Describing and Forecasting Video Access Patterns

GONCA GURSUN MARK CROVELLA IBRAHIM MATTA
Department of Computer Science, Boston University

***Abstract*—Computer systems are increasingly driven by workloads that reflect large-scale social behavior, such as rapid changes in the popularity of media items like videos. Capacity planners and system designers must plan for rapid, massive changes in workloads when such social behavior is a factor. In this paper we make two contributions intended to assist in the design and provisioning of such systems. We analyze an extensive dataset consisting of the daily access counts of hundreds of thousands of YouTube videos. In this dataset, we find that there are two types of videos: those that show rapid changes in popularity, and those that are consistently popular over long time periods. We call these two types *rarely-accessed* and *frequently-accessed* videos, respectively. We observe that most of the videos in our data set clearly fall in one of these two types. For each type of video we ask two questions: first, are there relatively simple models that can describe its daily access patterns? And second, can we use these simple models to predict the number of accesses that a video will have in the near future, as a tool for capacity planning? To answer these questions we develop two different frameworks for characterization and forecasting of access patterns. We show that for frequently-accessed videos, daily access patterns can be extracted via principal component analysis, and used efficiently for forecasting. For rarely-accessed videos, we demonstrate a clustering method that allows one to classify bursts of popularity and use those classifications for forecasting.**

I. INTRODUCTION

Video sharing is one of the most popular applications on the Internet. The largest video sharing site is YouTube, owned by Google Inc. According to [10], approximately 2 billion videos are watched and hundreds of thousands of new videos are uploaded every day. Today, Google generates $6-10\%$ of all Internet traffic and its largest contributor is YouTube [9]. This level of demand makes system design and capacity planning important issues for such sites.

Despite the importance of these issues, very little work has characterized the dynamics of individual video accesses over time. To help fill this gap, this paper makes two contributions. First, we characterize a workload that consists of user accesses to individual videos. Second, we show how to use these characterizations to predict future demand.

To do so, we analyze a dataset consisting of the daily time series of 100,000 YouTube videos. In this dataset, we find that there are two types of videos: those that show rapid changes in popularity, and those that are consistently popular over long time periods. We call these two types of videos *rarely-accessed* and *frequently-accessed* videos, respectively. We observe that most of the videos in our data set fall clearly into one of these two classes. Using this dataset we study two questions: first, are there relatively simple models that can describe the daily access patterns of these two kinds of videos? And second, can we use these simple models to predict the number of accesses that a video will have in the near future as a tool for capacity planning?

Our results show that there are small sets of common patterns that describe frequently-accessed and rarely-accessed videos respectively. We also show how to leverage these small sets of common patterns in order to predict future daily views for individual videos.

For frequently-accessed videos, we show that common patterns can be extracted via principal component analysis. We show that approximately 20 principal components are sufficient to summarize the most popular 1000 videos. We then use these principal components in order to efficiently predict future daily views for each video using autoregressive models. In this way, we show how to efficiently forecast next-day access counts for with low absolute relative error.

For rarely-accessed videos, we employ a clustering method to classify bursts of popularity. We show that approximately 10 classes are sufficient to represent most short-time bursts. Based on these clusters, we introduce a method that combines receiver operating characteristics (ROC) analysis with likelihood ratio testing in order to classify access patterns online, and forecast future access counts. We show that this method can achieve low relative error in forecasting next-day accesses for rarely-accessed videos.

The rest of the paper is organized as follows. We describe our dataset in Section II. We then introduce four key methods in Section III. In Section IV and V we present our main results on frequently-accessed and rarely-accessed videos, respectively. In Section VI we review related work and we summarize our contributions in Section VII.

II. DATASET

One of the strengths of our study derives from our dataset. We obtained it directly from Google, and it represents a global view of video accesses observed at YouTube servers. In contrast to datasets used in YouTube characterization to date, our data set is not restricted by video category (e.g. entertainment or sports) nor by the recommendation system of YouTube. The entire dataset is 326 GB in size and consists of millions of videos. From this large dataset, we select a subset consisting of the most popular 100,000 videos on April 1st, 2008. For each video, the available information is a one year long time series of daily views (from February 25th 2008 to February 25th 2009) and a unique identifier that does not reveal the video's actual name or category. Hence, no metadata on videos is available.

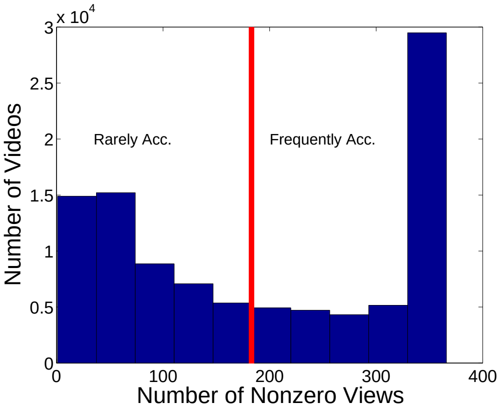


Fig. 1: Histogram of the number of days that have nonzero views.

As already mentioned, we find two different behaviors in video access time series: some videos are consistently popular over long time periods, while the others show rapid changes in terms of popularity and are viewed only on a small number of days. Based on this observation we divide our dataset into two categories: *frequently-accessed videos* and *rarely-accessed videos*, respectively.

Figure 1 illustrates the number of days a video has at least one view. Based on Figure 1, we separate videos into two categories: those that have at least one access on more than half of the days in the year (frequently-accessed) and those that are accessed on less than half of the days in the year (rarely-accessed). Figure 2 shows some examples illustrating the difference between frequently-accessed and rarely-accessed videos. The differences suggest that different characterization and forecasting techniques may work better for each category, which in fact we show in the following sections.

III. Methods

In this section, we briefly introduce the methods used for characterization and forecasting video time series.

A. Singular Value Decomposition (SVD)

For any $m \times n$ real matrix X, there exists a factorization of the following form:

$$X = U\Sigma V^T = \sum_{i=1}^{\min(m,n)} \sigma_i u_i v_i^T$$

where U and V are orthonormal matrices such that $UU^T = I$ and $VV^T = I$. Let u_i and v_i be the i^{th} columns of U and V respectively. Matrix Σ is a $m \times n$ diagonal matrix where each diagonal entry is a singular value, σ_i. The matrix Σ is arranged in such a way that $\sigma_i \geq \sigma_{i+1}$. This factorization is called the *Singular Value Decomposition (SVD)* of X.

