Predicting Popularity of YouTube Video

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

Abstract

In this work we address the problem of ranking two objects given their features. The problem is applied to a context of guessing the more popular of two given YouTube videos. We formulate this problem into two different approaches, namely binary classification and linear regression. Experiments are conducted in our data set crawled from YouTube over a one-month span. Results show that both approaches yield a slightly above 50% accuracy. We conclude that bag-of-words features with linear classifiers are not efficient in determining videos' popularity.

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 had content for a wide variety of topics. 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 use YouTube professionally. There are many factors to determine popularity of a video, but in general we can reduce them down to having good content and a good marketing plan. One of the techniques to attract more views is to make the video's "visual appearance" look appealing via catchy keywords, informative cover picture, etc.

The ranking problem (aka. Learning to rank[1]) is a typical supervised learning problem of predicting the rank of a set of items regarding some set of criteria. It has a numerous applications in a broad domains such as web search, multimedia retrieval, recommender systems, etc. Results from such problems can bring benefits to Internet users, such as saving their time by introducing only the content most relevant to their interests. In YouTube context, ranking problems arise in recommending the most relevant videos to a given video. Another application is to rank video popularity overall.

Problem statement. Given a set of videos, each is associated with a set of bag-of-word and numeric features extracted from their metadata. We use the number of views as our measure of popularity. Given two videos with their features, the video ranking problem is to construct a model to accurately predict which one has more views. In order to do this, we approach the problem in two different manners: Ranking by classification, and ranking by regression.

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

2 Ranking by Classification

Our goal is to construct a prediction rule f that can rank the videos with respect to their popularity using their meta-data as feature inputs. Since there are only two outcomes in ranking two videos, we choose to model the output of f as a binary class label and use binary classification, as described in Section 2.1. This approach poses two problems. First, it does not take into account the magnitude in the ranking, meaning there is no difference between the class of a pair of highly dissimilar viewcounts and a pair whose viewcount difference is small. We address this problem by borrowing ideas of re-weighted training set from the Boosting technique in Section 2.2. Second, video viewership also depends on the video's "age", i.e. number of days between when it was uploaded to YouTube to when it was crawled. An older video may appear much more popular in comparison to a newer one simply because it has been around longer. Therefore, we bin videos according to their "age", and learn a distinct classifier f for each group. We also construct an ensemble of these time-specific classifiers in order to enhance the overall predictive performance. This is described in Section 2.3.

2.1 Problem Formulation

Let $X_i \in \mathbb{R}^D$ and $X_j \in \mathbb{R}^D$ be feature vectors of two videos i and j correspondingly. Each video pair (i, j) is associated with a binary label Y_{ij} defined as follows

$$Y_{ij} = \begin{cases} 1, & \text{if \#\_of\_views\_i} \geq \text{\#\_of\_views\_j} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

We can form a representative vector of the pair (i, j) as follows

$$\mathcal{X}_{ij} = k(X_i, X_j), \tag{2}$$

where $k : (\mathcal{R}^D, \mathcal{R}^D) \rightarrow \mathcal{R}^{D'}$ is a feature transformation function. There are several options for k.

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

As we have as of yet not found any theories/signals to indicate which form of k is the most appropriate, we choose to represent X_{ij} as difference between X_i and X_j, meaning $D = D'$. A kernelized version is left to future work.

The functional form of classifier $P(Y_{ij}|\mathcal{X}_{ij}, \mathbf{w})$ is as follows

$$P(Y_{ij} = 1|\mathcal{X}_{ij}, \mathbf{w}) = \frac{1}{1 + \exp(w_0 + \sum_d w_d \mathcal{X}_{ij}^d)} = \frac{1}{1 + \exp(\mathbf{w}^T \mathcal{X}_{ij})} \tag{3}$$

The model parameters $\mathbf{w} \in \mathbb{R}^D$ can be learned using MAP.

