Multimedia Content Popularity: Learning and Recommending a Prediction Method

Nhan Nguyen-Thanh, Dana Marinca, Kinda Khawam, Steven Martin, Lila Boukhatem

Multimedia Content Popularity: Learning and Recommending a Prediction Method

Nhan Nguyen-Thanh[1,2], Dana Marinca[2], Kinda Khawam[2,1], Steven Martin[1], Lila Boukhatem[1]
[1] LRI, CNRS - Univ. of Paris Saclay, Univ. of Paris-Sud, France
[2] David Lab., Univ. of Paris Saclay, UVSQ, France
Email: nhan.nguyen@lri.fr, dana.marinca@uvsq.fr, kinda.khawan@uvsq.fr
steven.martin@lri.fr, lila.boukhatem@lri.fr

***Abstract*—In 5G networks, Mobile Edge Computing (MEC) has been proposed to enable computation and storage capabilities at the edge of Radio Access Networks. Proactive content caching in MEC is crucial to guarantee users' Quality of Experience thanks to the reduction of traffic latency. Predicting content popularity plays a key role in the effectiveness of proactive caching. In this paper, we propose a generic and flexible recommendation framework which allows recommending suitable learning and prediction algorithms among available ones, in order to predict content popularity. The investigated algorithms are categorized into two main classes: tree-based regressors and recurrent neural networks. Through the study case of YouTube video solicitation profiles, our proposed method, called Imputation-Boosted Collaborative-Filtering based Recommending Prediction Method (IBCF-RPM) shows its effectiveness in the prediction of content popularity for various popularity profiles. By running only 30% of the prediction algorithms, randomly chosen, on a given content profile, the proposed recommending method is able to estimate the accuracy of the other predictors and recommend a well-suited predictor for content popularity.**

***Index Terms*—content popularity prediction, multimedia caching, YouTube, collaborative filtering, recommendation, machine learning, RNN, GRU, LSTM, tree-based regressors.**

I. Introduction

The extraordinary increase in data traffic, mainly derived from mobile multimedia services [1], social networks and over-the-top applications such as YouTube, has imposed a huge challenge to 5G networks for dealing with the requirements of more backhaul resources [2]. Mobile Edge Computing (MEC), currently standardized by ETSI [3], is part of the effort toward a general solution for 5G-based network architecture. MEC allows computation and storage capabilities at the edge of Radio Access Network (RAN) [4], based on which new tasks such as mobile big data analytics and context-aware services can be implemented for performance optimization. Applications and services can be deployed and popular content items can be cached near end-users. This can help improve backhaul offloading and users' Quality of Experience (QoE) thanks to the reduction of traffic latency. Video on Demand is the heaviest data-driven among various multimedia services causing up to 80% of backhaul traffic by the year 2020 [5]. Thus, predicting the popularity of multimedia contents for their proactive caching is a crucial issue. Several previous solutions have been proposed for content popularity prediction [6], [7], [8]. However, most solutions are based on a prediction approach which is mainly effective for a small dataset. For the very large amount of contents, an effective estimation requires simultaneously the support of a big data-enabled network architecture and an adaptive prediction framework. Recent achievements in the use of machine learning for big data analysis [9] have encouraged us to aim at a learning-based approach for contents popularity predictions. YouTube [10], one of the largest video sharing platform, will be the study case in this paper. The number of the daily solicitations of a content represents a good indication of its popularity profile, so it is the mainly considered parameter to predict its popularity. The most solicited items will be cached to the nearby servers/base stations. Because daily requests evolution of a content is represented as time-series data, hereafter, we focus on time-series prediction. Predicting time-series has used various approaches such as exponential smoothing [7], [11] Auto Regressive Integrated Moving Average (ARIMA) [8], [12], Neural Network (NN) and Recurrent Neural Network (RNN) [13], [14], tree-based regressors [6], etc. The prediction methods may perform differently regarding the same time-series. Hence, finding the mechanism that optimally activates the most efficient prediction algorithm for a given context is a challenge. Combinations of several forecasters have been proposed in order to keep individual forecasters' advantages in a universal predictor [15]. However, the combinations are mainly based on weighted sum methods, and adjusting weights is cumbersome. In this paper, we propose to adopt a recommendation technique [16], for adaptively and dynamically selecting suitable prediction methods. This flexible technique can be applied to any time series, but in our context, it is applied for the prediction of YouTube contents' popularity.

The remainder of the paper is organized as follows. In Sect. II, we define the problem of multimedia content popularity prediction, YouTube dataset, and the methodology. In Sect. III-A, we focus on a selected set of learning-prediction methods. In Sect. III-B, we present the recommendation systems and the collaborative filtering (CF) technique that we use to propose a new recommendation method IBCF-RPM (Sect. IV) Sect. V, presents the simulation context and performance results and Sect. VI, concludes the study and mentions its perspectives.