One of the most popular applications of SVD is matrix approximation, i.e. approximating a matrix X with another matrix $\tilde{X}$ of lower rank r. To find a matrix $\tilde{X}$ with rank r that minimizes $||X - \tilde{X}||_F$, once can use the SVD of X as follows:[1]

$$\tilde{X} = U\tilde{\Sigma}V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

There are two pre-processing steps that may be applied on matrix X prior to SVD. The first one is mean centering, i.e. subtracting the column mean from each entry. The second is to normalize each entry by the l_2-norm of its column.

B. Autoregressive Moving Average Model

An Autoregressive Moving Average (ARMA) model is one of the most popular methods for modeling and predicting future values of a time series [1]. It consists of two parts: an Autoregressive (AR) model and a Moving Average (MA) model. Given a time series Y, an AR model of order p is defined as:

$$Y_t = \sum_{i=1}^{p} \alpha_i Y_{t-i} + \epsilon \qquad (1)$$

where $\alpha_1, ..., \alpha_p$ are the parameters of the model and ϵ is a white noise error term. An MA model of order q is defined as follows:

$$Y_t = \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \qquad (2)$$

where $\theta_1, ..., \theta_q$ are the parameters of the model and $\epsilon_t, ..., \epsilon_1$ are again white noise error terms. Combining Equations (1) and (2), an ARMA model of order (p, q) is written as follows:

$$Y_t = \sum_{i=1}^{p} \alpha_i Y_{t-i} + \epsilon_t + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \qquad (3)$$

The error terms, ϵ_t, are generally assumed to be Gaussian i.i.d. random variables with zero mean and constant variance.

C. Hierarchical Clustering

Clustering is an unsupervised learning technique. It groups data objects in such a way that the objects in the same cluster are more similar than are objects in different clusters. There are a variety of clustering algorithms in the literature; popular choices are hierarchical clustering and K-means [8]. In this work, we use hierarchical clustering.

Hierarchical clustering creates a tree of clusters where the root represents the set of all objects and each leaf represents an individual object. In this work, we use bottom-up (agglomerative) strategy: starting at the leaves, clusters are merged successively based on pairwise similarity between objects in each cluster. The number of clusters at which merging stops is a tunable parameter. When clustering time series, similarity can be defined in terms of the metrics such as Euclidean or Cosine distance. We use Euclidean distance for clustering, but also make use of Cosine distance (as described below).

[1] The Frobenius norm of an $m \times n$ matrix M is $||M||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} M_{ij}^2}$.

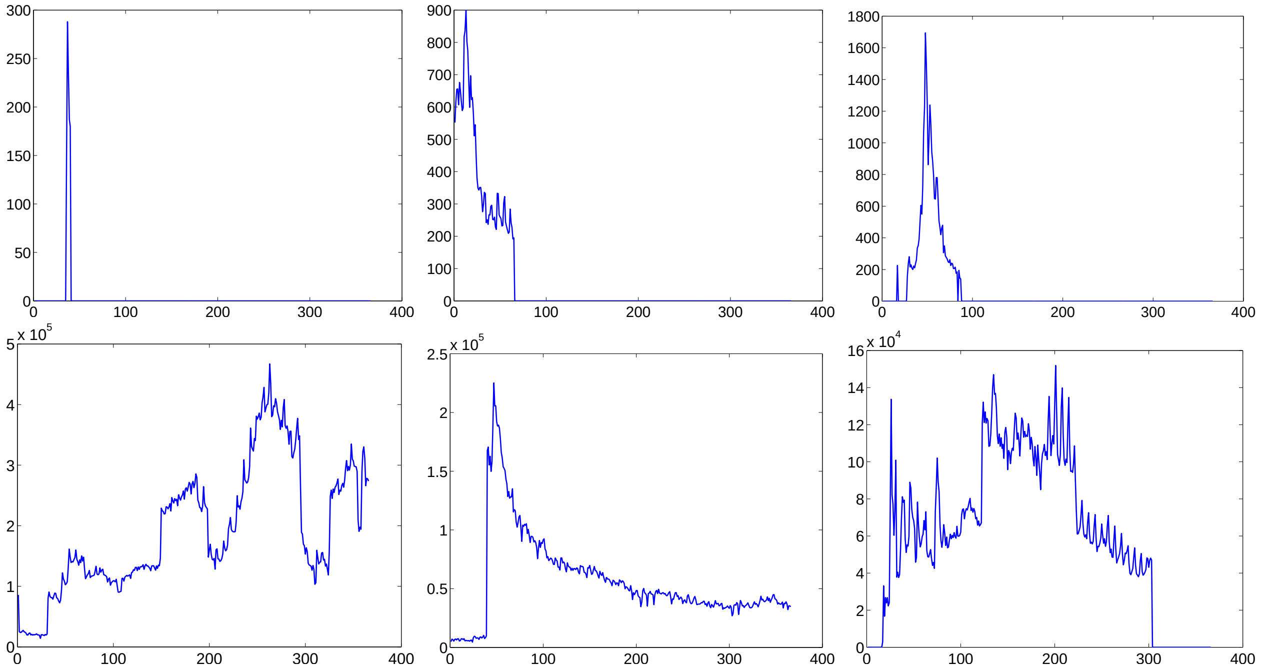


Fig. 2: Example time series of video accesses: rarely-accessed (top row) and frequently-accessed (bottom row). The x axis is time (in days) and the y axis is the number of daily views.

D. ROC Analysis

A Receiver Operating Characteristics (ROC) plot is a visualization of the performance of a classifier. ROC graphs depict the trade-off between true positives and false positives as a detection threshold is varied. We mostly use standard ROC analysis [6] with the only unusual aspect being the extension to multi-class detection. Let $c_1, ..., c_n$ be the classes to which instances are to be mapped. For each class c_i a separate ROC plot is generated as follows: Any instance that is classified c_i is considered positive and any instance that is classified to any other class is considered negative. Further discussion of multi-class ROC plots is available in [6].

