IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 27, 2017, accepted October 20, 2017, date of publication October 27, 2017, date of current version November 28, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2767104

Big Data Analytics for Program Popularity Prediction in Broadcast TV Industries

CHENGANG ZHU[1], GUANG CHENG[1], (Senior Member, IEEE), AND KUN WANG[2,3], (Senior Member, IEEE)

[1]School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, Nanjing 211189, China
[2]School of IoT, Nanjing University of Posts and Telecommunications, Nanjing, 210003, China
[3]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

Corresponding authors: Guang Cheng (gcheng@njnet.edu.cn) and Kun Wang (kwang@njupt.edu.cn)

This work was supported in part by NSFC under Grant 61572262, in part by the China Postdoctoral Science Foundation under Grant 2017M610252, and in part by the China Postdoctoral Science Special Foundation under Grant 2017T100297.

ABSTRACT The precise and timely prediction of program popularity is of great value for content providers, advertisers, and broadcast TV operators. This information can be beneficial for operators in TV program purchasing decisions and can help advertisers formulate reasonable advertisement investment plans. Moreover, in terms of technical matters, a precise program popularity prediction method can optimize the whole broadcasting system, such as the content delivery network strategy and cache strategy. Several prediction models have been proposed based on video-on-demand (VOD) data from YouKu, YouTube, and Twitter. However, existing prediction methods usually require a large quantity of samples and long training time, and the prediction accuracy is poor for programs that experience a high peak or sharp decrease in popularity. This paper presents our improved prediction approach based on trend detection. First, a dynamic time warping-distance-based K-medoids algorithm is applied to group programs' popularity evolution into four trends. Then, four trend-specific prediction models are built separately using random forests regression. According to the features extracted from an electronic program guide and early viewing records, newly published programs are classified into the four trends by a gradient boosting decision tree. Finally, by combining forecasting values from the trend-specific models and the classification probability, our proposed approach achieves better prediction results. The experimental results on a massive set of real VOD data from the Jiangsu Broadcasting Corporation show that, compared with the existing prediction models, the prediction accuracy is increased by more than 20%, and the forecasting period is effectively shortened.

INDEX TERMS Broadcast TV, popularity prediction, dynamic time warping, random forests regression, gradient boosting decision tree.

I. INTRODUCTION

As the maturity and popularity of high-definition (HD) and 3D technology increase, IP video traffic will become a major part of all consumer Internet traffic. According to data published by Cisco Visual Networking Index in July 2016 [1], the consumption of Internet video streams of broadcast TV will continue to grow at a rapid pace and will constitute 26% of consumer Internet video traffic by 2020. However, user attention is not uniformly distributed among all programs. Only a few programs can attract massive user attention; the remaining programs are left without anybody to watch them. Take Tencent video, for instance [2]. There have been 45 billion cumulative requests for the top-50 programs, which is more than 80 percent of the total number of requests.

In this context, it is of great importance to predict the popularity of broadcast TV programs. First, using the program popularity prediction results, the audience will save much time when trying to find valuable TV programs among massive collections of video resources, which will improve user satisfaction and retention. Second, based on program popularity forecasting data, a company will be able to maximize its advertising effect by choosing the TV programs with highest potential. Finally, with the help of the popularity prediction model, a broadcast TV operator will be able to optimize the configuration of the network in advance by deploying enough transmission and storage resources to distribute popular programs. For example, by applying Auto Regressive and Moving Average (ARMA) models to real

traces extracted from YouTube, accurate predictions can be obtained, as shown by Hassine [3]. Thus, an original solution that combines the predictions of several ARMA models is proposed to determine which contents should be cached. This will greatly help improve the ability of the Content Delivery Networks (CDNs) to promptly react to instant feedback regarding consumer demand in this new video era.

However, accurately predicting the popularity of broadcast TV programs is a challenging task. First, there are many factors influencing TV program popularity that are difficult to measure, such as the quality of the program and the interests of the audience. A hybrid-stream model is proposed to solve these problems for video analysis [4]. Moreover, the relationship between popular events in the real world and in TV programs cannot be easily introduced into the prediction model. Last, there is a massive gap between the popularity evolutionary trends of different programs, which should be considered when designing the prediction model. In this paper, cooperating with a broadcast TV operator, we analyze massive user behavior data and present our improved method for predicting the popularity of broadcast TV programs. The main contributions of our work on program popularity prediction are as follows:

(1) We apply a dynamic time warping (DTW) distance-based K-medoids algorithm to cluster programs with similar popularity into 4 evolutionary trends, which has the ability to capture the inherent heterogeneity of program popularity. This approach is computationally more efficient than previous methods that were used to delineate popularity evolutionary trends, such as K-Spectral Clustering (KSC) models [5]. Computation in these models is always extensive due to model training and the transformation of features in semantic spaces. In contrast, the DTW distance-based K-medoids algorithm is directly driven by raw data. Our method can be implemented without much human intervention and has much lower computational cost.

(2) We build trend-specific prediction models using random forests (RF) regression, which achieve higher overall predictive performances than a single model that was trained on the entire data set. The popularity prediction model was trained separately on different popularity trend data sets and can focus on particular types of programs to reduce the effects of noise. To the best of our knowledge, we are the first to tackle the inherent challenges of predicting broadcast TV program popularity by combining forecasting values from trend-specific models and classification probability.

The proposed method is evaluated using data collected from Jiangsu Cloud-media TV, which is one of the largest broadcast TV platforms in China. The rest of this paper is organized as follows. Section II discusses related work. Section III formally presents our new broadcast TV program popularity prediction model. Our evaluation methodology and main results are discussed in Section IV. Section V discusses our proposed method's drawbacks and our future work. At last, we conclude the paper in Section VI.

