. Viewer opinion is highly subjective and expensive to obtain. Hence, to make an objective judgment on effectiveness of our algorithms, we would require a significant amount of user study. Two possible solutions are 1) run a real services and record the users' feedback on the system and 2) using Amazon Mechanical Turk to evaluate the result. Both solutions are either costly or time-consuming, and not practical for this project's scope.

Yet another idea, not involving human objects but less objective, is to compare our results with YouTube's recommended videos (located at the sidebar on each video's site). Although it is reasonable to do so – since algorithmic outputs from YouTube recommender systems can be treated as a proxy to reflect majority's preferences – we only observe a small samples of them at a single point of time, and YouTube caters their recommended outputs regarding to each user's history view. It would be hard to judge the correctness of our algorithms and it's highest possible achievement would be mere imitation of an existing service; we would no longer be predicting the true tastes of YouTube users.

Our new goal is to make predictions concerning the popularity (number of views, percentage of likes, and percentage of dislikes) of a youtube video given the rest of its metadata. We formulate the goal as a ranking prediction, which can be efficiently measured by popular methods such as Recall@K, etc., and yet should still involve plenty of machine learning and make use of topics covered during the course. Details are described in following sections, which are organised as follows. First, we briefly introduce the ranking problem in the YouTube settings in Section 1. Section 2 describes some of plausible directions and our first-steps to solve the problem. Experimental study including basic statistics on dataset, as well as evaluation metrics is mentioned in Section 4. Related work is summarised in Section 5. Finally, we conclude this report with a list of to-do things and our stretch goals.

1 Introduction

YouTube is one of the most popular video-sharing online platform in the Internet. It attracts billion of unique user visit monthly and has diverse topics in their video content. YouTube users can earn money from the number of views of their uploaded videos. Hence, understanding the secrets of making a popular YouTube video is essential for people who want to make benefits from the site. There are many factors to determine popularity of a video, but in general we can pin down to have the following two: having good content and good marketing plan. One of the techniques to attract more viewerships is to make the video's "visual appearance" look appealing such as catchy keywords, informative cover picture, etc. Our approach is to utilise a video's metadata to predict their popularity.

Ranking problem (aka. Learning to rank¹) is a typical supervised learning problem of predicting the rank of a set of items regarding to a set of criteria. It has a numerous applications in a broad domains such as web search, multimedia retrieval, recommender systems, etc. Results from these such problems can bring benefits to Internet users such as saving their time by introducing the most relevant products/articles/web pages to their interests. In YouTube context, ranking problems can be raised to recommend most relevant videos to a given video. Another application is to predict the ranking of videos w.r.t to their popularity. This can reveal the most important visual factors in determine a popularity of a video.

Problem statement. Given a set of videos, each is associated with a set of bag-of-word and numeric features. A pair of videos is said to have an order on their popularity by comparing their number of views. Video with more viewership is considered to be more popular than the other. Given two videos with their features, the ranking problem here is to construct a model to well predict the exact order between the two videos.



2 Proposed solutions

2.1 Logistic Regression on binary classification

We can reformulate the ranking problem between two videos as a binary classification. To be specific, let $\mathcal{X}_u \in \mathbb{R}^d$ and $\mathcal{X}_v \in \mathbb{R}^d$ represent features of video u and v . We can form a representative vector of the two as follows

$$\mathcal{X}_{uv} = f(\mathcal{X}_u, \mathcal{X}_v), \quad (1)$$

where $f : (\mathcal{R}^d, \mathcal{R}^d) \rightarrow \mathcal{R}^k$ is a transformation function. We have several options for the transformation function f .

- Difference between two feature vectors: $\mathcal{X}_{uv} = \mathcal{X}_u - \mathcal{X}_v$
- Concatenation of two feature vectors: $\mathcal{X}_{uv} = [\mathcal{X}_u, \mathcal{X}_v]$ (Matlab notation)
- Kernel functions, e.g. $\mathcal{X}_{uv} = \|\mathcal{X}_u - \mathcal{X}_v\|^2$

At the moment, we cannot find any theories/signals to indicate which form of f is the most appropriate. Therefore, we plan to implement all of them and compare their performance empirically.

Let define Y_{uv} , the label associated with the video pair (u, v) , as follow

$$Y_{uv} = \begin{cases} 1, & \text{if \#_of_view_u} \geq \text{\#_of_view_v} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

We can assume the function form of classifier $P(Y_{uv}|\mathcal{X}_{uv})$ as follows

$$P(Y_{uv} = 0|\mathcal{X}_{uv}) = \frac{1}{1 + \exp(w_0 + \sum_i w_i \mathcal{X}_{uv}^i)} \quad (3)$$

The model parameters w can be learnt efficiently using Gradient Descent algorithm.