IV. Frequently Accessed Videos

We these tools in hand, we can now describe our main results. In this section we analyze frequently-accessed videos, and in the next section we analyze rarely-accessed videos. As mentioned above, frequently-accessed videos are those that are continuously popular during the year, i.e. viewed almost every day. For this analysis, we concentrate on the most popular 1000 videos as measured by the total number of views.

A. Characterization

Our characterization focuses on (1) understanding common patterns in data, and (2) using that understanding as an aid to prediction.

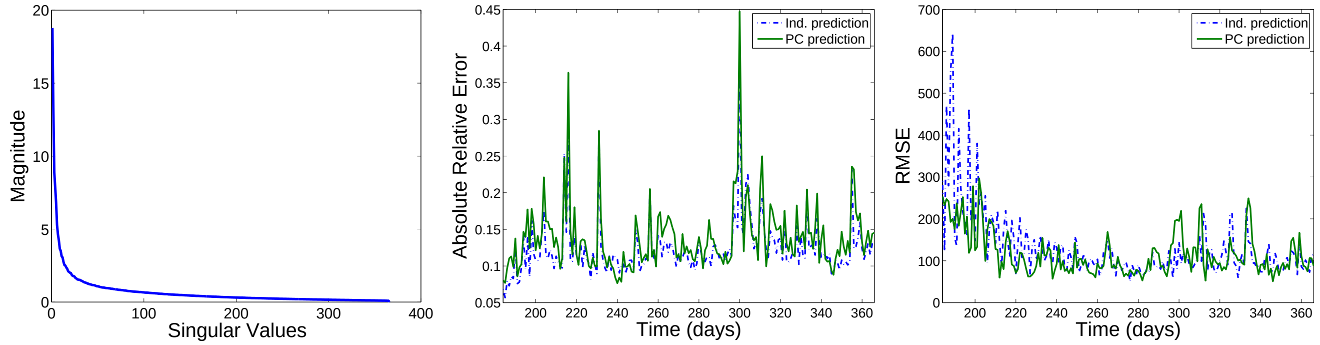
We observe that there are temporal correlations in our frequently-accessed data set. By employing SVD, we decompose the time series into their main constituents. Let X be a 366×1000 matrix, where each column of X is a 366-day time series of a video. Prior to SVD, we mean center and normalize X as explained in Section III-A. Figure 3a demonstrates the magnitudes of the singular values of X. In this figure, it is seen that there is a knee around the 20^{th} singular value. To be more accurate, 88% of energy level is achieved by largest 20 singular values, where total energy is defined as a function of singular values, σ, of X as follows: $\sum_{i=1}^{366} \sigma_i^2$. This suggests that there is a considerable structure and only 20 principal components are enough to approximate our collection of videos.

The largest three principal components are presented in the top row of Figure 4. The first principal component shows a steady increase during the year. The second principal component shows increase until the middle of the year and then decrease. The third component shows two fluctuations during the year. One common behavior in these principal components is that they all show distinct 7-day fluctuations.

In the next section, we show that this compact representation helps efficiently predict the daily accesses that videos receive in the future.

B. Forecasting

As described in Section III-B, one of the most popular techniques for time series ARMA modeling. To assess the utility of ARMA modeling for our data, we first apply this method on each video individually. To set the order of the model (p, q), we use one-day-ahead ARMA predictions with order values ranging from $(4, 4)$ to $(12, 12)$. We find that $(7, 7)$ is the smallest order that yields good results. This suggests that using observations of one week past is sufficient for modeling the behavior of the next day. In fact, this is understandable in light of the weekly fluctuations seen in our time series.



(a) Singular Values of Most Popular 1000 Videos

(b) Absolute Relative Error

(c) RMSE

Fig. 3: Video access characterization and forecasting accuracy.

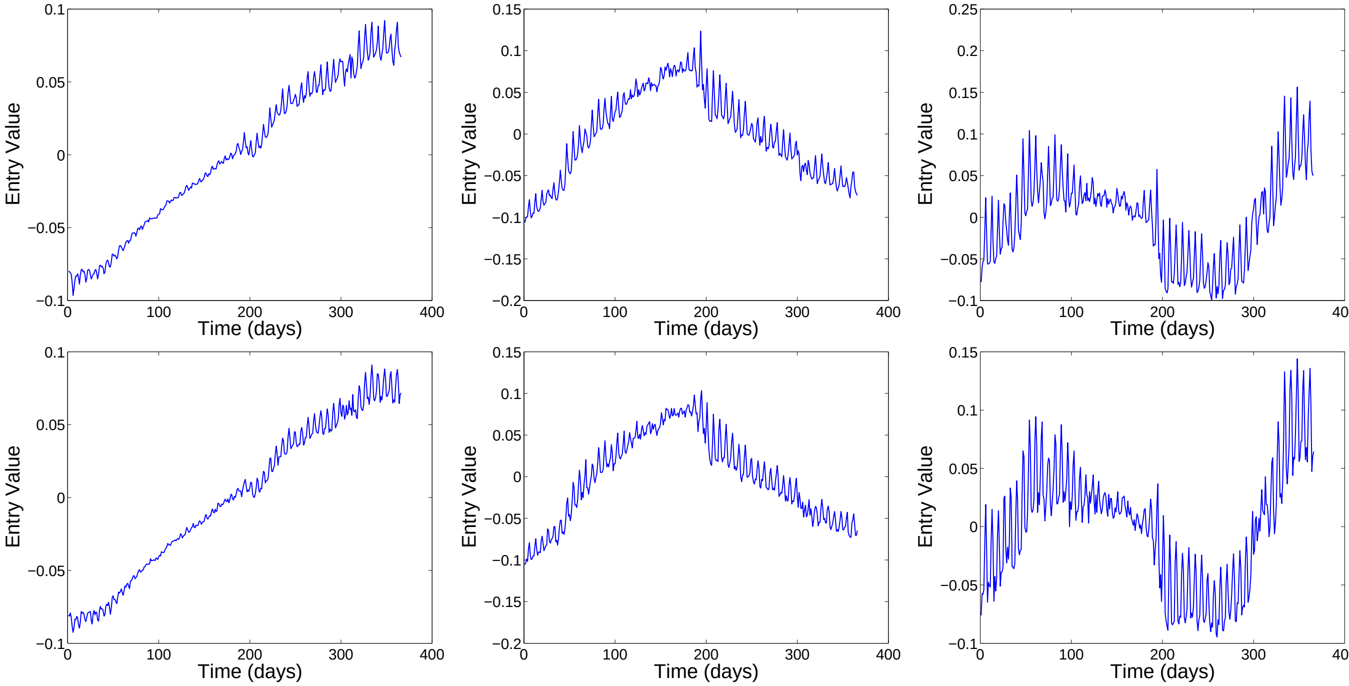


Fig. 4: The largest three principal components (upper) and their ARMA predictions (lower). The x axis is the time (in days) and the y axis is the magnitude of the principal component.