II. RELATED WORKS

Online content prediction began with news articles, with methods that predict the news comment volume, popularity of news articles and so on, such as those of Tsagkias et al. [6] and Tatar et al. [7]. Pinto et al. [8] used YouTube video data to predict the future popularity of Web content based on historical information given by early popularity measures.

Due to the rich variety and timeliness of TV content, the semantic understanding of broadcast TV programs is more difficult than that of news, microblogging or other web content. An ideal prediction model for broadcast TV programs achieves not only high prediction accuracy but also good calculation performance, which means that the prediction result is available before audience interest fades. At present, there is little research on program popularity prediction for broadcast TV. The existing popularity prediction methods are for other media formats but can be used as references. Commonly used web content popularity prediction methods include cumulative growth, temporal analysis and evolutionary trends.

Cumulative growth. Researchers have studied the cumulative growth of attention, such as the amount of attention that a single item received from the moment it was published until the prediction moment. Kaltenbrunner et al. [9] proposed that depending on the time of publication, news stories followed a constant growth pattern. A log-linear model was proposed by Szabo and Huberman [10], which outperformed the constant growth models in terms of mean squared error (MSE). A hierarchical framework [11] balances the traffic load and enables a longer lifetime of the whole system. A different approach was proposed by Lee et al. [12]. They used a survival analysis model to detect the threads that would receive more than 100 comments in MySpace with 80% accuracy. A context-aware system architecture is proposed by Wang et al. [13]. Tatar et al. [14] used a simple linear regression based on the early number of comments to predict the final number of comments for news articles. Kim et al. [15] used a linear model on a logarithmic scale to predict popularity ranges for political blog posts. Predicting the popularity of web content, based on the aggregate user behavior, has also been addressed as a classification problem. Jamali and Rangwala [16] trained different classification methods to predict the popularity class of a Digg story with an accuracy of 80%. Wang et al. [17] introduced a big-data-enabled storage planning scheme based on wireless big data computing. Video lifetime was introduced by Su [18] as a coefficient in a popularity prediction model, and a multi-linear model based on historical view count, future bust state and video lifetime was proposed to predict future video popularity. In addition to the regression-based methods, other methods such as reservoir computing [19] and hidden Markov model (HMM) [20] were also utilized to predict online content popularity.

Temporal analysis. Multiple researchers performed temporal analyses of how content popularity evolved over time until the prediction moment. Pinto et al. [8] relied on a multivariate

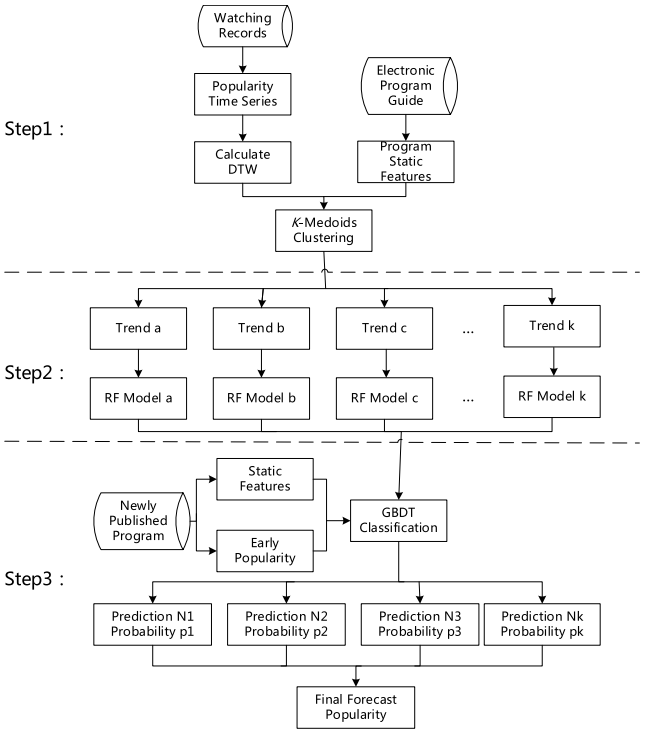

FIGURE 1. Overview of the broadcast TV program popularity prediction method.

linear regression model to predict the popularity of YouTube videos. Maass et al. [21] built a large recurrent neural network that could consider more complex interactions between early and late popularity values. Wang et al. [22] proposed a local data processing architecture on a local server to analyze collected data. Gursun et al. [23] observed that the daily number of views could be modeled through a time-series prediction model using the Autoregressive Moving Average (ARMA).

Evolutionary trends. Other researchers used clustering methods to find web items with similar popularity evolutionary trends. Crane et al. [24] observed that a Poisson process could describe the attention received by the majority of videos and the remaining videos followed three popularity evolutionary trends. An interest-based reduced variable neighborhood search queue architecture was proposed by Wang et al. [25], and Ahmed et al. [26] proposed a model that used a more granular description of the temporal evolution of content popularity, which showed significant improvement over the log-linear model. A log-linear regression model was proposed by Szabo et al. [10] to predict the long-term popularity of YouTube videos based on the early popularity of online content.

Most of the previous studies focus on building a general model to predict the popularity of certain content in a specific medium but neglect the massive gap that develops as content popularity evolution progresses. As a result, those methods are generally ineffective for program popularity prediction for broadcast TV, especially when predicting programs with early peaks and later bursts of popularity. To the best of our knowledge, no other work has studied the predictive power of features extracted from an electronic program guide. In summary, we are first to detect different popularity evolutionary trends of broadcast TV automatically and develop an integrated prediction model by combining forecasting values from trend-specific models and classification probability.