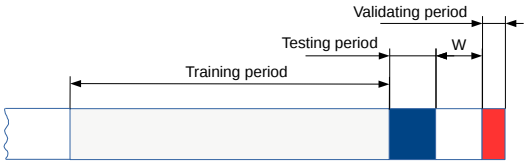

Fig. 1: Content profile splitting

II. Problem statement

We address the problem of selecting the most appropriate method from a set of well-known methods to predict the popularity of multimedia contents. A large set of prediction methods going along with several configurable parameters lead to a huge number of predictors. Applying all available predictors to a very large catalog of contents (e.g. YouTube) in order to select the best one is intractable. We propose hereafter a recommendation framework able to recommend a suitable predictor for a given content profile based on the computed accuracy of a limited number of predictors and the similarities between different profiles. The predicted popularity can be used to supply a proactive caching of the most popular contents close to requesting users, but this issue is outside the scope of this paper.

A multimedia content popularity profile represents the number of *daily requests* for this content and the associated characteristics (e.g., number of *likes*). The daily requests number impacts explicitly the decision of caching it to the nearby servers in 5G MEC network. Because the number of daily requests is basically a time-series data, we focus on time series prediction.

With the complexity of sequences' dependencies and without any restrictions on the distribution, on the frequency or on the amplitude of a time-series data, the sequential prediction is difficult but is an important methodology applied to a wide range of problems. Therefore, the problem considered in this paper can be extended to other time-series prediction contexts.

A. YouTube popularity profile dataset

In this study, the dataset is constituted by real traces crawled from the YouTube site. A trace associated to a multimedia content includes the content uploading date, the subscribers' number, the shares' number and its daily solicitation until the crawling day. The list of YouTube videos is extracted from the YouTube-8M dataset [17] composed of approximately 8 million videos. The crawled dataset includes contents having at least 10000 total solicitations between Nov. 2017 and Dec. 2017. For this study, we randomly selected 200 contents having over 1.5 million access and at least 500 solicitations/day.

Each content profile in the selected set is divided into three periods as shown in Fig.1. The training period, constituted by several consecutive days, represents the training data. The testing period of consecutive days follows. The validation period composed by several consecutive days is separated from the testing period by a window W of variable length. The separating window is used to ensure the objectivity of the validation. This operating mode is usual in any time-series prediction problems.

B. Methodology

Various individual methods have been adopted for time-series predictions such as Exponential Smoothing [7], [11], ARIMA [8], Prophet [18], tree-based regressors [19], [20], [21], neural networks [13], [22], etc. Depending on the characteristics of the time-series at hand, the prediction performances could be totally different. In order to dynamically select the most appropriate prediction methods for different content popularity profiles, we propose hereafter a recommendation system framework composed of the following 3 sets. (1) A collection of time-series type generic data allowing to learn and predict. In our context, it represents YouTube popularity profiles. (2) Different prediction methods having multiple parameters configurations. The details of selected methods are described in Section III-A. (3) The recommendation framework which requires to randomly run a limited number of predictors on each content popularity profiles. Collaborative Filtering (CF) technique [16], [23], [24] will be used in order to estimate the similarities between the popularity profiles. This technique will be detailed in Sections III-B and IV.

C. Metric definition

Performances of prediction methods are evaluated based on the following error metrics:

- Mean Squared Error: $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2$
- Mean Absolute Error: $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - p_i|$
- Mean Absolute Percentage Error:

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - p_i|}{|y_i| + |p_i|} \times 100\%$$

where N is the total predicted samples, and y_i and p_i denote the real value and the predicted one, respectively. MSE and MAE tend to provide us the accuracy of the output while MAPE, slightly different, provides us the relative value of the error, which indicates the quality of the predictor. For example, with the same MAE = 1, the predictor in $y_i = 10$ has lower quality than the predictor in $y_i = 100$. To evaluate the quality of various predictors for the recommendation systems, we mainly use MAPE through its complements defined by:

- Accuracy: $ACC = 100\% - MAPE$

III. Preliminaries

A. Learning and prediction methods

We select for this study two categories of learning and prediction methods: tree-based regressors and RNN-based predictors, which can predict without retraining requirements.

1) Tree-based regressors: Tree-based learning algorithms are one of the most widely used supervised learning methods [19]. Tree-based learning uses predictive models in a tree form, called a *decision tree*. Learning a decision tree is essentially the process of splitting a training dataset based on recursive algorithms such as CHAID [25], ID3 [26], C4.5 [21] and CART[19]. Given a training dataset $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1..m}$,

the objective of a prediction problem is to find an estimation (hypothesis) function h that maps an input $\mathbf{X}$ to an output $\mathbf{Y}$. Learning a decision tree t is to find $h_t : \mathbf{X} \rightarrow \mathbf{Y}$ based on one of the above algorithms. Due to their simplicity, decision trees have high execution speed. But they cannot be generalized for an arbitrary problem with possible unseen data. For improving flexibility and diversity, Random Forest [27] uses multiple decision trees $\{t\}, t = 1, .., \tau$, called *weak learner*, constructed based on several randomly selected subspaces $\mathcal{S}_t$ of the training dataset $\mathcal{D}$, i.e., $\mathcal{S}_t \subset \mathcal{D}$. After individually training $\{h_t\}_t$, the output prediction for an input x' is computed by a discriminant function given by: $H_\tau(x') = \frac{1}{\tau}\sum_{t=1}^{\tau} h_t(x')$. The discriminant function is essentially an average one and the final predictor generalizes their prediction aggregately.