$$\begin{aligned} \hat{\mathbf{w}}_{MAP} &= \arg\max_{\mathbf{w}} \prod_{(i,j)} P(Y_{ij}|\mathcal{X}_{ij}, \mathbf{w}) P(\mathbf{w}) \qquad (P(\mathbf{w}) \sim \mathcal{N}(0, \tau^2 I)) \\ &= \arg\max_{\mathbf{w}} \prod_{\{(i,j)|Y_{ij}=1\}} P(Y_{ij}|\mathcal{X}_{ij}, \mathbf{w}) P(\mathbf{w}) \qquad (Y_{ij} + Y_{ji} = 1, \mathcal{X}_{ij} = -\mathcal{X}_{ji}) \\ &= \arg\max_{\mathbf{w}} \sum_{\{(i,j)|Y_{ij}=1\}} lnP(Y_{ij}|\mathcal{X}_{ij}, \mathbf{w}) + lnP(\mathbf{w}) \\ &= \arg\max_{\mathbf{w}} \sum_{\{(i,j)|Y_{ij}=1\}} ((1 - Y_{ij})\mathbf{w}^T \mathcal{X}_{ij} - ln(1 + \exp(\mathbf{w}^T \mathcal{X}_{ij}))) - \lambda_w \|\mathbf{w}\|_2^2 = l(\mathbf{w}) \end{aligned} \tag{4}$$

We can optimize Equation 4 (a concave function) using the Gradient Ascent algorithm with the following update rule

$$w_d^{t+1} \leftarrow w_d^t + \eta \frac{\partial l(\mathbf{w})}{\partial w_d} = w_d^t + \eta \Big(\sum_{\{(i,j)|Y_{ij}=1\}} \mathcal{X}_{ij}^d ((1 - Y_{ij}) - \frac{\exp(\mathbf{w}^T \mathcal{X}_{ij})}{1 + \exp(\mathbf{w}^T \mathcal{X}_{ij})}) - \lambda_w w_d \Big) \tag{5}$$

Since the size of training set is less than the number of features, we opt to using Stochastic Gradient Ascent algorithm (described in Algorithm 1) with L2-regularization.

$$w_d^{t+1} = w_d^t + \eta(\mathcal{X}_{ij}^d((1 - Y_{ij}) - \frac{\exp(\mathbf{w}^T \mathcal{X}_{ij})}{1 + \exp(\mathbf{w}^T \mathcal{X}_{ij})}) - \lambda_w w_d) \tag{6}$$

Algorithm 1: Stochastic Gradient Ascent Algorithm

Initialize **w** as zero-valued vector.
Initialize $epoch = 1$.
Initialize $\lambda = 0.01$.
Initialize $\eta = 1$.
while *epoch* $<$ *maxIter* **do**
  for $\forall i, j$ **do**
    Update **w** using Equation 7
  epoch++;
Return **w**.

2.2 Extension 1: Re-weighting the features

By definition in Equation 1, Y_{ij} can only capture which video has more viewerships, but not how much their viewerships differ. Hence, the classifier f may have a difficult time correctly weighting the important features in determining video's popularity. Assume that we have three YouTube videos v_1, v_2 and v_3, each attracted 1000, 10 and 1 views after one day since they were uploaded to the web. Also assume that we only use bag-of-word features and there are five words in our dictionary $\{w_1, w_2, w_3, w_4, w_5\}$. The following table contains all necessary information for this example.

Video	Title	Number of views	Bag of word features
v_1	$\{w_1, w_2, w_3\}$	1000	$\{1, 1, 1, 0, 0\}$
v_2	$\{w_2, w_3, w_4\}$	10	$\{0, 1, 1, 1, 0\}$
v_3	$\{w_3, w_4, w_5\}$	1	$\{0, 0, 1, 1, 1\}$

Using the scheme from Section 2.1, we can represent the pair (v_1, v_2) and (v_2, v_3) with feature vectors $\mathcal{X}_{12} = \{1, 0, 0, -1, 0\}$ and $\mathcal{X}_{23} = \{0, 1, 0, 0, -1\}$. When training a classifier f on these two vectors with both Y_{12} and Y_{23} equal to 1, we can see no differences between the weights (i.e. model parameters) on feature w_1 and w_2, w_4 and w_5. Hence, f cannot correctly rank a pair of videos with titles of $\{w_1, w_4\}$ and $\{w_2, w_5\}$.

We wish to somehow incorporate the magnitude in difference between two videos' viewcounts into f, so that we can have more weight on features that are frequently present in popular videos. In this work we propose two ad-hoc solutions: 1) Augmenting the representative feature vectors $\mathcal{X}_{ij}$ and 2) Re-weighting the gradient.