III. METHODOLOGY

A. PROBLEM STATEMENT

The program popularity prediction problem can be defined as follows. Let $c \in C$ be an individual program from a set of programs C that are observed during a period T. We use $t \in T$ to describe the age of a program (i.e., the time since it was first published) and mark two important moments: the indication time t_i, which is the time at which we perform the prediction, and the reference time t_r, which is the moment of time for which we want to predict program popularity. Let $N_c(t_i)$ be the popularity of c from the time a program was published until t_i and $N_c(t_r)$ be the value that we want to predict, i.e., the popularity at a later time $N_c(t_r)$. We define $\hat{N}_c(t_i, t_r)$ as the prediction outcome: the predicted popularity of program c at time t_r using the information available until t_i. Thus, the better the prediction, the closer $\hat{N}_c(t_i, t_r)$ is to $N_c(t_r)$.

B. METHOD OVERVIEW

Our method follows 3 steps, as shown in Fig. 1. The first step is to detect popularity evolutionary trends. We calculate the DTW distances between historical-record time series and try to cluster the popularity evolutionary trends into optimal trends. Eleven static features extracted from EPG are introduced to strengthen the results of clustering. A few trials are performed to determine an appropriate value for the number of popularity trends (k) in our case study.

For TV program popularity, there exist different types of propagation trends. Different propagation trends have different high-level features. If we could separate them and train the model using data from a certain type of propagation trend, we could obtain better results for each type. Therefore, our first step is to identify the propagation trends and separate them into different types (clusters). For TV propagation trends, typical time series clustering is performed, for which we can use DTW-based K-medoids. DTW is one of the best distance-measuring tools; later, we will give a more detailed introduction to DTW-based K-medoids.

The second step is to build trend-specific prediction models using RF regression. We split the view records into 4 groups according to the above trends and feed them to the RF regression model, together with static features. According to several empirical studies, clustering program popularity into more than 4 trends will not improve the accuracy of the prediction model significantly. Therefore, we decide to cluster the popularity evolutionary trends of broadcast TV programs into 4 prediction models.

The third step is to use gradient boosting decision tree (GBDT) to classify the popularity time series of newly

published programs into the trends and obtain the final prediction results based on the prediction values of the 4 models and the classification probability.

C. POPULARITY TREND DETECTION

In this section, we describe the details of our method for K-medoids [27] clustering of program popularity time series with DTW [28] distance. In time-series data analysis, the DTW distance is an accurate measure of the similarity between two temporal signals, which may have different speeds. A non-linear mapping of one signal to another is obtained by minimizing the distance between the two signals. This approach is widely used in detecting similarities between temporal sequences of audio, image, or video data, or any data that can be transformed into a linear sequence. Decades ago, DTW was introduced in the academic community to solve for different speaking speeds in automatic speech recognition problems.

To find an optimal match between two time-series sequences, a ''warped'' path minimizes the warping cost to determine a measure of their similarity that is independent of certain non-linear variations in the time dimension. An optimal alignment and distance between two sequences $P = (p_1, p_2 \ldots p_n)$ and $Q = (q_1, q_2 \ldots q_m)$ can be determined as follows:

$$DTW(P, Q) = \sqrt{dist(p_n, q_m)}, \tag{1}$$

$$dist(p_i, q_j) = (p_i - q_j)^2 + \min \begin{cases} dist(p_{i-1}, q_j) \\ dist(p_i, q_{j-1}) \\ dist(p_{i-1}, q_{j-1}). \end{cases} \tag{2}$$

The DTW distance is calculated through dynamic programming to determine the minimum cumulative distance of each element in an $n \times m$ matrix. In addition, the warping path between two sequences can be found by tracing back from the last cell. In this work, the DTW distance is used to measure the similarity between each program's popularity time series data and cluster centers to give more accurate results.

The K-medoids algorithm is similar to the well-known K-means algorithm for performing clustering analysis. However, these two methods differ in how they update the center location for a certain cluster. In the K-means approach, the center of a cluster is virtual because it represents the mean position of the members that are currently within the cluster. However, the K-medoids method treats the center as the median of the cluster; thus, the center coincides with one of the members. Owing to this difference, the K-medoids algorithm is more robust to outliers in the dataset.

The K-medoids algorithm based on DTW algorithm is described briefly as Algorithm 1. First, we arbitrarily choose k programs in D as the initial medoids and assign each remaining program to the cluster with the nearest medoids. Then, we randomly select a non-medoid program to compute the new DTW distance of the trends. If the new DTW distance is less than the previous one after swapping, we swap to form a new set of k medoids. The above steps are repeated until there is no change of programs in each trend.

Algorithm 1 K-Medoids Based on the DTW Algorithm (KMDTW(D, C))

1. D: the data set containing program popularity time series
2. C: the number of trends
3. K: the set of trend centers
4. M: the set of popularity sequences in each trend
5. initialize C as trend centers of K
6. **do**
7. **for** i = 1:size(D)
8. **for** k = 1:K
9. $Dist_{Di,Ck}$ = DTW(D_i, C_k)
10. **end for**
11. **if**($Dist_{Di,Ck}$ is min)
12. assign D_i into M_k
13. **end if**
14. **end for**
15. **while**(the cluster membership changes)
16. **return** K, M

D. TREND-SPECIFIC PREDICTION MODELS

In this section, we describe the details of training trend-specific prediction models using Random Forests (RF) regression [29]. Random Forests is an improved algorithm based on bagged decision trees. Bootstrap Aggregation (bagging) is a simple and powerful ensemble method, which combines the predictions from other machine learning algorithms to produce a more accurate prediction than single machine learning algorithms.

Bootstrap Aggregation can generalize to reduce the variance for algorithms with high variance, such as decision trees, e.g., classification and regression trees (CART). Decision trees are sensitive to the specific data on which they are trained. If the training data are changed, the resulting decision tree will become quite different, which will influence the prediction.