Boosting algorithms [20], [28] provide us with more complex frameworks for combining *weak learners*, e.g., decision trees. AdaBoost [28] applies a *weak learner* repeatedly over multiple rounds $t = 1..\tau$. For each round t, the *weak learner* h_t is trained, predicting error ϵ_t is computed and used for updating the weights of inputs for the next round $w_{t+1}^{(i)}$ and the weight α_t of h_t. $w_t^{(i)}$ is initialized equally but the weights of incorrect learning inputs, i.e., $w_t^{(i)}$ of $x^{(i)}$ with $h_t\left(x^{(i)}\right) \neq y^{(i)}$, get smaller in each round. Thus, the weak learner has to concentrate on the 'hard' inputs in the training set. In contrast, the weight α_t of h_t increases as predicting error ϵ_t decreases. The final output is determined as a weighted voting of *weak learners*.

Gradient Boosting method [29] is a more general boosting approach than AdaBoost. With the same input data set $\mathcal{D}$, the goal is to reconstruct the function $H^*(x) = \underset{H}{argmin}\mathbb{E}[L(y, H(x))]$, such that the loss function L is minimized. Gradient Boosting approximates $H^*(x)$ by a weighted sum of weak learners $h_t(x)$: $H(x) = \sum_{t=1}^{\tau} \alpha_t h_t(x)$. The steepest-descent algorithm is adapted for finding this approximation.

2) RNN-based predictors: RNN has been adopted in a wide range of machine learning tasks, particularly with time-related input or output [22]. Owing to the recurrent structures in which hidden states depend on both the current input and the network states at the previous time steps, RNNs are suitable for capturing the relationships among sequential data. In detail, the update of the recurrent hidden state of the simple RNN unit is given by: $h_t = g\left(x_t W_{xh} + h_{t-1} W_{hh} + b_h\right)$, where g is the activation function, x_t is the t-th input, W_{xh} is the input weight matrix, and W_{hh} is the recurrent weight matrix, b_h is the bias of the simple hidden unit. The simple unit outputs a distribution of the next element of the data sequence which is a composition of its previous state and its current input. Indeed, given a sequence $x = (x_1, x_2, ..., x_T)$, the sequence probability is given by: $p(x_1, x_2, ..., x_T) = p(x_1)p(x_2|x_1)...p(x_T|x_1, ..., x_{T-1})$. The simple RNN model provides an output which is equivalent to $h_t \sim p(x_t|x_1, ..., x_{t-1})$. Therefore, this model is able to capture the sequence probability of the input time-series data.

Due to the difficulties of training RNN for capturing long-term dependencies of time series data when gradients tend to vanish or explode, alternative units has been proposed. LSTM [30] and GRU [31] are the most popular ones. LSTM unit is formulated by the following equations:

$$i_t = \sigma\left(x_t W_{xi} + h_{t-1} W_{hi} + c_{t-1} W_{ci} + b_i\right) \tag{1a}$$
$$f_t = \sigma\left(x_t W_{xf} + h_{t-1} W_{hf} + c_{t-1} W_{cf} + b_f\right) \tag{1b}$$
$$\widehat{c}_t = \tanh\left(x_t W_{xc} + h_{t-1} W_{hc} + b_c\right) \tag{1c}$$
$$c_t = f_t c_{t-1} + i_t \widehat{c}_t \tag{1d}$$
$$o_t = \sigma\left(x_t W_{xo} + h_{t-1} W_{hi} + c_t W_{ci} + b_o\right) \tag{1e}$$
$$h_t = o_t \tanh\left(c_t\right) \tag{1f}$$

By introducing input gate i_t, forget gate f_t and output gates o_t, the memory content c_t, the output and current state of LSTM unit h_t are controlled. In detail, o_t controls the amount of memory content exposure c_t at the output h_t in (1f). f_t and i_t decide the updating amount of the new memory content $\widehat{c}_t$ and the forgetting amount of the old one c_{t-1} on the memory content c_t in (1d).

