Forecasting Content Popularity: a Recommendation Method for Aggregating Predictors

Nhan Nguyen-Thanh, Kinda Khawam, Elena Simona Lohan, Dana Marinca, Steven Martin, Dominique Quadri, Lila Boukhatem

▶ **To cite this version:**

Nhan Nguyen-Thanh, Kinda Khawam, Elena Simona Lohan, Dana Marinca, Steven Martin, et al.. Forecasting Content Popularity: a Recommendation Method for Aggregating Predictors. Knowledge and Information Systems (KAIS), In press. hal-02284831

HAL Id: hal-02284831
https://hal.science/hal-02284831

Submitted on 12 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Noname manuscript No.
(will be inserted by the editor)

Forecasting Content Popularity: a Recommendation Method for Aggregating Predictors

Nhan Nguyen-Thanh · Kinda Khawam · Elena Simona Lohan · Dana Marinca · Steven Martin · Dominique Quadri · Lila Boukhatem

Received: date / Accepted: date

Abstract Currently, Video on Demand (VoD) is the heaviest data-driven service among various multimedia services. Therefore, predicting the popularity of multimedia contents, particularly video contents, in order to supply their proactive caching is a crucial issue. Mobile Edge Computing is proposed as a general solution for the upcoming 5G-based networks to allow computation and storage capabilities at the edge of Radio Access Network. Henceforth, applications and services can be deployed near the end-user, improving the backhaul resources utilization, energy consumption, and users Quality of Experience thanks to the reduction of traffic latency. Therefore, the pivotal problem addressed in this paper is to find the most appropriate prediction methods from a large set of predefined methods, able to forecast, with high accuracy, the multimedia content popularity. A real YouTube dataset was used for simulations. The proposed Recommendation System framework is very flexible because it can be extended to different time-series prediction contexts. It also displays good scalability and can be dynamically adapted for analyzing huge sets of time-series with large sets of predictors.

1 Introduction

Data traffic explosion, stemming from mobile multimedia services [3], social networks, and over-the-top applications such as YouTube, has imposed a huge challenge to 5G and beyond-5G networks for providing ever more backhaul resources [44]. By 2023, it is expected to have 1 billion 5G subscriptions, and the mobile data traffic (of which 73% is forecast as mobile video traffic) could increase up to 107

N. Nguyen-Thanh, S. Martin, D. Quadri, L. Boukhatem
LRI, University Paris-Saclay, Paris-Sud, France
E-mail: nhan.nguyen@lri.fr; steven.martin@lri.fr; dominique.quadri@lri.fr;lila.boukhatem@lri.fr

K. Khawam, D. Marinca
David Lab., University Paris-Saclay, UVSQ, France
E-mail: kinda.khawam@uvsq.fr; dana.marinca@uvsq.fr

Elena Simona Loha
Tampere University of Technology, Finland
E-mail: elena-simona.lohan@tut.fi

exarbytes per month (seven-fold growth in 2017 mobile data traffic) [8]. Mobile Edge Computing (MEC), currently standardized by ETSI [33], is part of the effort toward a general solution for 5G-based network architectures. MEC allows computation and storage capabilities at the edge of Radio Access Network (RAN) [1], 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 the backhaul offloading and enhance users' Quality of Experience (QoE) owing to traffic latency reduction.

Video on Demand is the heaviest data-driven multimedia service causing up to 80% of backhaul traffic by the year 2020 [24]. In the near future, immersive format videos such as 360-degree YouTube videos would consume four-to-five times more bandwidth than the standard video streams with the same resolution [8]. Therefore, it is vital to predict the popularity of multimedia contents, particularly videos, for their proactive caching.

The popularity profile of a multimedia content is explicitly indicated by the number of daily solicitations. Since the solicitation evolution of a multimedia content is represented as time-series data, the prediction of the popularity profile is basically a time-series prediction. Many previous works [18, 19, 42] have tackled this problem. However, most of devised solutions are based only on a single prediction approach which is generally not well-adapted for this kind of big-data application containing billions of multimedia contents such as YouTube. For instance, exponential smoothing approach [18,43], Auto Regressive Integrated Moving Average (ARIMA) [17,19], Neural Network (NN) and Recurrent Neural Network (RNN) [9, 20, 28, 30, 45], tree-based regressors [42], etc., were used for predicting such kind of time-series. The performance of these methods varies widely regarding the same time-series. Hence, finding the mechanism that optimally activates the most efficient prediction algorithm for a given context is paramount. Combinations of several forecasters in order to formulate a universal predictor keeping individual forecasters' advantages have been proposed in several previous works [5, 7, 12, 15, 16, 21]. Ensemble techniques such as Bagging technique, e.g., Random Forest Algorithm [21], and Boosting technique, e.g., AdaBoost [14] or Gradient Boosting method [15], are used for the aggregation process. However, these methods only exploit the internal relation of weak predictors on different samples of the original dataset. The external relation between elements of the dataset for selecting appropriate predictors, was wrongly overlooked. To the best of our knowledge, this is the first work on combining different types of predictors for big data analytics.