When bagging with decision trees, we are less concerned about individual trees overfitting the training data. The individual decision trees are grown deep and are not pruned. These trees will have both high variance and low bias. These are important characteristics of sub-models when combining predictions using bagging.

Random Forests improves the bagged decision tree algorithm. One problem with decision trees such as CART is that they use a greedy algorithm that minimizes error, which will cause the decision trees to have high structural similarity and, in turn, high correlation in their predictions. This makes it less advantageous to combine predictions from multiple models in ensemble methods.

Random Forests changes the way that the sub-trees are learned in the algorithm such that the resulting predictions from all subtrees are less correlated.

In CART, the optimal split-point is selected, while the random forests algorithm is limited in its search to a random sample of features.

Tin Kam Ho utilized the random subspace method [30] to construct the first random decision forests in [31], which implement the ''stochastic discrimination'' approach to classification, which was proposed by Eugene Kleinberg [32]–[34]. Leo Breiman [35] and Adele Cutler [36] extended this work and named it ''Random Forests.'' The extension combines Breiman's ''bagging'' idea with random selection of features. The idea was first proposed by H. Tin [30] and later independently by Amit and Geman [37] to construct a collection of decision trees with controlled variance. Wang et al. [38] proposed a novel electricity price forecasting model by merging the Random Forests and Relief-F algorithms.

RF is an extension of bagging and a competitor to boosting. It uses either categorical (i.e., classification) or continuous (i.e., regression) response variables, and either categorical or continuous predictor variables. In RF modeling, the following training parameters have to be specified: (i) the number of trees to grow in the forest (*n tree*), the number of randomly selected predictor variables at each node (*m try*), and the minimal number of observations at the terminal nodes of the trees (*node size*). In our study, those were set to 1000, 12, and 5, respectively. The default value of *n tree* was 500, but it has been observed that more stable results for estimating variable importance are achieved with a higher value. The training data that were left out of the bootstrap (i.e., Out-Of-Bag, OOB) samples were used to estimate the prediction error and variable importance. In the error estimation, the OOB samples were predicted by the respective trees, and by aggregating the predictions, the mean square error of OOB was calculated by (3).

$$MSE_{OOB} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_{i_{OOB}})^2, \tag{3}$$

where $\hat{y}_{i_{OOB}}$ is the OOB prediction for observation y_i. To calculate variable importance, the values of a specific predictor variable were randomly permuted in the OOB data of a tree, while the values of other predictors remained fixed. The modified OOB data were predicted, and the differences between the MSEs obtained from the permutated and original OOB data were used as a measure of variable importance. In our dataset, we will use the attributes of the broadcast TV programs, the first-7-day viewing records and some derivation as the predictor, and the 30^{th}-day records as y. For each cluster, we train a unique model to fit the dataset.

E. CLASSIFICATION OF NEWLY PUBLISHED PROGRAMS' POPULARITY

Gradient Boosting Decision Trees (GBDT) [39] is a powerful method for building predictive models, which is generalized from AdaBoost. The key question is whether a weak learner can be modified to become a strong one, which is called boosting; this is articulated by Michael Kearns in [40]. A weak learner is a learner whose performance is at least slightly better than random chance. Hypothesis boosting is used to filter observations, to leave observations on which the weak learner performed well and focus on developing new weak learners to handle observations on which the previous weak learner performed poorly [41]. AdaBoost is the first realization of the boosting idea for binary classification problems [42]. Adaboost weights all the observations by putting more weight on difficult-to-classify instances and less on those that have already been well classified. New weak learners are added sequentially, which continues to improve the training for difficult patterns. Wang et al. [43] proposed an improved mechanism called the multidimensional learning factor to lower the learning error and increase the convergence rate.

Adaboost was recast in a statistical framework by Breiman [44] to produce the ARCing algorithm. ARCing is an acronym for Adaptive Reweighting and Combining. This framework was further improved by Friedman [45], who proposed Gradient Boosting Machines, which was later called Gradient Tree Boosting. Wang [46] et al. incorporated the big data approximate analytics algorithm into the hyperplane fitting for optimization and analysis. The statistical framework cast boosting as a numerical optimization problem in which the objective is to minimize the loss of the model by adding weak learners using a gradient-descent-like procedure. Algorithms in this class were described as stage-wise additive models because one new weak learner is added at a time and existing weak learners in the model are frozen and left unchanged.

Gradient Boosting Decision Trees is an additive regression model that consists of an ensemble of decision trees. A single decision tree has the problem of over-fitting; however, the GBDT algorithm can overcome this by combining hundreds of weak decision trees, each consisting of a few leaf nodes. GBDT has a few advantages, including the ability to find non-linear transformations, the ability to handle skewed variables without requiring transformations, computational robustness and high scalability.

In this paper, we build decision trees to classify newly published programs into 4 popularity evolutionary trends. Eleven program attributes are extracted from an electronic program guide, which are described in Table 1.

These attributes and first-7-day viewing records are the predictors and the 4 trends are the targets of classification. With the help of the GBDT and RF prediction models, for each program, we obtain its probability P_{cj} of belonging to each trend k and the temporary popularity value $\hat{N}_{ck}(t_i, t_r)$ predicted by the corresponding model. To maximize the information gain, we use Eq. 4 to calculate the final predicted popularity of program c at time t_r using the information available until t_i.

$$\hat{N}_c(t_i, t_r) = \sum_{k=1}^{4} P_{ck}\hat{N}_{ck}(t_i, t_r) \tag{4}$$

TABLE 1. Descriptions of attributes extracted from EPG.