¹http://en.wikipedia.org/wiki/Learning_to_rank

2.2 Linear Regression on number of views

Another approach to the ranking problem is to predict the number of views from a video's metadata (excluding the actual views) and use the predicted number to determine which video is more popular. This can be treated as a simple linear regression problem. Again, our linear function can be written in the functional form of feature vectors as input X_u plus some noise ϵ

$$Y_u = \beta X_u + \epsilon \quad (4)$$

We can use the closed form or Gradient Descent to learn the β parameters. After the learning stage, the predicted ranking can be done as follows

$$\hat{Y}_{uv} = \mathbb{I}(\beta X_u > \beta X_v), \quad (5)$$

where \mathbb{I} is the indicator function, return 1 if the expression as argument is true, and 0 otherwise.

3 Evaluation Metrics

Some evaluation measures that are widely used to evaluate ranking approaches are 0-1 loss function, and Area Under Curve (AUC). **0-1 loss function** is the ratio of correctly ordered video pairs over total number of pairs in testing set.

$$0/1_{loss} = \sum_{(u,v)_{test}} \frac{1}{|(u,v)_{test}|} \mathbf{1}[\hat{Y}_{uv} - Y_{uv} \neq 0] \quad (6)$$

Since our label values in $\{0, 1\}$, we use **AUC Loss** as another ranking-based performance metric.

$$AUC_{loss} = 1 - AUC, \quad (7)$$

where AUC is the area under the ROC curve.



3.1 Comparing Order-of-Magnitude

For the purpose of this midway report, however, we are making direct predictions, not comparisons, and we require, at least for now, some other evaluation metric.

It is important to decide what we will consider as being "close to correct". 0-1 loss is sufficient for our comparison-based version, and for our predictors of the percentage of likes and dislikes, we can simply consider our loss in terms of the square of the difference between our prediction and the true value. When predicting the number of views, however, we must deal with the gigantic variance in our observed data.

Ideally, we wish to consider orders of magnitude rather than direct counts, and for this we will set our loss function equal to the square of the difference between the log of our prediction and the log of the observed value. The motivation is that we wish to reflect the human intuition that there is more difference between the popularities of two videos with 10 and 1,000 views (respectively) than between two videos with 1,000,000 and 1,001,000. This will prevent petty among between the most popular videos from drowning out the differences in all others.

The decision to use a log-number-of-views-based loss function may be reversed at a later time if we find better results without it.

Currently we are using a linear regression to predict the number of views, and we deal with the number of views in log scale for the regression. One anticipated effect of this is that features will be expected to contribute multiplicatively, rather than additively, to the popularity of a video. While this certainly seems interesting, it is something that we may ultimately change our minds about as we hone our results to finer detail. As our model grows more sophisticated than mere linear regression, it may be possible to achieve multiplicative effects when needed even while working directly with the number of views rather than its log.

4 First Steps

4.1 Dataset


We implemented our own crawler in Java and started to collect information from YouTube since October, 1st, 2014. Our crawling strategy is to initialize the crawler with several random "seed" videos, which mostly are in the *Movie* and *Music* category, , and recursively explores all other videos that YouTube suggests as being related to that videos. For each video, we extract all of its metadata such as title, uploader, description, upload date, number of views/likes/dislikes, video length, and a number of other attributes, as well as a list of around 30 videos YouTube recommends as being similar.

4.1.1 Preliminary Data Statistics

Although the crawler was suspended twice due to technical issues and upgrades, thus far we have gathered a decent amount of data:

- Number of videos crawled: 335,373
- Number of uploaders: 146,655
- Most viewed video: 2,107,560,304 views.
- Most "liked" video: 8,647,905 likes.
- Most "disliked" video: 4,184,459 dislikes.
- Size of "bag of words" dictionary produced for the titles: 129,553 entries.

As the statistics show, we have a large magnitude in ranges of number of view, likes and dislikes. This motivates us to do some preprocessing on data, such as feature normalisation or data standardization, to ensure the numerical stability and good speed of convergence on the learning algorithm. We discuss more on this step in the Section ??.

We plot in Figure 1 the distribution of number of views in our  data. Interestingly, the distribution is in the shape of Gaussians, instead of following the Power Law distribution as frequently observed in social network. We guess this observation is due to the way we sample the data, by following the recommended links, and imply that popularity of a video is one of the highly-impacted factors in YouTube recommender systems. Whether we can uncover the underlying reasons of links suggested by YouTube, namely, based on similarity or popularity or both, is an interesting question and may be addressed in our future work.