For each time series, we use the first half of the year as the training set for generating an ARMA model. Then for each of the remaining 183 days, we forecast one day ahead.[2]

To define an error metric, let X_{ij} be the true view count and $\hat{X}_{ij}$ be the predicted view count on day i for video j. Then, average absolute relative error is defined as $\frac{1}{N}\sum_{j=1}^{N}\frac{|\hat{X}_{ij}-X_{ij}|}{X_{ij}}$ and average root mean squared error (RMSE) is defined as $\frac{1}{N}\sqrt{\sum_{j=1}^{N}\left(\hat{X}_{ij}-X_{ij}\right)^2}$, where N is the number of videos.

We find that ARMA modeling works successfully for forecasting future daily accesses. The dashed line in Figure 3b and Figure 3c illustrate average absolute relative error, and average RMSE, respectively. Errors are averaged over all videos from day 184 to 366. The average absolute relative error is below 0.15 and RMSE is below 200 for most days.

While this method shows good accuracy, its computational cost is unfortunately high, and scales with the number of videos to forecast, because it requires generating a model for each video separately. Our strategy for making cost managable combines the two observations made so far: approximation via PCA and forecasting with ARMA models. Our approach is to apply ARMA modeling on the principal components of the data instead of the individual time series. In other words, instead of directly forecasting the individual time series, we forecast the principal components, an approach we call *PC forecasting*. Just as for individual time series forecasting, in PC forecasting we use the first 183 days as the training set to generate ARMA models with order $(7, 7)$ and predict one-day ahead for the rest of the year. The bottom row of Figure 4 shows the ARMA predictions of the first three

[2] Note that forecasting more than one day ahead is possible, albeit with less accuracy. We omit this analysis for lack of space.

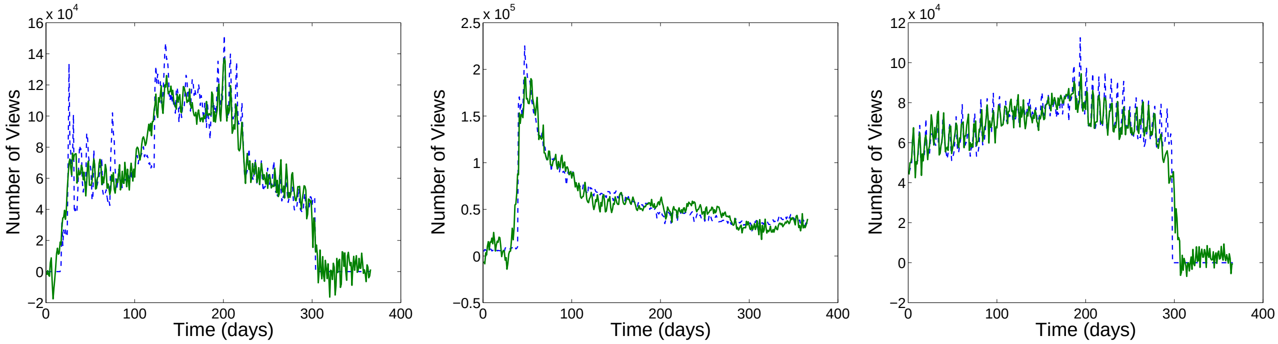


Fig. 5: Examples of one-day ahead PC forecasting. Dashed lines are actual time series and solid lines are PC forecasts.

principal components. As can be seen, ARMA $(7,7)$ models can accurately forecast the principal components.

However, our main goal is predicting not the principal components but the original daily views. This requires transforming PC forecasts into the individual forecasts. Let $X_{1:t}$ be the rows of X from day 1 to day t and $\tilde{X}_{1:t}$ be the pre-processed form of $X_{1:t}$ prior to SVD (see Section III-A). $\tilde{X}_{1:t}$ is decomposed into its $U_{1:t}$, Σ', and V' after SVD is applied. The columns of $U_{1:t}$ are the principal components of matrix $\tilde{X}_{1:t}$ and $\tilde{U}_{1:t}$ represents the first 20 columns of $U_{1:t}$. Initially, an ARMA model is generated for each column of $\tilde{U}_{1:183}$. Then, by using these models, on any day t, $\tilde{U}_{t+1}$ can be computed. At this point, we can approximate $\tilde{X}_{1:t+1}$ as the product of $\tilde{U}_{1:t+1}\Sigma' V'^T$. The last step is to reverse the pre-processing prior to SVD, by converting $\tilde{X}_{1:t+1}$ to $X_{1:t+1}$.

Figure 5 shows some examples of time series and their PC forecasts. It shows that PC forecasting can be very successful in predicting the next day's accesses. To obtain a sense of overall error, Figures 3b and 3c compare the performance of PC forecasting and individual forecasting. In both figures, the dashed line represents the day by day error in individual forecasting and the solid line represents error in the PC forecasting. The x-axis starts from day 184, since we start forecasting on day 183. In Figure 3b, for both individual and PC forecasting, the absolute relative error is quite low. The mean absolute relative error is around 0.12 for individual forecasting and 0.14 for PC forecasting. Figure 3c shows the RMSE on each day. The mean RMSE is about 130 for individual forecasting and 117 for PC forecasting. These values are low given that the daily views of the videos range between the scale of 10^4 - 10^7.