Attribute	Description
Time	The time when the program is first published
Name	The formal name or nickname of the program
Duration	Time length of the program
Actors	The names of the main actors and actresses
Director	The names of the directors of the program
Language	The language of dialogue and subtitles
Area	The country where the program was filmed
Category	The type of the program, such as news or cartoon
Rating	The scores given by douban.com
Publisher	The program production company name
Summary	Brief description of the program

IV. EXPERIMENTS

A. DATASETS

The experimental data originate from Jiangsu Cloud-media TV, which is one of the largest broadcast TV platforms in China. The data set is summarized in Table 2. It contains broadcast TV requests over 213 days between January 1st and July 31st, 2016. During this period, 423254 programs were requested. The data set includes more than 2 billion requests from more than 1.3 million clients.

TABLE 2. Summary of the data set.

	Requests	Programs	Clients
Daily max	2300747	20447	407133
Daily min	1066706	17552	225290
Daily median	1678506	19098	318052
Total	201420717	423254	1309381

By cleaning the RTSP (Real Time Streaming Protocol) packets from the video server and analyzing the EPG information, we obtained popularity time series and 11 static features for 110 thousand programs. The static features include the directors' names, writers' names and actors/actresses' names, country, language, categories, duration, premiere channel, premiere time and content description. These experimental results were computed using 10-fold cross-validation. We split the dataset into 10 folds, 9 of which were used as the training set and one as the test set, and rotated the folds such that each fold is used for testing once.

B. EXPERIMENTAL ENVIRONMENT

The main experimental environment is described below, including the hardware configuration and software AI framework, as shown in Table 3.

The main hardware configuration gives helpful information for reaching a better understanding and comparison in terms of time consumption.

C. PERFORMANCE METRICS

A comprehensive and reasonable error analysis can effectively evaluate the performance of the prediction model. Commonly used metrics are divided into absolute metrics,

TABLE 3. Computer configuration.

Attribute	Description
CPU	Intel Xeon E5-2680 v3 @ 2.50 GHz (x2)
Memory	64 G (DDR4 2133 MHz)
Hard disk	512 G
AI Framework	Scikit-learn, Keras

such as MSE, Root of MSE (RMSE) Eq. (5) and Mean Absolute Error (MAE), and the relative ones, such as Mean Relative Error (MRE) and Mean Relative Squared Error (MRSE). When using the absolute metrics, researchers must have a clear understanding of the numerical range of the prediction values. Relative metrics are useful for comparing the efficiencies of prediction algorithm across studies, except when the actual value is zero. The quality of a numerical prediction can also be reported using the correlation coefficient or the coefficient of determination (R2), which is shown in Eq. (6). To compare the performance of our method to those of existing methods and avoid the zero-inflation problem, we choose RMSE and R2 as performance metrics.

$$\mathrm{RMSE} = \sqrt{\frac{1}{|C|}\sum_{c\in C}\left(\hat{N}_c\left(t_i, t\right) - N_c\left(t_r\right)\right)^2} \tag{5}$$

$$\mathrm{R}^2 = 1 - \frac{\sum_{c\in C}\left(N_c\left(t_r\right) - \hat{N}_c\left(t_i, t\right)\right)^2}{\sum_{c\in C}\left(N_c\left(t_r\right) - \bar{N}_c\left(t_r\right)\right)^2} \tag{6}$$

D. PREDICTION RESULTS

We use scikit-learn [47], which is a Python machine learning package, to implement the required clustering and regression algorithms in this study. A few trials are performed to determine an appropriate value for the number of popularity trends (K) in our case study. This is achieved by running the DTW-distance-based K-Medoids method with different values of K (ranging from 2 to 10).

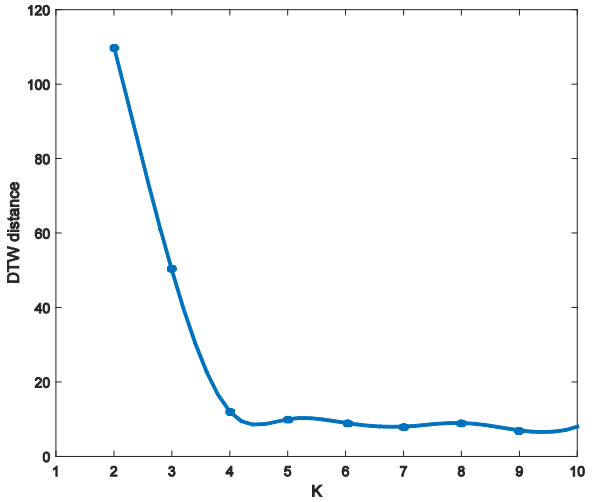

FIGURE 2. The DTW distances between trends with different values of K.

Fig. 2 shows that as the value of K increases, the DTW distance value between different trends decreases significantly,