GRU unit is similar to LSTM unit with the use of gating blocks for adjusting the flows inside the unit but does not include memory cells [32]. GRU unit is formulated by the following equations. In a GRU unit, the update gate u_t adjusts the updates of its content h_t via (2d), whereas the reset gate r_t, similar to the forget gate in LSTM unit, decides to what extent the previous state h_{t-1} should be memorized in (2c).

$$r_t = \sigma\left(x_t W_{xr} + h_{t-1} W_{hr} + b_r\right) \tag{2a}$$
$$u_t = \sigma\left(x_t W_{xu} + h_{t-1} W_{hu} + b_z\right) \tag{2b}$$
$$\widehat{h}_t = \sigma\left(x_t W_{xh} + r_t h_{t-1} W_{hh} + b_h\right) \tag{2c}$$
$$h_t = \left(1 - u_t\right) h_{t-1} + u_t \widehat{h}_t \tag{2d}$$

B. Recommendation framework

A large number of predictors going along with a huge number of available contents profiles leads to a very large number of combinations. Testing all predictors for each content or determining one predictor suitable for all available contents is impossible. Therefore, the idea is to find a recommendation framework which automatically recommends a suitable predictor for each content, while testing a reduced number of predictors for each content. Accordingly, a recommendation system (RS) has proven to be a valuable technique that enables us to handle a large amount of data for estimating the most appropriate set of suggestions [33]. Multiple recommendation systems have been proposed including content-based RS, collaborative filtering (CF)-based RS, and hybrid RS [23]. Among them, CF which enables us to learn from historical interactions of the original *user-item* recommendation, could be adopted to our *content-predictor* selection problem.

For recommending appropriate items for a user, the similarity between user u and user v is computed based on Pearson correlation coefficient given by [34]:

$$sim\left(u, v\right) = \frac{\sum_{j=1}^{n}\left(r_{u,j} - \overline{r_u}\right)\left(r_{v,j} - \overline{r_v}\right)}{\sqrt{\sum_{j=1}^{n}\left(r_{u,j} - \overline{r_u}\right)^2}\sqrt{\sum_{j=1}^{n}\left(r_{v,j} - \overline{r_v}\right)^2}} \tag{3}$$

where n is the total number of items, $r_{u,j}$ is the rating of user u on item j, and $\overline{r_u}$ is the mean of the rating of user u.

Various ways for predicting a user's rating can be found in the literature such as the weighted sum of all other users' rating, Top-K recommendations, default voting, inverse user frequency, imputation-boosted CF, and weight majority prediction [24]. In this study, due to the specificity of our recommendation problem, we select the Top-K recommendation approach. The prediction of the rating of user u given to an item i based on the Top-K recommendation is given by:

$$pred\,(u,i) = \overline{r_u} + \frac{\sum_{v \in Top-K} sim\,(u,v)\,(r_{v,i} - \overline{r_v})}{\sum_{v \in Top-K} sim\,(u,v)} \quad (4)$$

where Top-K is the set of the K most similar users to user u.

For improving recommendation performances, we consider some additional imputation-boosted (IB) approach. The purpose of imputation is used for handling the pervasive problem of missing data [24], [35]. Then, we propose an IBCF-based recommending prediction method detailed in the next section.

IV. IBCF-based Recommending Prediction Method

Typically, for selecting a suitable predictor for a content, one needs to find out the performances of all predictors, each one being a couple of a prediction method and a set of parameters values. To avoid this costly process, we propose to train and measure, for each content, the prediction accuracy for a ratio of randomly chosen predictors during the test period. The accuracies of other predictors are estimated thanks to CF technique. The traditional recommendation method recommends for each *user* a well suited *item*. In our context, a *user* is assimilated to a *content* and an *item* to a well suited *predictor*. The *rating* that a user sets to an item is equivalent to the *prediction accuracy* that a predictor can provide to a content.

Table I summarizes our notations.

TABLE I: Notations

N	number of available predictors
P_n	a predictor $n = 1..N$
Pr	predictors set $\{P_{n=1..N}\}$
T	number of content profiles
$x_j^{(i)}$	solicitation number for the content i during day j
$X^{(i)}$	profile of the content i : $\left\{x_{j=1..m}^{(i)}\right\}$ between days 1 and m
$D_l^{(i)}$	training data for content i: $\left\{x_{j=1,..,l}^{(i)}\right\}$
$D_t^{(i)}$	testing data for content i: $\left\{x_{j=l,..,m}^{(i)}\right\}$
$ratio$	ratio of randomly selected predictors to run on a profile
$ACC_{i,p}$	accuracy value of the predictor p for the content i
ACC	accuracy matrix of size TxN
	In Eq.(3) **and Eq.** (4) **changes in signification**
u, v	any two content profiles
$sim(u,v)$	similarity value between contents u and v
$r_{u,i}$	accuracy of the predictor i for the content u : $ACC_{u,i}$
$\overline{r_u}$	mean accuracy for the content u

A conventional CF-based Recommendation Prediction Method (CF-RPM) and the proposed Imputation-Boosted CF-based Recommendation Prediction Method (IBCF-RPM) includes the following steps:

- S_1**:** Call Alg. 1 to compute the ACC matrix.
- S_2**:** Call Alg. 2 to compute the similarity matrix.
- S_3**:** Call Alg. 3 for both CF-RPM and IBCF-RPM to compute a *complemented ACC matrix*
- S_4**:** For a given profile i, recommend the predictor offering the highest value of ACC referred to the *complemented ACC matrix*.