While the increase in error due to PC forecasting is small, the improvement in scalability is large. PC forecasting requires training only 20 ARMA models, a number that is not expected to change significantly as the number of videos modeled grows. Even for only 1000 videos, this is a considerable saving in running time. For example, on a four processor 2.66GHz Intel with 4GB of RAM, running 64-bit Linux, individual forecasting takes 844 seconds, whereas PC forecasting takes 150 seconds to finish. Thus, for our set of 1000 videos, PC forecasting is about 5.5 times faster than individual forecasting.

In sum, we see that exploiting the structure inherent in the data means that, instead of constructing thousands of ARMA models, one only needs to construct a small number of models. This provides a significant improvement in scalability, with very little penalty in accuracy.

V. Rarely Accessed Videos

Rarely-accessed videos are those that are viewed during a small portion of the year, e.g. a couple of days, weeks or months. Approximately 50,000 videos (half of the videos in our data set) fall into this category. Among these 50,000 videos, we randomly choose 10,000 of them as the training set and we use the remaining 40,000 videos as our test set for the methods developed in the next sections.

A. Characterization

Rarely-accessed videos typically show short-time popularity bursts. These bursts may occur for a variety of reasons including information dissemination in social networks (since YouTube is a socially-driven system, and its content can be embedded in online social networking sites). For that reason, identifying the shape of popularity bursts is particularly important. Unlike the time series in the frequently-accessed videos, PCA is not suitable for characterizing rarely-accessed videos since these time series have large amount of zero-valued daily views. Moreover, we are interested in the shape of the bursts themselves instead of the principal components that generate the time series. Hence, in order to reveal these shapes, we use turn to a different dimensionality-reduction method: clustering.

Before clustering, we extract a time window of 64 days from each video around its maximum daily view. We select the maximum (peak) value of each time series as its central value and take the first 31 values before and 32 values after the center, so that maximum value of all time series appears on day 32. Moreover, we normalize each time series by its maximum value so that all maximum values are equal to 1.

After this preprocessing, we cluster time series using hierarchical clustering with the Euclidean distance metric. In order to choose the right number of clusters, we look at how the average standard deviation of time series in the same cluster

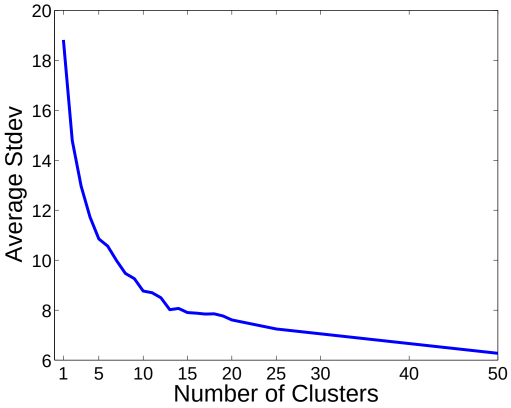


Fig. 6: Average within-cluster standard deviation vs. number of clusters.

varies with different numbers of clusters. Figure 6 shows that approximately 10 clusters are enough to achieve relatively low clustering error.

We use our training set of 10,000 time series to extract the clusters. In Figure 7, the solid lines are the mean of each cluster and the dashed lines show ± one standard deviation. These clusters represent the shapes of typical bursts in video views. For example, clusters 2, 1, and 4 represent videos that are popular only during a couple of days, one week and one month, respectively. In cluster 5, the constant zero views after day 32 may reflect the removal of video from YouTube. Likewise, cluster 7 may represent the group of videos which receives the largest number of views on the day it is uploaded. Unlike other clusters, 10 shows a more stable popularity with only weekly fluctuations. Another notable property is that many of the clusters are approximately pairwise reflections. Clusters 5 and 7, 6 and 8, and 4 and 9 are examples of pairwise reflections. In the next section, we use these clusters as the base models for predicting the future number of daily views for individual videos.

B. Forecasting

In order to use these clusters as a basis for forecasting, we adopt a two-step process. The first step is to map the video to one of the clusters introduced above and the second step is to predict the next day view count based on the mean of the mapped cluster.

1) Mapping to Clusters: Let A be the time series of daily views vector of video x, and let $A_{1:t}$ denote the portion of this time series from day 1 to day t. Our aim is to forecast the number of views that the video x will receive on day $t+1$, denoted by A_{t+1}.

In order to predict A_{t+1}, we start by mapping x to one of the clusters. Let Y be the mean time series of the cluster c_i. Because of the normalization applied to time series before clustering, Y and A will typically be of different scale. Hence we can not simply map $A_{1:t}$ to the cluster whose mean time series returns the minimum Euclidean distance. Instead, we use a metric that is insensitive to absolute magnitude: cosine distance. Treating $A_{1:t}$ and $Y_{1:t}$ as two vectors in t-dimensions we calculate the angle between them, θ_i, via the following equation:

$$\theta_i = \arccos\left(\frac{A_{1:t}}{|A_{1:t}|} \times \frac{Y_{1:t}^T}{|Y_{1:t}|}\right) \tag{4}$$

Given the angle between $A_{1:t}$ and $Y_{1:t}$ for each cluster, we seek the cluster to which video x most likely belongs. In order to do that, we use a likelihood ratio test. First, for each cluster, we construct the empirical distribution of the angle between the mean of the cluster and each training set time series mapped to it. Likewise, we also construct the distribution of the angles between the mean of the cluster and each training set time series that is not mapped to it. Then, given a video that has angle α with cluster c_i, we compute the likelihood ratio for assigning the video to that cluster as:

$$l(i,\alpha) = \frac{p(v \text{ has angle } \alpha \text{ with the mean of } c_i \,|\, v \in c_i)}{p(v \text{ has angle } \alpha \text{ with the mean of } c_i \,|\, v \notin c_i)} \tag{5}$$

Having computed likelihood ratios for all clusters, we assign video x to the cluster c_i that maximizes $l(i,\alpha)$.

For any given video, cluster assignment based on Equation (5) can be be performed for each day t. However, as new daily view values become known, the cluster that a video is mapped to may change. In particular when t is small the results of the likelihood ratio test may not be reliable, resulting in a high false positive rate. To address this, we can put a threshold τ on $l(i,\alpha)$ values so that any cluster which returns a value of $l(i,\alpha)$ less than τ is considered to be an unreliable mapping and can be ignored. We would like to compute values of τ such that for any mapping, the false positive rate is below some acceptable limit; in what follows, we choose the false positive rate limit to be 0.1. The appropriate value of τ will vary with t since as more daily views become known, mapping becomes more accurate. The goal is to bound the false alarm rate, which is most likely to be large for small t.