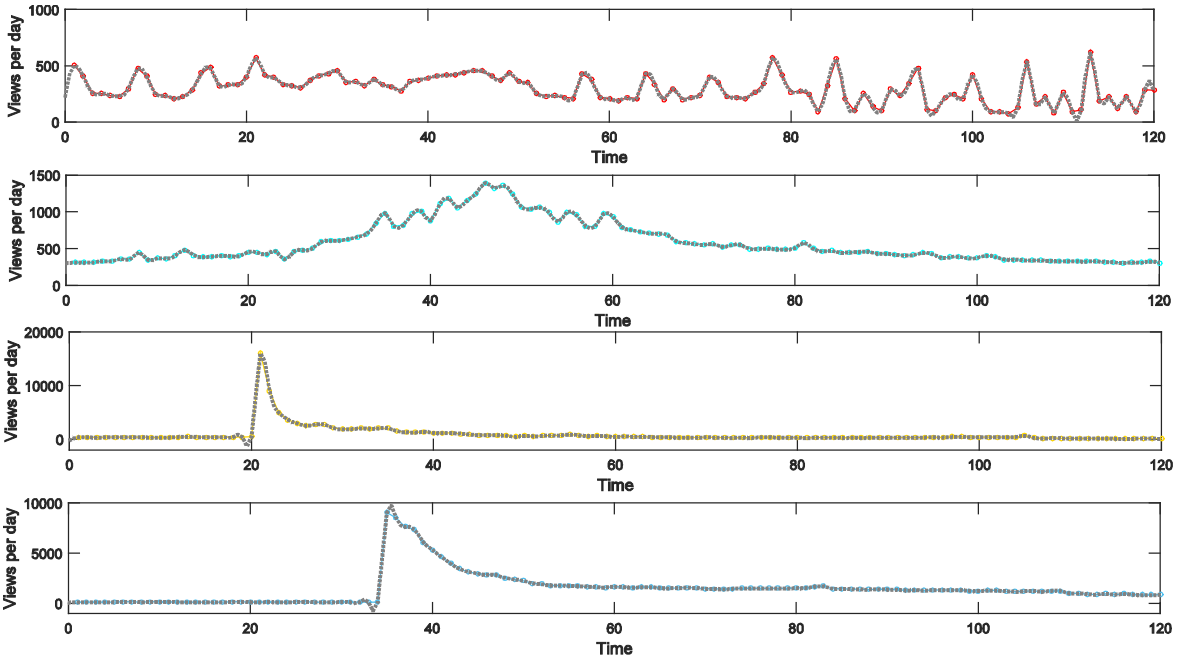

FIGURE 3. Prevalent popularity trends of broadcast TV programs.

TABLE 4. Prediction RMSE for S-H, ML, MRBF and our model for programs with different popularity trends.

Popularity Trend	Numbers of Programs	S-H	ML	MRBF	Our Model
Trend a	1554	0.4588±0.0283	0.1402±0.0274	0.1268±0.0319	0.1039±0.0263
Trend b	80528	0.2086±0.0040	0.1788±0.0041	0.1713±0.0040	0.1576±0.0040
Trend c	6636	0.1921±0.0124	0.1707±0.0283	0.1490±0.0199	0.1302±0.0172
Trend d	24143	0.3351±0.0099	0.2929±0.0120	0.2641±0.0111	0.2223±0.0104
Overall	112861	0.2382±0.0038	0.2022±0.0043	0.1892±0.0032	0.1677±0.0028

reaching 8.3 at $K = 4$. However, the DTW distance value decreases much more slowly when K is within [5], [10]. This implies that clustering program popularity into more than 4 trends will not improve the accuracy of the prediction model; instead, it will degrade the performance. Therefore, we decide to cluster the popularity evolutionary trends of broadcast TV program into 4 categories.

Fig. 3 shows the popularity trends that were discovered in our dataset. Each graph shows the number of views as a function of time. We note that the 4 categories of trends produced by K-Medoids clustering using the DTW distance algorithm are consistent with the trends identified in other research [5], [10]. Although the 4 trends cannot match the popularity evolution progress of all programs, the most prevalent trends are detected, which can greatly improve the accuracy of our prediction models.

We measured the RMSE and R^2 to evaluate the prediction performance, and compared our prediction method with three other existing methods: the Szabo-Huberman (S-H) [10], Multivariate Linear [15] and MRBF [8] models. Table 4 shows the MRSE results produced by all 4 models, with $t_i = 7$ and $t_r = 30$. Result for other values of t_i and t_r are quite similar. The overall MRSE reduction achieved by our model over the others, across all trends, is 20% on all datasets. The grains are especially large for trend c, whose popularity reaches a high peak and then declines sharply.

Fig. 4 shows the coefficient of determination (R^2) values produced with different indication times by all 4 models. As the history data accumulates, the R^2 values approach 100%. The sooner reliable prediction results are obtained, the more profitable broadcast TV service is. To obtain 95% R^2, the other 3 models need to collect the popularity data for at least 12 days, while our model needs only 9 days of data, which means our model can give a reliable result much earlier.

V. DISCUSSION

Compared with three conventional methods, our proposed method obtains better results. However, there is still work to be done to improve our ideas. First, the GBDT method can be substituted by Extreme Gradient Boosting (XGBoost) [48], which has great advantages in parallel processing to accelerate the computation speed and can utilize column sampling and normalization to reduce the overfitting problem to further optimize the whole algorithm.

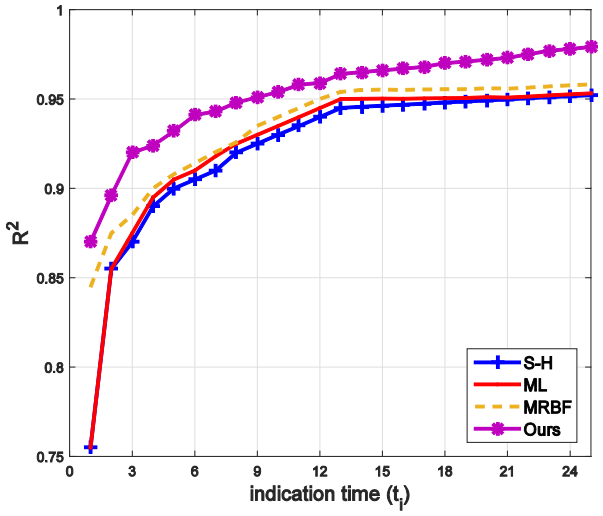

FIGURE 4. Comparison of coefficient of determination as a function of indication time for different methods.

Second, the current model requires historical-record data as input to generate our prediction; thus, determining how to solve a cold-start problem is still a big challenge for our operator.

Third, there are valuable factors that are not considered in our model, such as the audience, the rating for each program, the reputations of the director and actors, and public sentiment analysis data. Some text analysis methods will be utilized to improve our model's accuracy in our future work.