2.2.1 Augmenting the representative feature vectors

In this approach, we scale the representative feature vectors $\mathcal{X}_{ij}$ for pairs of videos (i, j) in the training set, and train a classifier f on these augmented features. The scaling factor is determined based on the ratio of numbers of views of the corresponding video pair i and j. To avoid the overflow problem caused by high variance in number of views (2 billions versus 10), we transform the number of views into log space before computing the ratio. Let consider the pair $\mathcal{X}_{12}$ in the example above. The scaling factor is computed as $\alpha = \frac{\log 1000}{\log 10} = 3$. Hence the augmented representative features $\mathcal{X}_{12} = \{3, 0, 0, -3, 0\}$.

There are various alternate methods to compute the scaling factor α such as using a different log base, or normalizing all the view counts, etc. However, most of these approaches are akin to the proposed one, and do not have any additional theoretical benefits for overall performance, so we demonstrate only our current log base 10 method in this work.

2.2.2 Re-weighting the gradient

Another way to tackle the above problem is to re-weighting the gradient. As before, we compute a scaling factor α from the two videos' viewcounts. However, rather than augmenting the representative feature vectors $\mathcal{X}_{ij}$, we now re-weight the gradient in Equation 6 as follows

$$w_d^{t+1} = w_d^t + \eta\alpha(\mathcal{X}_{ij}^d((1 - Y_{ij}) - \frac{\exp(\mathbf{w}^T\mathcal{X}_{ij})}{1 + \exp(\mathbf{w}^T\mathcal{X}_{ij})}) - \lambda_w w_d) \tag{7}$$

The idea is borrowed from the AdaBoost technique by re-weighting the training data points. However, the difference is that AdaBoost keeps updating weights for every iteration, while we fix the weight. Our goal is to stress the importance of high-viewcount-variance pairs by forcing the corresponding parameters to update more elements of the gradient for these pairs.

2.3 Extension 2: Ensemble of time-specific classifiers

The number of views for a video depends largely on time passed since it was first uploaded. It is more comparable to measure the popularity between videos sharing the same age. Such videos are clustered into a bin and a distinct classifier f is trained on them. For the above example, we have 3 different bins, $B_{1000} = \{v_1\}$, $B_{10} = \{v_2\}$ and $B_1 = \{v_3\}$, with corresponding classifiers f_{1000}, f_{10} and f_1. Given a new video pair v_4 and v_5, we first find the bin for each video according to their respective ages, and then use the corresponding classifiers to compare their popularity.

This approach, however does suffer from data sparseness, which is a serious drawback. Since our data is a one-month sample from YouTube repository, it is possible to have keywords do not exist in the observed video pairs of a bin. To reduce this problem, we considered an ensemble of all bins' classifiers constructed to enhance the predictive performance.

According to [4], ensemble methods are learning algorithms that blend results from different hypotheses to perform some prediction tasks on new data points. There are two main reasons behind these methods. The first is statistical: Basically, we often do not have sufficient data to identify the best among several equally accurate hypotheses. By taking the majority votes or average results of these hypotheses, we can reduce the risk of choosing the wrong hypothesis. Secondly, different hypotheses may have different starting points and explore different local optima. Hence an ensemble of these hypotheses can give a better approximation than any of individual hypothesis. In the literature, Bagging and Boosting are two common ensemble algorithms. We therefore make an ensemble of bin-specific classifiers with selection of the majority weighted voting scheme.

Because a keyword may not exist in an arbitrary bin's training data, we can borrow information from those other bins where it does in order to compute an acceptable overall weight w. Second, to incorporate the fact that the video's viewcount growth rate will be diminishing over time, the weights from bins further away from the selected bin must have smaller impact than more immediate neighbors. We therefore introduce a majority weighted voting scheme as follow. Let f_t be the classifier trained on data of bin B_t, containing videos of t-days old. Assume that we have T such bins. For a pair of videos i, j in the bin B_t's testing set, we compute the probability of ranking as follows

$$\begin{aligned} P(Y_{ij}^t = 1|\mathcal{X}_{ij}^t, f_1, \ldots, f_T) &= \sum_{t'} \frac{1}{1 + |t - t'|} P(Y_{ij}^t = 1|\mathcal{X}_{ij}^t, f_{t'}) \\ &= \sum_{t'} \frac{1}{1 + |t - t'|} P(Y_{ij}^t = 1|\mathcal{X}_{ij}^t, \mathbf{w}^{t'}) \end{aligned} \tag{8}$$

We can compute $P(Y_{ij}^t = 0|\mathcal{X}_{ij}^t, f_1, \ldots, f_T)$ similarly. The class with higher probability is selected as final prediction.