To compute the proper threshold τ, we use ROC analysis. For a given cluster c_i and threshold τ, the videos whose true cluster is c_i are the set of *total positives* (having size P) and those whose true cluster is not c_i constitute the set of *total negatives* (having size N). As already mentioned, videos having $l(i,\alpha)$ values less than the corresponding τ are not mapped to cluster c_i, even if c_i is the cluster that maximizes $l(i,\alpha)$ — i.e., such videos are ignored because insufficient evidence exists to map them to any cluster. Among those that have $l(i,\alpha) \geq \tau$, the videos whose true clusters are c_i form the set of *true positives* (having size TP) and those whose true clusters are not c_i, are the set of *false positives* (of size FP). Then true positive rate tp is TP/P, and the false positive rate fp is FP/N.

Figure 8 shows the ROC graphs for clusters 6, 8 and 10 as an example. In each ROC plot, there are curves for t values of 5, 25, 30, 35, and 60. Each point on a curve uses a different value of τ; values of τ vary from 0 to 10. The figure shows that as t increases the ratio tp/fp increases, confirming that more accurate mapping is possible with increasing t. By using

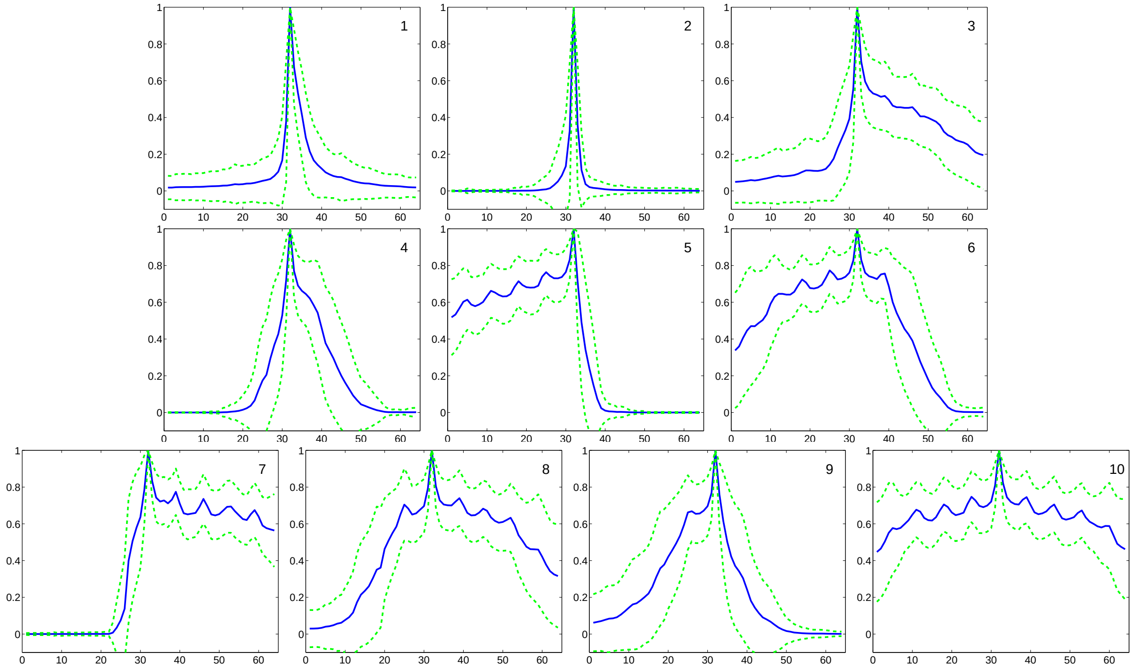


Fig. 7: Mean and ± one standard deviation for each of the 10 clusters extracted from rarely-accessed videos.

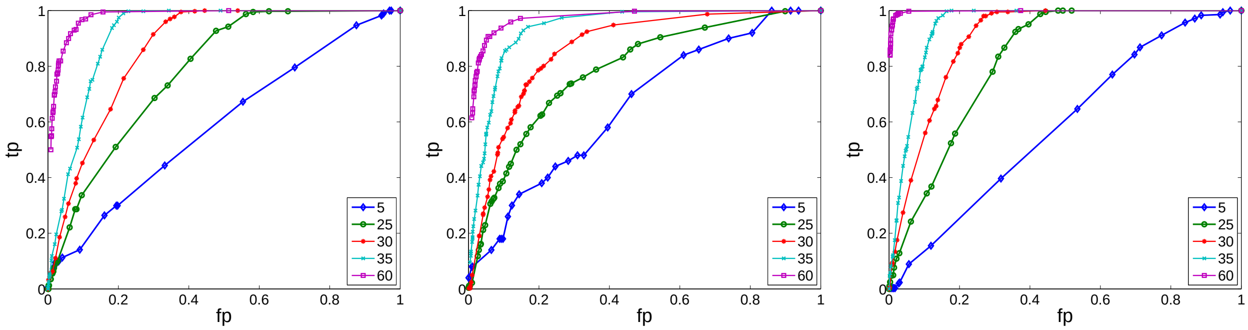


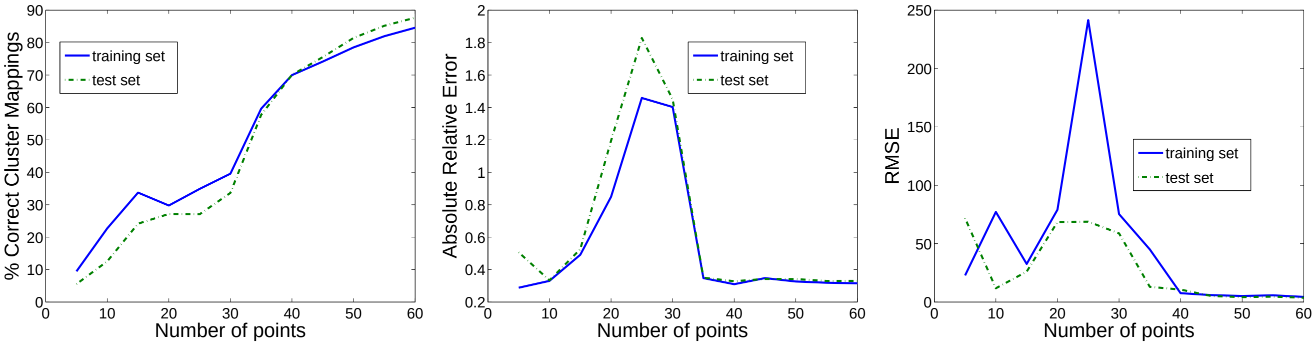
Fig. 8: ROC plot examples for cluster 6 (left), 8 (middle) and 10 (right).

this style of ROC analysis for each cluster c_i and day t, we find the threshold value τ that limits the fp rate to be less than 0.1.