At last, currently, the training data is mainly focused on half a year, from 1.1.2016 to 7.1.2016. However, program broadcasting has some periodic features, which will result in significantly different trend data and challenge our model's robustness. For example, the winter and summer holidays will have significantly different features. Thus, our future work will expand the window of data on which our model is trained to further improve the model's accuracy.

VI. CONCLUSIONS

In this paper, we have analyzed massive user behavior data and presented our improved method to predict the popularity of broadcast TV programs. To the best of our knowledge, this is the first work to tackle the problem of predicting program popularity in the broadcast TV platform. We applied a dynamic time warping (DTW)-distance-based K-Medoids algorithm to group programs with similar popularity into 4 evolutionary trends, which has the ability to capture the inherent heterogeneity of program popularity. Moreover, we built trend-specific prediction models using random forests regression, which have better overall predictive performance than a single model trained on the entire data set.

We performed an extensive experimental evaluation of our method, in which we compared it with 3 representative methods. Our method outperforms the others, with a gain in accuracy of at least 20%, and can give reliable prediction results much faster.

In the future, we plan to apply our method to the infrastructure of the broadcast TV platform and try to develop a cache replacement strategy that can proactively adapt to the evolution of program popularity.

REFERENCES

[1] "Cisco visual networking index: Forecast and methodology, 2016–2021," Cisco, San Jose, CA, USA, Tech. Rep., 2016.
[2] Tencent Video. (2016). *Top Video Lists*. [Online]. Available: http://v.qq.com/rank/detail/2_-1_-1_-1_1_1.html
[3] N. B. Hassine, R. Milocco, and P. Minet, "ARMA based popularity prediction for caching in content delivery networks," in *Proc. Wireless Days*, Mar. 2017, pp. 113–120.
[4] K. Wang, J. Mi, C. Xu, Q. Zhu, L. Shu, and D.-J. Deng, "Real-time load reduction in multimedia big data for mobile Internet," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 12, no. 5s, p. 76, 2016.
[5] F. Figueiredo, J. M. Almeida, M. A. Gonçalves, and F. Benevenuto, "TrendLearner: Early prediction of popularity trends of user generated content," *Inf. Sci.*, vols. 349–350, pp. 172–187, Jul. 2016.
[6] M. Tsagkias, W. Weerkamp, and M. de Rijke, "News comments: Exploring, modeling, and online prediction," in *Advances in Information Retrieval*. Cham, Switzerland: Springer, 2010, pp. 191–203.
[7] A. Tatar, P. Antoniadis, M. D. de Amorim, and S. Fdida, "From popularity prediction to ranking online news," *Social Netw. Anal. Mining*, vol. 4, pp. 174–183, Dec. 2014.
[8] H. Pinto, J. M. Almeida, and M. A. Gonçalves, "Using early view patterns to predict the popularity of youtube videos," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, 2013, pp. 365–374.
[9] A. Kaltenbrunner, V. Gómez, and V. López, "Description and prediction of slashdot activity," in *Proc. Latin Amer. Web Conf. (LA-WEB)*, 2007, pp. 57–66.
[10] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Commun. ACM*, vol. 53, no. 8, pp. 80–88, 2010.
[11] K. Wang, Y. Wang, Y. Sun, S. Guo, and J. Wu, "Green industrial Internet of Things architecture: An energy-efficient perspective," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 48–54, Dec. 2016.
[12] J. G. Lee, S. Moon, and K. Salamatian, "Modeling and predicting the popularity of online contents with Cox proportional hazard regression model," *Neurocomputing*, vol. 76, no. 1, pp. 134–145, 2012.
[13] K. Wang, H. Lu, L. Shu, and J. J. P. C. Rodrigues, "A context-aware system architecture for leak point detection in the large-scale petrochemical industry," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 62–69, Jun. 2014.
[14] A. Tatar *et al.*, "Predicting the popularity of online articles based on user comments," presented at the Proc. Int. Conf. Web Intell., Mining Semantics, Sogndal, Norway, 2011.
[15] S.-D. Kim, S.-H. Kim, and H.-G. Cho, "Predicting the virtual temperature of Web-blog articles as a measurement tool for online popularity," in *Proc. IEEE 11th Int. Conf. Comput. Inf. Technol.*, Sep. 2011, pp. 449–454.
[16] S. Jamali and H. Rangwala, "Digging digg: Comment mining, popularity prediction, and social network analysis," in *Proc. Int. Conf. Web Inf. Syst. Mining*, 2009, pp. 32–38.
[17] K. Wang *et al.*, "Wireless big data computing in smart grid," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 58–64, Apr. 2017.
[18] B. Su, Y. Wang, and Y. Liu, "A new popularity prediction model based on lifetime forecast of online videos," in *Proc. IEEE Int. Conf. Netw. Infrastruct. Digit. Content (IC-NIDC)*, Sep. 2016, pp. 376–380.
[19] T. Wu, M. Timmers, D. D. Vleeschauwer, and W. V. Leekwijck, "On the use of reservoir computing in popularity prediction," in *Proc. 2nd Int. Conf. Evol. Internet*, 2010, pp. 19–24.
[20] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Ann. Math. Statist.*, vol. 37, no. 6, pp. 1554–1563, 1966.
[21] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, 2002.
[22] K. Wang, Y. Shao, L. Shu, G. Han, and C. Zhu, "LDPA: A local data processing architecture in ambient assisted living communications," *IEEE Commun. Mag.*, vol. 53, no. 1, pp. 56–63, Jan. 2015.
[23] G. Gürsun, M. Crovella, and I. Matta, "Describing and forecasting video access patterns," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 16–20.