3 Ranking by Regression

Another approach to the ranking problem is to predict the number of views for each video and then rank the videos based on predicted values, rather than comparing two videos directly. We anticipated

that this method will perform worse than logistic regression at the ranking problem (since our logistic regression pertains precicely to that problem), but has two advantages: Firstly, it can more easily take into account the magnitude of difference between two videos' viewcounts. Secondly, this method has the added utility of allowing predictions on a video without needing to compare it to another, allowing us to directly anticipate popularity for some specific time in future given only a video's current status.

3.1 Problem Formulation

Our linear function assumes that are labels Y_i come from our input X_i plus some noise ϵ:

$$Y_u = X_u\beta + \epsilon, \tag{9}$$

where Y_i is the number of views of video i and X_i is the associated feature vector. We therefore seek a function of the form

$$f(X) = X\beta \tag{10}$$

and attempt to minimize the mean squared error loss function, giving

$$\hat{\beta} = \arg\min_{\beta} 1/n(A\beta - Y)^T(A\beta - Y) \tag{11}$$

where $A = [X_1...X_n]^T$ and $Y = [Y_1...Y_n]^T$, n is the number of training data points.

In order to solve this problem, we can use either the closed form or Gradient Descent to learn the β parameters. However, since our feature space may be quite large, we opt for the latter. We therefore initialize β^0 to 0, and thereafter use the update step

$$\beta^{t+1} = \beta^t - \eta A^T(A\beta^t - Y) \tag{12}$$

Because the conditioning is poor, we opt to use Stochastic Gradient Descent with L2-Regularization, just as we did in the classification-based version. The new update function is

$$\beta^{t+1} = \beta^t - \eta(x_i(x_i\beta^t - Y_i) + \beta^t) \tag{13}$$

After the learning stage is complete, the predicted ranking can be easily obtained from our regression problem via the equation

$$\hat{Y}_{uv} = \mathbb{I}(\beta X_u > \beta X_v), \tag{14}$$

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

3.2 Comparing Order-of-Magnitude

It is important to decide what we will consider as being "close to correct". We use least-squared regression, where we minimize the total of the squares of the differences between our prediction and the true value. 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 views. This will prevent less important (and yet larger in magnitude) variations among the most popular videos from drowning out the differences in all others.

We therefore 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. As an aside, it must be noted that we also tried running our analysis without using log scale, and we failed to observe the differences we anticipated between the two techniques.

4 Experimental study

4.1 Dataset

We crawled and extracted our data from YouTube for one month (Oct - Nov, 2015). The crawling strategy was a bread-first search, with the crawler initialized to several random "seed" videos, which mostly are in the *Movie* and *Music* categories. New videos to explore were found by following all likes to related videos, as recommended by YouTube's standard recommendation system. 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. We later crawled the channel data for all channels that uploaded videos we had crawled, in order to obtain the number of subscriptions.

4.1.1 Data Statistics

Although the crawler was suspended twice due to technical issues and upgrades, we have gathered a sizeable amount of data, with tremendous variation in the observed values:

- Number of videos crawled: 1,432,213
- Number of uploaders: 628,072
- Number of "bag of word" features (extracted from the titles): 2,447,603 entries
- Video length: $[1 \dots 107,373]$ (seconds)
- Range of number of day passed since upload: $[1 \dots 3423]$ (days)
- Range of number of views: $[7 \dots 2,104,518,656]$.
- Range of number of likes: $[1 \dots 8,639,650]$.
- Range of number of dislikes: $[1 \dots 4,184,769]$.