Algorithm 1: ACC matrix computation

Data: Pr, $X^{(i)}$, $D_l^{(i)}$, $D_t^{(i)}$, $ratio$
for *each profile:* $i = 1..T$ **do**
  $P_i \leftarrow SelectRandomSubset(Pr, ratio)$;
  for p *in* P_i **do**
    Training p with $D_l^{(i)}$;
    Compute $ACC_{i,p}$ of the trained p with $D_t^{(i)}$;
Result: Sparse ACC matrix with size $T \times N$

Algorithm 2: Similarity matrix computation

Data: Sparse ACC matrix $[ACC_{i,p}]$
for $u = 1, ..., T$ **do**
  for $v = 1, ..., T$ **do**
    $sim(u, v)$ based on Eq. (3)
Result: Similarity matrix $[sim_{u,v}]$ with size $T \times T$

Algorithm 3: CF-RPM and IBCF-RPM

Data: Sparse ACC matrix, similarity matrix $[sim_{i,i}]$
for $i_1 = 1, ..., T$ **do**
  $Top_K^{(i)} \leftarrow$ The K most similar items from $[sim_{i,i}]$;
  for $p = 1, ..., N$ **do**
    if $IsEmpty(ACC_{i,p})$ **then**
      for k *in* $Top_K^{(i)}$ **do**
        $r_{k,p} \leftarrow ACC_{k,p}$;
        if *the method is IBCF-RPM* **then**
          if $IsEmpty(ACC_{k,p})$ **then**
            $r_{k,p} \leftarrow min(ACC_{i,p} \forall i = 1, ..., T)$
      $ACC_{i,p} \leftarrow pred(p, i)$ based on Eq. (4)
Result: Complemented ACC matrix $[ACC_{i,p}]$

Due to the sparsity of the ACC matrix given in S_1, empty ACC data in the set of Top-K similar neighbors can occur. In CF-RPM, the empty data is ignored.

For handling the pervasive problem of missing data, in [36], [37], the authors proposed to adopt imputation for collaborative filtering (IBCF). IBCF implements the recommendations from imputed data instead of the original data. The imputed data is obtained thanks to machine learning algorithms [36]. However, unlike *rating* in *user-items* problems, the predicted ACC is more sensitive to magnitude. This makes the imputation task more difficult. Furthermore, adopting Pearson correlation coefficient is equivalent to imputing the missing ACC with the mean value of ACCs of each prediction method because of the normalization nature in Eq. 3. Therefore, for imputation-boosted CF in our problem, we propose to perform the imputation in S_4 instead of S_2.

We propose to impute the missing ACC of a given prediction method by the minimum value of ACCs of that method.

TABLE II: Predictor parameters

RNN-based predictors			Tree-based regressors		
	No. of Unit			Max. depth	No. of Estimators
Config.	Layer 1	Layer 2			
C1	1	0			
C2	7	0	Decision Tree	4	1
C3	14	0	AdaBoost	4	20
C4	1	2	GradientBoosting	4	300
C5	7	14	Random Forest	4	300
C6	14	28			

The minimum ACC is selected because it represents the worst accuracy that the predictor could provide a content. Then, the proposed IBCF can be considered as a *maxmin* problem if there are any missing data in the Top-K similar neighbors set. Without imputation, the predicted ACCs for more missing data cases and less missing data cases are treated similarly. The imputation, hence, increases the prediction reliability. Notice that the processing cost of the imputation is ignorable because it is just an insertion of a minimum value along with computational process.

V. Performance Evaluation

In this section, we perform the evaluation for individual prediction methods presented in Sect. III-A and the proposed IBCF-RPM described in Sect. IV.

A. Context of simulation

The simulation is based on the YouTube profiles dataset, presented in Sect. II-A. For predicting daily access number of YouTube videos, we use the slicing window prediction style. The number of access to a YouTube video at the consequent day is predicted based on the historical accessing information extracted from a look back window of W preceding days. For example, with $W = 3$, the predicting output at day t is estimated based on the real data at day $t-3$, $t-2$, and $t-1$. In our simulation, the value of W is selected in the set of $\{1, 2, 3, 5, 7, 14, 30\}$ day(s).

In this paper, an individual predictor is defined as a couple of a prediction method and an appropriate set of parameters values. Without loss of generality, the parameters values for individual prediction methods are loosely chosen because we aim at the RS which is able to recommend the most appropriate method for a content. The selected parameters given in Table II combined with the 7 above cases of W form 28 tree-based predictors and 84 RNN-based predictors.

B. Performances of individual predictors

We evaluate the performances of the individual predictors on the YouTube dataset presented in Sec. II-A. For each content profile, a predictor is trained on the training period and then tested on the testing period. Afterward, it is validated on the validation period.

History and configurations. Figure 2 shows the average MSE of predictors on the testing period. Because of lack of space, the average MAE could not be displayed, but we observed that the trends of MAE and MSE are consistent. With the same W, LSTM-based and GRU-based predictors have lower errors than the tree-based predictors. Within the tree-based predictors, the Random Forest one is the best. Within the RNN-based predictors, the configuration C3 with only one layer is better than the configuration C6 with two unit layers. With the same configuration and W, the predicting errors of LSTM-based predictors and GRU-based predictors are almost similar. Decision Tree and Random Forest predictors, which are simpler, have errors increasing with W. AdaBoost and Gradient Boosting show little differences in error when $W = 1..7$ and the error for $W = 30$ is smaller than the error for $W = 14$. For RNN-based predictors, the error for $W = 30$ is the smallest one.