[24] R. Crane and D. Sornette, "Robust dynamic classes revealed by measuring the response function of a social system," *Proc. Nat. Acad. Sci. USA*, vol. 105, no. 41, pp. 15649–15653, 2008.
[25] K. Wang, Y. Shao, L. Shu, C. Zhu, and Y. Zhang, "Mobile big data fault-tolerant processing for ehealth networks," *IEEE Netw.*, vol. 30, no. 1, pp. 36–42, Jan. 2016.
[26] M. Ahmed, S. Spagna, F. Huici, and S. Niccolini, "A peek into the future: Predicting the evolution of popularity in user generated content," presented at the Proc. 6th ACM Int. Conf. Web Search Data Mining, Rome, Italy, 2013.
[27] W. Sun, L. Xiang, X. Liu, and D. Zhao, "An improved K-medoids clustering algorithm based on a grid cell graph realized by the P system," in *Proc. 2nd Int. Conf. Human Centered Comput., (HCC)*, Colombo, Sri Lanka, Jan. 2016, pp. 365–374.
[28] A. Mueen and E. Keogh, "Extracting optimal performance from dynamic time warping," presented at the Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Francisco, CA, USA, 2016.
[29] M. A. Hannan, J. A. Ali, A. Mohamed, and M. N. Uddin, "A random forest regression based space vector PWM inverter controller for the induction motor drive," *IEEE Trans. Ind. Electron.*, vol. 64, no. 4, pp. 2689–2699, Apr. 2017.
[30] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1. 1995, pp. 278–282.
[31] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.
[32] E. M. Kleinberg, "An overtraining-resistant stochastic modeling method for pattern recognition," *Ann. Statist.*, vol. 24, no. 6, pp. 2319–2349, 1996.
[33] E. M. Kleinberg, "On the algorithmic implementation of stochastic discrimination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 5, pp. 473–490, May 2000.
[34] E. M. Kleinberg, "Stochastic discrimination," *Ann. Math. Artif. Intell.*, vol. 1, pp. 207–239, Sep. 1990.
[35] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[36] A. Liaw, "Documentation for R package randomForest," Feb. 2013. [Online]. Available: https://www.rdocumentation.org/packages/randomForest/versions/4.6-12
[37] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Comput.*, vol. 9, no. 7, pp. 1545–1588, 1997.
[38] K. Wang, C. Xu, Y. Zhang, S. Guo, and A. Zomaya, "Robust big data analytics for electricity price forecasting in the smart grid," *IEEE Trans. Big Data*, to be published.
[39] Y. Wang, D. Feng, D. Li, X. Chen, Y. Zhao, and X. Niu, "A mobile recommendation system based on logistic regression and gradient boosting decision trees," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2016, pp. 1896–1902.
[40] M. Kearns, "Thoughts on hypothesis boosting," *Mach. Learn.*, vol. 45, p. 105, Oct. 1988.
[41] C. Arney, "Probably approximately correct: Nature's algorithms for learning and prospering in a complex world," *Math. Comput. Edu.*, vol. 48, pp. 126–136, Jun.2014.
[42] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Proc. Eur. Conf. Comput. Learn. Theory*, 1995, pp. 23–37.
[43] K. Wang, L. Zhuo, Y. Shao, D. Yue, and K. F. Tsang, "Toward distributed data processing on intelligent leak-points prediction in petrochemical industries," *IEEE Trans. Ind. Informat.*, vol. 12, no. 6, pp. 2091–2102, Dec. 2016.
[44] L. Breiman, "Prediction games and arcing algorithms," *Neural Comput.*, vol. 11, pp. 1493–1517, Dec. 1999.
[45] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.
[46] K. Wang, H. Li, Y. Feng, and G. Tian, "Big data analytics for system stability evaluation strategy in the energy Internet," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1969–1978, Aug. 2017.
[47] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller, "Scikit-learn: Machine learning without learning the machinery," *GetMobile: Mobile Comput. Commun.*, vol. 19, pp. 29–33, Jan. 2015.
[48] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," presented at the Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Francisco, CA, USA, 2016.

CHENGANG ZHU is currently pursuing the Ph.D. degree with the Computer Science and Engineering School, Southeast University. His research interests include networking measurement behavior analysis and future networks.

GUANG CHENG (SM'10) received the D.Eng. degree. He is currently a Professor and a Doctoral Tutor with the Computer Science and Engineering School, Southeast University. He is also the Director of the Key Laboratory of Computer Network and Information Integration, Southeast University, Ministry of Education, and the Secretary of the School of Computer Science and Engineering, School of Software, Southeast University. He is a Senior Member of the Chinese Computer Federation. He has served as the Director for the Computer Network Committee for the Micro Computer Application Association in Jiangsu province. He is a Standing Committee Member of CCF TCI and the Nanjing Branch of the China Computer Federation. He has been the Vice President of the Jiangsu Software Talent Training Association and the Computer Ethics and Occupation Training Committee.

KUN WANG (M'13–SM'17) received the B.Eng. and Ph.D. degrees from the School of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004 and 2009, respectively. From 2013 to 2015, he was a Post-Doctoral Fellow with the Electrical Engineering Department, University of California, Los Angeles, CA, USA. In 2016, he was a Research Fellow with the School of Computer Science and Engineering, University of Aizu, Aizu-Wakamatsu City, Japan. He is currently a Research Fellow with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, and also a Full Professor with the School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing. He has published over 100 papers in referred international conferences and journals. His current research interests are mainly in the area of big data, wireless communications and networking, smart grid, energy Internet, and information security technologies. He is a member of ACM. He has received the Best Paper Award at the IEEE GLOBECOM16. He was the symposium Chair/Co-Chair of the IEEE IECON16, the IEEE EEEIC16, the IEEE WCSP16, the IEEE CNCC17. He serves as an Associate Editor of the IEEE Access, *Journal of Network and Computer Applications*, and *EAI Transactions on Industrial Networks and Intelligent Systems*, and as Editor of the *Journal of Internet Technology*.