Having extracted the appropriate threshold values via ROC analysis, we can now map each video to the cluster that yields the maximum likelihood ratio among all those that are above thresholds. Figure 9a shows the percentage of correct mappings of videos to clusters. The x-axis is t, the number of days used for prediction. The solid line corresponds to the training set and the dashed line to the test set. The percentage of correct predictions starts around 10$\%$ when we only use the first 5 days of video access counts, and increases up to 90$\%$ when most of the time series is used.

We consider a particularly important problem to be computing a forecast for a video's access count just before its peak. Around day 32, the percentage of correct predictions is around 45%. This is in fact a reasonable success rate for a couple of reasons. First of all, given that we have 10 clusters, random assignment would yield a success rate of only 10%. More importantly, the main reason for the observed success rate is similarity between cluster shapes. For instance, clusters 1, 4 and 7 show very similar trends up until day 32, and are clearly distinguishable from each other only after day 32. The same situation holds for clusters 5, 6, and 10 and also for clusters 8 and 9. Thus, for any $t < 32$ it is reasonable that a video that belongs to cluster 1 may be mapped to cluster 4 or 7. Since we are interested in one-day-ahead forecasting, incorrect mappings to clusters with the similar shape up to the current day does not strongly affect forecasting success.

2) *Next Day Forecasting:* After mapping video x to cluster c_i on day t, the next step is predicting the number of views of the next day (A_{t+1}) by using the scaled mean of the cluster c_i, i.e. Y. Note that Y is computed from the time series which are normalized by their maximum values. Thus the daily view values of Y are in range [0,1], whereas the daily view values in $A_{1:t}$ are the true number of views. In order to bring Y and



(a) Success of Cluster Mapping

(b) Absolute Relative Error

(c) RMSE

Fig. 9: Success rate and resulting forecasting error for cluster mapping.

A to the same scale, we need to estimate a *scaling factor*. This scaling factor is calculated by using the least squares approach. The scaling factor is a constant a that minimizes $||Y_{1:t} - aA_{1:t}||_2$. Having found the scaling factor a, A_{t+1} is then predicted as $A_{t+1} = Y_{t+1}/a$.

Figure 10 shows some examples of the original videos and their predictions for different values of t. In the figure, the dashed line is the original time series and the solid line is its prediction. The dot indicates the day on which the forecasting was done. The forecasts are done on day 25, 30 and 35 from top to bottom row respectively. In the figure, instead of only one day ahead forecasting, the examples show the prediction of the rest of the time series, i.e. $64-t$ day ahead forecasting. This results in a less accurate forecasting compared to the one-day ahead version in which we compute a new cluster and scaling factor on each day. However, even with this less-accurate version of forecasting, Figure 10 illustrates the success of our forecasting method.

Finally, Figures 9b and 9c show the performance of one-day ahead forecasts in terms of mean absolute relative error and RMSE for both the training and the test sets. The figure shows that the days around the peak value are the most difficult to predict accurately. This is to be expected, as the cluster standard deviation is typically higher around the peaks (see Figure 7). The average absolute relative error is 0.56 and 0.65 for training and test sets, respectively. Given that the videos in the test set are not used for constructing the clusters, generating the ROC curves, or computing the thresholds, absolute relative error difference of 0.09 is very low. The average RMSE is 50 and 29 for training and test sets, respectively. The lower RMSE in test set is due to the fact that there are more videos in test set (40,000) than in the training set (10,000) and the few outlier videos that do not fit well in the clusters are averaged out in the test set.

VI. Related Work

Our works relates to a broad spectrum of topics, from workload characterization to compact representation of large data streams. In this section, we briefly mention related work on these topics.

A number of studies have examined the characteristics of user-generated video sharing systems. Among these studies, some specifically focus on YouTube [2], [3], [4], [7]. In [2], Cha et al. analyze the popularity distribution of YouTube videos and how users' requests are distributed across popular and unpopular videos. They also analyze the popularity evolution of videos, i.e., the change in popularity as the videos get older. They show that an unpopular video is unlikely to get popular as it ages. Based on these observations, they suggest that future popularity of videos could be predicted but do not provide a way doing so. In [3], Cheng et al. study statistical properties of YouTube videos such as the distribution across different video categories (e.g. music, sports etc.), video lengths, active life span of videos, and growth trend in uploading new videos. One finding in that paper related to ours is that most videos have a short active life span. This is consistent with the fact that almost 50% of the videos in our dataset are in the rarely accessed category. In [7], Gill et al. characterize YouTube usage from an edge network perspective by studying characteristics such as file size, video durations, video bit rates and usage patterns within a campus network.

Our work differs from these works in several aspects. First, we use a complete and global dataset observed at YouTube servers. Our collection of videos is not biased by the recommendation system of YouTube, a specific group of users, or video categories (e.g. entertainment or sports). Second, our work does not depend on meta information, such as the long it has been since the video was uploaded, its rankings, etc. Most importantly, these previous studies do not propose a framework that can be used for quantitative forecasting.

There are also studies that focus on the social networking aspects of YouTube. In [4] Cheng et al. propose a peer-to-peer short video sharing framework that leverages social networks. In [5], Crane et al. investigate how a social system responds to bursts of exogenous and endogenous activity by using the time series of daily views of YouTube videos, and they find four different shapes for bursts. Their models are relatively simple compared to our clustering-based results, which reveal more complex shapes.