We plot in Figure 1 different histograms reflecting characteristics of our data. In the red-colored histogram, we look for the distributions of videos and their view counts. Interestingly, the distribution is in the shape of a Gaussian in log scale, instead of following the Power Law distribution that is frequently observed in social networks. We guess this observation is due to the way we sampled the data, by following the recommended links. It is intuitive since videos with similar view counts are usually recommended together.

The green-colored histogram presents the distribution of uploaders and their number of uploaded videos. Most of the YouTube users upload around 158 videos. A small portion of them have uploaded more than 3,000 videos. The blue-colored histogram shows the number of videos in each of our bins. As mentioned above, we picked the first 30 bins, which accumulate a significant enough number of videos. One of our stretch goals would be to run through the whole dataset at a future time; however, given our current results, we believe that some technique alterations may be more useful than having additional data.

As can be easily seen, we have very large ranges of number of views, and we measured large variations in the number of likes and dislikes as well. This motivated us to do some preprocessing on data, such as feature normalization or data standardization, to ensure the numerical stability and good speed of convergence on the learning algorithm for logistic regression. One example of a decision influenced by this can be seen in Section 3.2.

4.1.2 Feature extraction

In this section we discuss the feature engineering process in the project. First, we built a dictionary mapping each uploader to the number of videos they have uploaded and the total number of views for their videos. We also took care to prevent "cheating": In order to ensure that our predictor has only such information as would be available before the video is published, we temporarily reduce the 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 considered many features, which ultimately include:

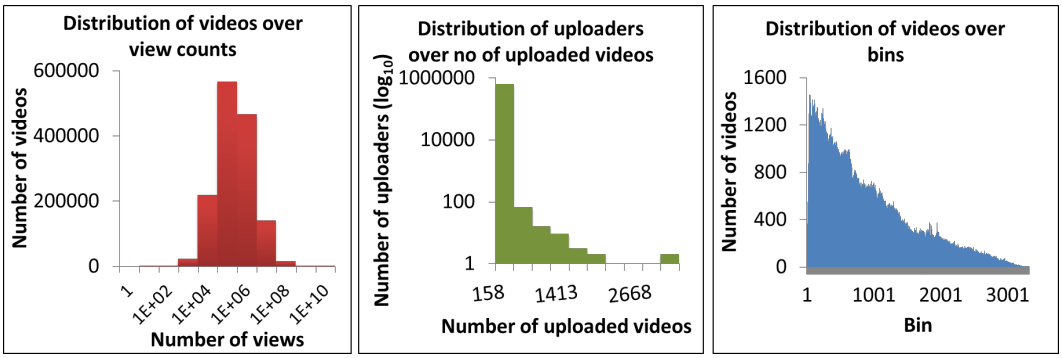


Figure 1: Histograms indicating the distribution vedeos and uploaders with respect to viewcount, upload count, and bin number.

- Features extracted via a bag-of-words model on the title, using TF-IDF.
- The number of videos uploaded by the uploader prior to the current video's upload date. Because of our desire for caution against "cheating", we count only those videos that we have crawled.
- The total number of views for the uploader due to videos released prior to the current video's upload date. Again, we count only those videos that we have crawled. This date-conscious counting is particularly important because there are many cases where there is only one video crawled for a given uploader, meaning that this feature would become a nearly perfect predictor for such videos.
- The number of subscribers for the uploader. We lack sufficient data to know how subscription counts changed over time, so we simply had to keep this constant.
- The runtime of the video, in seconds.
- The age of the video at the time of crawling, in days.
- The number of likes/dislikes. Since these are expected to scale with the number of views, we forbade ourselves from using the number directly, but we did allow certain combinations, such as the log of the like-vs-dislike ratio.
- Various combinations of the above features (for example, the log of some other feature, or the ratio between two features).

4.2 Evaluation Metrics

Normally, a rank correlation [2] is a common metric in ranking problems. Since our problem is reformulated as a classification and regression problems, we evaluate our methods using corresponding metrics such as Root Mean Square Error (RMSE) for Regression and F_1-score and Accuracy for Classification.

4.2.1 Classification

The error matrix (or confusion matrix) is a common way to measure performance of any arbitrary classifiers. Several metrics derived from this matrix are Accuracy, Precision-Recall, F_1-score or Receiver Operator Characteristics (ROC). In our work we employ two of them: Accuracy and F_1-score [3].