4.2 First Regression

The data gathering required significant time, and is a big part of the project, but apart from this we had to change most of our plans regarding first steps. Therefore, what we will present in this report is the results of our first attempt to predict the popularity of a video. For this, we have considered the data from just five days of crawling, and we have reduced the number of fields considered to the title and uploader. We consider only the number-predicting version, not the direct-comparison-prediction version.

We perform a linear regression on the log of the number of views ("Y") and some features extracted from our video data ("X"), to find the best β for $Y = \beta * X$. We also use the log of the number of views as our loss function, rather than using the number of views directly.

We then randomly select 80% of our data for training, and reserve the other 20% for testing. Once the training is finished, we run the other 20% of our data into the trained model, and consider how much loss was observed.

4.2.1 Feature extraction

The first step is to build a dictionary mapping the uploader to the number of videos they have uploaded and the total number of views there videos have. We also take care to prevent "cheating": In order to ensure that our predictor has only such information as would be available before the

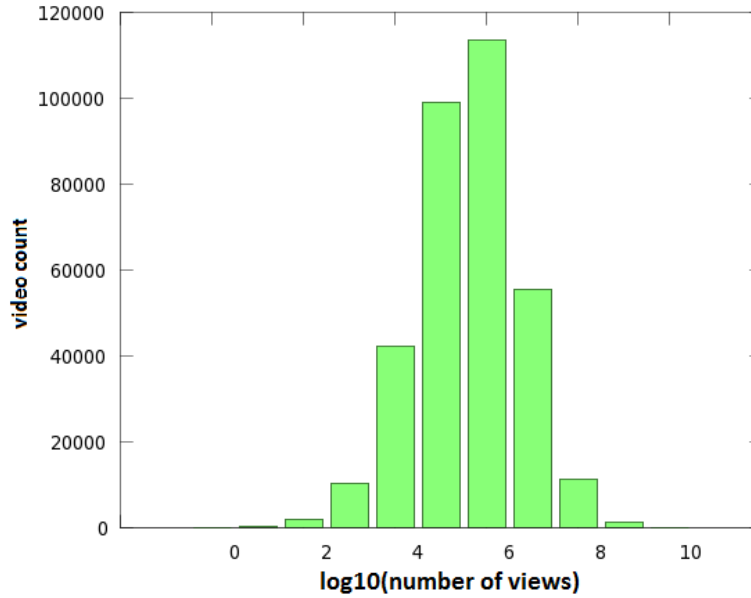


Figure 1: Histogram on the distribution of number of views (in log-scale).

video’s publishing is ever used, we temporarily reduce these number of video-views and the total number of video uploads for the uploader according to the publish date of the video under current consideration.

We train a linear regression model for each of our three outputs on the following features:

- Many features extracted via a bag-of-words model on the title, using TF-IDF.
- The # of videos uploaded by the uploader prior to the current video’s upload date.
- The total # of views for an uploader due to videos released prior to the current video’s upload date.
- The fraction of the previous two features (the average number of views per video for videos uploaded by the same uploader prior to the current video’s upload date).

We also desired to add features for the upload date and video runtime, but since those are stored as region-dependent strings we have had some trouble extracting them on time for this mid-way report.

4.2.2 Results

Preliminary results were, as expected, very poor: Our log-based predictor only was, on average, within two orders of magnitude of the true value, and our not-log-based predictor had an average loss of 60 million. This is not surprising, considering that our feature selection is, for now, very primitive.



5 Related work

- Local Collaborative Ranking [1]
- Logits model for sets of ranked items [2]
- AdaRank [3]



To be updated fully later with brief summary on each work

6 Conclusion

6.1 To-do list

There is one additional piece of information that we decided our crawler should collect, which still needs to be added: the number of subscribers for each user. Since this was not needed for our original proposal, we will need to go back and gather this information, which may take some time considering the vast quantity of videos crawled.

6.2 Stretch Goals

We certainly hope to attempt increasingly sophisticated learning techniques to reduce our loss as much as possible. Time allowing, there are also other interesting results we can pursue. Chief among these in our mind is to make more time-dependent predictions. It may be, for example, that video A is very popular at first, but that video B maintains its popularity better over time, eventually overtaking video A in terms of the numbers of views and of likes.



To be specific, we will collect the number of views for a given video over time. Having these temporal factors can help us to construct time-sensitive weights, which can estimate the rate of popularity of a video given its metadata.

Acknowledgments



Our sincere thanks to Anthony for his patience and value inputs to improve our project.

References

- [1] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon and Yoram Singer. Local Collaborative Ranking. Proceedings of the 23rd International World Wide Web Conference (WWW) 2014
- [2] Allison, Paul D., and Nicholas A. Christakis. "Logit models for sets of ranked items." Sociological methodology 24.1994 (1994): 199-228.
- [3] Xu, Jun, and Hang Li. "Adarank: a boosting algorithm for information retrieval." Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007.