Finally, another set of related work concerns representing large data streams with smaller-sized approximations. This

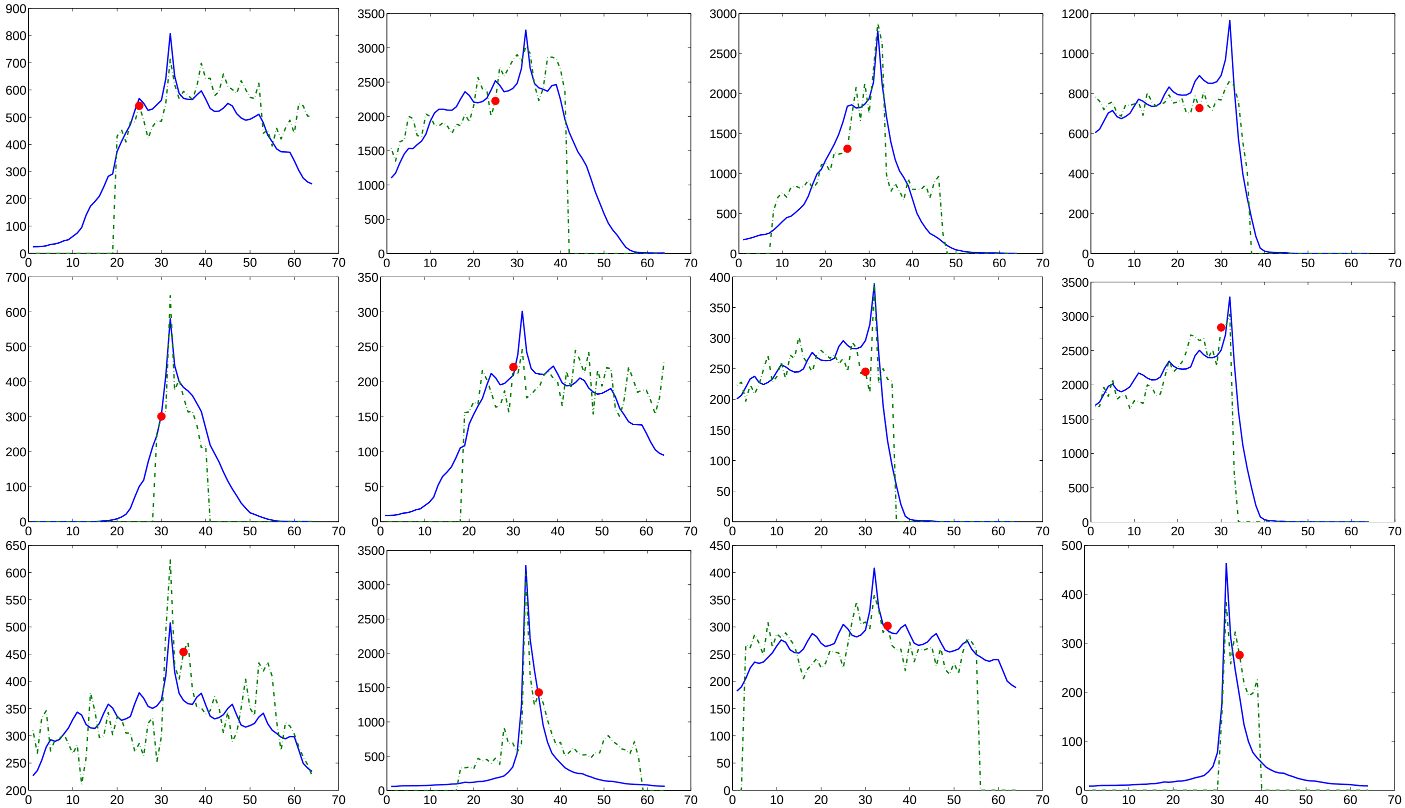


Fig. 10: Prediction Examples: x axis is time in days and y axis is the number of daily accesses. The dot corresponds to the day on which forecasting is made.

is a well studied topic with wide application in the field of data mining. Two typical examples are [11] in which Korn et al. use SVD for data compression of large data sets, and [12] in which Papadimitriou et al. summarize the key trends in data streams by extracting their principal components. In both cases, however, these results are not used as tool for forecasting, but rather as a general form of data compression.

VII. Conclusion

In this paper, we have analyzed a large dataset consisting of daily access counts of hundreds of thousands of YouTube videos. We find that there are two types of videos: those showing rapid changes in popularity, and those that are consistently popular over long time periods. Our study shows that, for both types of videos, there are relatively simple models that can describe the daily access patterns and these simple models can be effectively used to predict the number of accesses that a video will have in the near future. We show that for frequently-accessed videos, daily access patterns can be extracted via principal component analysis, and used efficiently for forecasting via autoregressive models. For rarely-accessed videos, we demonstrate a clustering method that allows one to classify bursts of popularity and use this clustering with ROC analysis for forecasting. Keeping in mind the importance of video-sharing as a traffic driver and workload type in today's Internet, our results represent a useful step towards efficient and effective forecasting for video-sharing sites.

References

[1] P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*. Springer Series in Statistics. Springer-Verlag, second edition, 1991.

[2] M. Cha, H. Kwak, P. Rodriguez, Y. yeol Ahn, and S. Moon. I tube, you tube, everybody tubes: Analyzing the worlds largest user generated content video system. In *In Proceedings of the 5th ACM/USENIX Internet Measurement Conference (IMC07)*, 2007.

[3] X. Cheng, C. Dale, and J. Liu. Understanding the characteristics of internet short video sharing: Youtube as a case study. *CoRR*, abs/0707.3670, 2007.

[4] X. Cheng and J. Liu. Nettube: Exploring social networks for peer-to-peer short video sharing, 2009.

[5] R. Crane and D. Sornette. Robust dynamic classes revealed by measuring the response function of a social system, 2008.

[6] T. Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[7] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. Youtube traffic characterization: a view from the edge. In *IMC*, pages 15–28, New York, NY, USA, 2007.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[9] http://asert.arbornetworks.com/2010/03/how-big-is google/.

[10] http://www.youtube.com/.

[11] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 289–300, New York, NY, USA, 1997. ACM.

[12] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 697–708. VLDB Endowment, 2005.