The accuracy can be equivalently stated as a 0-1 loss function over total number of pairs in testing set.

$$0/1_{loss} = \sum_{(u,v)_{test}} \frac{1}{|(u,v)_{test}|} \mathbf{1}[\hat{Y}_{uv} - Y_{uv} == 0] \tag{15}$$

[2] http://en.wikipedia.org/wiki/Rank_correlation

[3] http://en.wikipedia.org/wiki/F1_score

F_1-score is defined as follows

$$F_1 = 2\frac{precision * recall}{precision + recall}, \tag{16}$$

where

$$\begin{aligned} precision &= \frac{\sum_{(u,v)_{test}} \mathbf{1}[\hat{Y}_{uv} = Y_{uv} = 1]}{\sum_{(u,v)_{test}} \mathbf{1}[\hat{Y}_{uv} = Y_{uv} = 1] + \mathbf{1}[\hat{Y}_{uv} = 0 \wedge Y_{uv} = 1]} \\ recall &= \frac{\sum_{(u,v)_{test}} \mathbf{1}[\hat{Y}_{uv} = Y_{uv} = 1]}{\sum_{(u,v)_{test}} \mathbf{1}[\hat{Y}_{uv} = Y_{uv} = 1] + \mathbf{1}[\hat{Y}_{uv} = 1 \wedge Y_{uv} = 0]} \end{aligned} \tag{17}$$

4.2.2 Regression

The linear regression can be evaluated in more than one way.

The first way to measure our results is to look at its predictions directly. Our RMSE can be found via

$$RMSE = \sqrt{\frac{\sum_i (Y_i - X_i)^2}{n}} \tag{18}$$

The second form of evaluation is to compare the results for each pair of videos and then use the **0-1 loss function**, so that results can be compared to the logistic regression results.

4.3 Results

Due to our limitation in computing resources, we can only present our approaches for the first 30 bins (meaning videos with age ranging from 1 day to 30 days). For each bin, we create 5-fold cross-validation with 80% of our data for training, and the other 20% for testing. The average results over these are reported.

4.3.1 Classification Results

The accuracy and F_1-score of our logistic regression are presented in Table 1 and Table 2 correspondingly. For each metric, we present four approaches: original logistic regression (ORG), logistic regression with augmented features (AUG), logistic regression with weighed gradients (GRAD) and logistic regression with both extensions (BOTH). Each approach we compare with its corresponding ensemble version as described in section 2.3.

Although all the approaches return barely above 50%, there are some observations. First, the extension with augmented features have better performance than original approach. Meanwhile, the weighted-gradient extension is actually less effective at prediction. That makes senses since we put more weights on the gradients, and thus need much better control on the learning rate to avoid having parameters would oscillate between both side of the optimal values. It also explains why the 'BOTH' method performs the worst in accuracy and F_1-scores among the methods explored.

Table 1: Average of Accuracy over 5 folds of 30 bins.

Approach	Single Predictor	Ensemble
BOTH	0.513	**0.526**
GRAD	0.514	**0.527**
AUG	0.516	**0.534**
ORG	0.514	**0.532**

Most of the ensemble methods have better performance (both in accuracy and F_1-scores) than single predictions. That supports our hypothesis that by partitioning the videos into bins, we worsen the sparseness in each bin's data. Hence by using an weighted combination of all classifiers, we can pass information between the bins to achieve better performance. Recall that the ensemble methods work only if all the individual predictors have good performance in their own data. Since 'BOTH'

Table 2: Average of F_1-score over 5 folds of 30 bins.

Approach	Single Predictor	Ensemble
BOTH	**0.468**	0.463
GRAD	**0.451**	0.443
AUG	0.473	**0.503**
ORG	0.455	**0.468**

and 'GRAD' have lower accuracy in prediction, ensemble methods do not work when using them as basic classfiers. That explains the interesting observation that F_1-scores are better for single predictors in these cases.