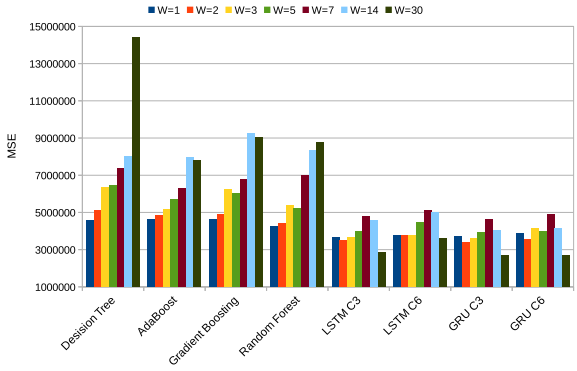

Fig. 2: MSE-Average testing errors of individual predictors

ACC on training, testing and validation period. Table III shows that the average ACC of the individual predictors on the training set is higher than those on testing and validation sets. For tree-based predictors, the highest ACC corresponds to $W = 30$ in the training set, and to $W = 1$ in both testing and validation sets, which is obvious. For RNN-based predictors, the best ACC is obtained when $W = 7$ in the validation set, mostly with $W = 1$ and $W = 7$ in testing set, and with $W = 7$ and $W = 30$ in training set. In summary, the ACC of a predictor changes with the set of data. Through training, a predictor can achieve high ACC but in out-of-sample predicting (i.e., predicting in testing and training sets), the ACC is generally lower.

C. Performances of CF-RPM and IBCF-RPM

The performances of CF-RPM and the proposed IBCF-RPM are compared with the best individual predictors during testing. After the training phase, all individual predictors were tested, *a full information prediction accuracy matrix* was created. Based on the best value of ACC in this matrix, the set of the best ACC predictors for each profile was selected. In Fig. 3 and 4, the *"Best recommend (Full information)"* curve represents the best ACC obtained by the optimal choice of individual predictors after testing phase.

The graphics in Fig. 3 shows the ACC obtained by different predictors during testing and validating phase. We can see in the first graphic of 3 that the proposed IBCF-RPM achieves almost the best value of ACC, whereas the CF-RPM can only achieve around the ACC of the best individual predictor. The highest testing ACCs of individual predictors in each

TABLE III: Average ACC of individual predictors

	Training					Testing					Validation				
	W=1	W=3	W=7	W=14	W=30	W=1	W=3	W=7	W=14	W=30	W=1	W=3	W=7	W=14	W=30
DecisionTree	94.97	95.24	95.70	95.87	***96.06***	***88.96***	87.92	87.54	86.42	84.64	***87.48***	86.68	86.46	84.97	81.60
AdaBoost	94.38	95.15	96.07	96.45	***96.89***	***88.06***	87.63	87.43	86.34	85.28	***86.60***	86.37	86.29	84.68	82.43
Grad.Boosting	95.61	96.36	97.07	97.45	**97.82**	***88.47***	87.83	87.11	85.82	84.37	***86.94***	86.41	85.82	83.76	81.28
RandomForest	95.04	95.38	95.93	96.12	***96.29***	***89.35***	88.61	88.03	86.91	85.42	***87.76***	87.26	86.86	85.35	82.55
LSTM_C1	90.59	92.40	93.37	93.08	***93.42***	84.84	87.43	***88.82***	87.87	88.06	83.65	87.05	89.07	87.65	87.27
LSTM_C2	93.47	93.52	93.96	93.94	***94.04***	***90.22***	90.06	89.99	89.80	89.31	89.09	89.30	90.41	90.01	88.51
LSTM_C3	93.30	93.64	***94.05***	93.89	93.90	90.03	***90.28***	90.07	89.57	89.52	88.89	89.60	90.61	90.10	88.76
LSTM_C4	91.66	92.41	***92.91***	92.60	92.86	85.76	87.46	***87.69***	86.79	85.84	84.18	86.75	87.61	86.24	84.30
LSTM_C5	93.13	93.03	93.67	93.57	***93.85***	***89.66***	89.32	89.49	88.82	88.71	88.59	88.86	89.50	88.81	87.84
LSTM_C6	93.05	92.88	***93.66***	93.08	93.56	***89.58***	88.93	89.10	88.71	88.84	88.76	88.70	89.43	89.21	87.75
GRU_C1	91.63	92.69	93.22	92.96	***93.36***	86.20	88.36	***88.93***	88.06	88.35	84.95	87.97	***89.17***	88.00	87.68
GRU_C2	93.51	93.61	94.00	93.66	***94.08***	**90.55**	90.31	90.19	89.45	89.50	89.61	89.77	**90.90**	90.07	88.86
GRU_C3	93.30	93.53	***93.95***	93.42	93.76	89.98	90.10	***90.35***	89.12	89.34	88.99	89.67	***90.91***	89.84	88.50
GRU_C4	92.95	92.69	***93.15***	92.91	93.03	***88.81***	88.42	88.75	87.87	87.71	87.86	88.01	***89.24***	88.29	86.87
GRU_C5	93.17	93.21	93.44	93.16	***93.52***	***90.28***	89.92	89.45	88.65	89.15	89.23	89.56	***89.94***	88.93	88.40
GRU_C6	92.74	92.94	93.12	93.10	***93.31***	***89.53***	89.31	89.27	88.63	89.21	88.59	88.92	***89.78***	88.68	88.38

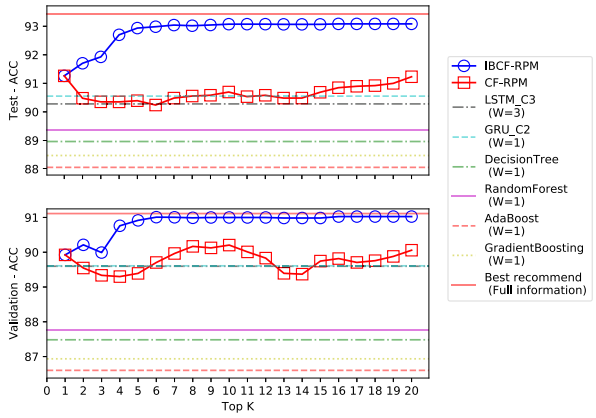

Fig. 3: Accuracy vs. Top-K with running ratio=0.3

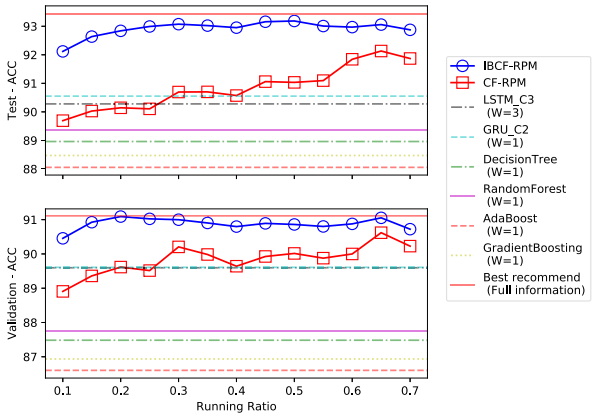

Fig. 4: Accuracy vs. running ratio with Top-K=10

prediction type are shown for comparison. For example, the GRU-based predictor has the highest testing ACC in configuration C2 for $W = 1$. This value can be also found in Table III. The trend of prediction accuracy of the proposed IBCF-RPM is almost stable for $K > 6$. For $K = 1$, the ACCs of IBCF-RPM and CF-RPM are similar because the imputation is not effective in this case.

The ACC obtained during validation phase is also depicted in the second graphic in Fig. 3. It should be noted that for these graphics, we selected only 30% of the defined predictors to compute their accuracy. The accuracy of the remaining 70% is estimated and the well-suited predictor is selected based on estimated or real accuracy of the Top-K most similar contents. The depicted curves of the second graphics of 3 indicate the effectiveness of the predictors selection during validation phase, based on the prediction performances during the testing phase. Above all, our proposed method achieves almost an ACC as high as that of the best recommendation with full information.

Figure 4 confirms the efficiency of the proposed method. The selection ratio represents the percent of randomly selected predictors to be executed for a given content, the accuracy of the remaining predictors being estimated based on the accuracy of similar contents. For all of the investigated values of implementing ratio, ranging from 0.1 to 0.75, IBCF-RPM always approaches the value of the best recommendation with full information, when the implementing ratio equals to 1. In addition, IBCF-RPM is better than CF-RPM and any individual predictors in both testing and validating set.

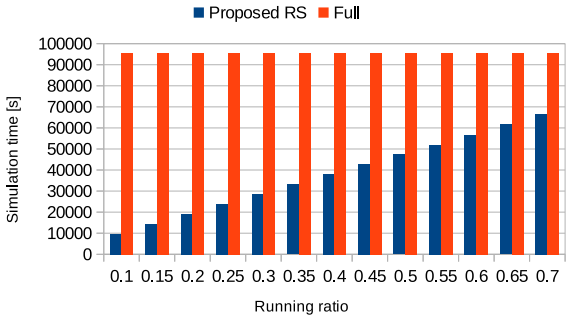

Fig. 5: Simulation time vs. running ratio

Figure 5 shows the time efficiency of the proposed IBCF-RPM. Compared to the best recommendation, where training and testing are required for all predictors, our method has very close performances while running only a small ratio of available predictors.