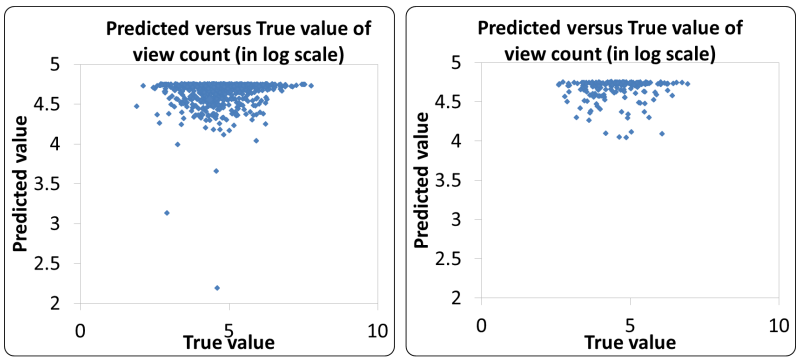
4.3.2 Regression Results

When evaluating our predictions directly, we found an RMSE of 0.93, indicating that we were on average off of the true value by just under an order of magnitude. Without the context from the **0-1 loss function**, however, it is hard to say whether we should be impressed.

The **0-1 loss function** allowed us to compare this method's results to those of the logistic regression method. Our **0-1 loss function** results were not significantly any greater than 50%. While it is not surprising to us that linear regression performs more poorly than classification for the ranking problem, we are surprised that our accuracy is so comparable to that of random guessing, indicating that this technique yielded no measurable value for the ranking problem.

A likely cause is that this problem may not be linear in nature – in fact, it would admittedly be surprising if it were. We tested our accuracy on the training data as well as on the testing data, and found that it performed just as poorly in each, indicating that the weak results on our testing data did not stem from over-fitting.

In order to investigate further, we plotted our output against the true number of views (for a random subset of our data):



a) Training Set b) Testing Set

Figure 2: True value vs prediction (in log-scale).

We observe a strong tendency towards the same output value (roughly 50,000 views), which corresponds to the average number of views. This matches the typical outcome when linear regression is inadequate: failing to match trends, the average is selected nearly always in order to minimize loss.

5 Related work

Our problem is a special case of the more general **Learning to rank** problem, which is widely used in many applications such as Information Retrieval, Data Mining, etc. [1] gives a short survey of

state-of-the-art approaches to the problem of ranking documents relevant to user queries. Most of them are implemented in Ranklib[4].

Another form of the problem is ordinal classification (a.k.a. ordinal regression). There are several approaches, mostly adapted from classification techniques, such as support vectors [2], Gaussian processes [3], etc. Ordinal classification is related to the ranking problem in that both involve predicting the objects' orders, and is, like ranking, a supervised learning tasks. The important difference is in the level of orderings. As [1] discussed, learning to rank cares more about the accurate ordering of objects, while the latter focuses on a categorization that happens to be orderable. For example, ordinal regression would try to categorize movies into "one star" through "five stars" categories, but there are no specific rankings between movies with the same number of stars, while the learning to rank algorithm has an absolute ordering between all members.

6 Conclusion

Two simple linear approaches, classification and regression, are explored in a context of predicting videos having more view counts. Two types of features are extracted, namely bag-of-word (from videos' title) and numeric features (video's metadata). The slightly above 50% accuracy shows that the features/approaches we selected are not indicative enough to separate the cases. Of our two main method types, classification performed consistently better.

As future work, there are few things we would like to do. First, we want to try out more complicated approaches (e.g. non-linear), such as AdaRank, SVMRank, Gaussian Processes etc. Secondly, we may need to extract more indicative features by exploiting the videos' comments. Thirdly, we want to experiment on all bins, not limited to the first 30 bins.

Acknowledgments

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

References

[1] Hang, L. I. "A short introduction to learning to rank." IEICE TRANSACTIONS on Information and Systems 94.10 (2011): 1854-1862.

[2] Herbrich, Ralf, Thore Graepel, and Klaus Obermayer. "Support vector learning for ordinal regression." (1999): 97-102.

[3] Chu, Wei, and Zoubin Ghahramani. "Gaussian processes for ordinal regression." Journal of Machine Learning Research. 2005.

[4] Dietterich, Thomas G. "Ensemble methods in machine learning." Multiple classifier systems. Springer Berlin Heidelberg, 2000. 1-15.

[4] http://people.cs.umass.edu/ vdang/ranklib.html