VI. Conclusion

In this paper, we have addressed the problem of predicting content popularity based on the study of YouTube contents. We have proposed a recommendation framework which allows recommending appropriate predictors for each content solicitation profile. The framework is based on the proposed IBCF technique. Our method, IBCF-RPM, is applicable to a large set of contents and prediction methods. We considered tree-based regressors (Decision Tree, Random Forest, Gradient Boosting and AdaBoost) and Recurrent Neural Network-based prediction methods (LSTM-based, GRU-based) as individual predictors. IBCF-RPM recommends a well-suited predictor for a given content while testing the accuracy of a small ratio of the available predictors, randomly chosen. The method is based on the evaluation of the similarities between contents to estimate the accuracy of untested methods. For a running ratio of 30%, based on similarities with a quite low number (Top-K=6) of other contents, IBCF-RPM gives performances close to the full implementation case.

For future work, the effectiveness of the popularity prediction for proactive caching in MEC should be further investigated and evaluated.

Acknowledgement

The research leading to these results has received funding from DigiCosme.

References

[1] N. Al-Falahy and O. Y. Alani, "Technologies for 5g networks: Challenges and opportunities," *IT Professional*, vol. 19, no. 1, pp. 12–20, Jan 2017.

[2] E. Zeydan, E. Bastug, M. Bennis, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data caching for networking: Moving from cloud to edge," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 36–42, 2016.

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, Fourthquarter 2017.

[4] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.

[5] C. V. networking Index, "Forecast and methodology, 2016-2021, white paper," *San Jose, CA, USA*, vol. 1, 2016.

[6] A. Tatar, M. D. de Amorim, S. Fdida, and P. Antoniadis, "A survey on predicting the popularity of web content," *Journal of Internet Services and Applications*, vol. 5, no. 1, pp. 1–20, 2014.

[7] N. B. Hassine, D. Marinca, P. Minet, and D. Barth, "Popularity prediction in content delivery networks," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symposium on*. IEEE, 2015, pp. 2083–2088.

[8] N. B. Hassine, R. Milocco, and P. Minet, "Arma based popularity prediction for caching in content delivery networks," in *Wireless Days, 2017*. IEEE, 2017, pp. 113–120.

[9] L. Deng, "Achievements and Challenges of Deep Learning," Tech. Rep., 2015. [Online]. Available: https://www.microsoft.com/en-us/research/publication/achievements-and-challenges-of-deep-learning/

[10] Y. Statistics, "Url" http://www. youtube. com/yt/press/statistics. html," *Accessed in January*, 2018.

[11] J. W. Taylor and R. D. Snyder, "Forecasting intraday time series with multiple seasonal cycles using parsimonious seasonal exponential smoothing," *Omega*, vol. 40, no. 6, pp. 748–757, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.omega.2010.03.004

[12] E. S. Gardner Jr and E. McKenzie, "Forecasting trends in time series," *Management Science*, vol. 31, no. 10, pp. 1237–1246, 1985.

[13] T. Hill, M. O'Connor, and W. Remus, "Neural network models for time series forecasts," *Management science*, vol. 42, no. 7, pp. 1082–1092, 1996.

[14] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *International Conference on Machine Learning*, 2017.

[15] C. Conflitti, C. D. Mol, and D. Giannone, "Optimal combination of survey forecasts," *International Journal of Forecasting*, vol. 31, no. 4, pp. 1096–1103, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207015000606

[16] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261–273, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1110866515000341

[17] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan, "Youtube-8m: A large-scale video classification benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[18] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, no. just-accepted, 2017.

[19] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[20] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[21] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[22] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[23] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, June 2005.

[24] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," *Advances in Artificial Intelligence*, vol. 2009, no. Section 3, pp. 1–19, 2009. [Online]. Available: http://www.hindawi.com/journals/aai/2009/421425/

[25] G. V. Kass, "An exploratory technique for investigating large quantities of categorical data," *Applied statistics*, pp. 119–127, 1980.

[26] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[27] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1. IEEE, 1995, pp. 278–282.

[28] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *European conference on computational learning theory*. Springer, 1995, pp. 23–37.

[29] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[32] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," pp. 1–9, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[33] E. Bigdeli and Z. Bahmani, "Comparing accuracy of cosine-based similarity and correlation-based similarity algorithms in tourism recommender systems," in *2008 4th IEEE International Conference on Management of Innovation and Technology*, Sept 2008, pp. 469–474.

[34] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender systems: an introduction*. Cambridge University Press, 2010.

[35] R. J. A. Little, "Missing-data adjustments in large surveys," *Journal of Business & Economic Statistics*, vol. 6, no. 3, pp. 287–296, 1988. [Online]. Available: http://www.jstor.org/stable/1391878

[36] X. Su, T. M. Khoshgoftaar, X. Zhu, and R. Greiner, "Imputation-boosted collaborative filtering using machine learning classifiers," *Proceedings of the 2008 ACM symposium on Applied computing - SAC '08*, no. 2, p. 949, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1363686.1363903

[37] W. Xia, L. He, J. Gu, K. He, and L. Ren, "Boosting collaborative filtering based on missing data imputation using item's genre information," in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, Aug 2009, pp. 332–336.