In our previous work [37], we have proposed a recommendation framework for adaptively and dynamically selecting suitable prediction methods for time-series content profile. Although the presented framework is well-adapted and well-performing, the scalability issue of the adopted recommendation system [25] is a big challenge when applied to large time-series datasets. To overcome this problem, in this work, we propose a light recommendation framework for time-series predictors. This framework is based on the evaluation of various time-series similarity against a reference-set of time-series. The diversity of several prediction methods is exploited to enhance the prediction accuracy. We will consider two approaches. In the first one, a set of the most appropriate predictors are selected for each time-series. The predicted output will then be the aggregation of the selected pre-

dictors. In the second approach, several similarity-evaluating methods are adopted to propose multiple predictors for the output aggregation.

The remainder of the paper is organized as follows. In Section 2, we define the problem of multimedia content popularity prediction, the YouTube dataset that we put to test, and the developed methodology. In Section 3, we focus on a selected set of single prediction methods. In Section 4, we present our proposed recommendation system for selecting appropriate predictors for time-series. In Section 5, we present the simulation context and performance results. Finally, in Section 6, we conclude the study and mention its perspectives.

2 Problem statement

In this work, we address the problem of selecting the most appropriate predictor from a set of well-known methods to predict the *popularity evolution* of each multimedia content from a huge collection of contents (e.g. offered by a Content Delivery Network (CDN)). 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 large set of prediction methods going along with several configurable parameters lead to a huge number of predictors. Applying all available predictors to each item of a very large catalog of contents (such as YouTube), in order to single out the best one, is intractable. To overcome this problem, we propose hereafter a recommendation framework able to recommend a suitable predictor for each multimedia content profile of a given catalog. The most appropriate predictor will be identified based on (1) the computed prediction accuracy of all methods on a reference set, composed by a low number of content profiles and (2) the similarities between the content profiles in the catalog and the reference set. A *multimedia content profile popularity* represents the number of *daily requests* for this content and the associated characteristics (e.g., number of *likes*). The number of daily requests for a content impacts explicitly the decision of caching it to the nearby servers in 5G MEC. Because the number of daily requests represents basically time-series data, we focus on time-series prediction, that we deem in the rest of the paper *multimedia content profile* or just *content profile*.

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 different time-series prediction contexts.

2.1 YouTube dataset - Contents Profile Popularity

In this study, the dataset is composed of real traces crawled from the YouTube site. A trace associated with 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 [2], composed of approximately 8 million videos. The crawled dataset includes contents' traces having at least 10000 total solicitations. For this study,

we randomly selected traces of 200 contents having over 1.5 million requests and at least 500 solicitations per day. Fig.1 displays the number of daily solicitations for our selected videos (the legend shows their YouTube's IDs). The traces start at the uploading time (after 2007) and end at the crawling time (Sept. 2017).

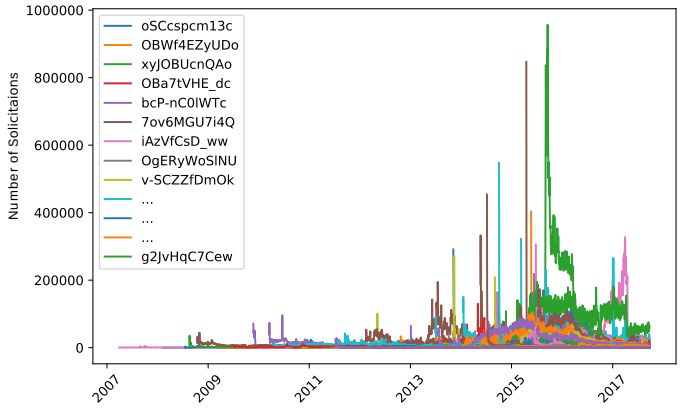


Fig. 1 YouTube content solicitation traces

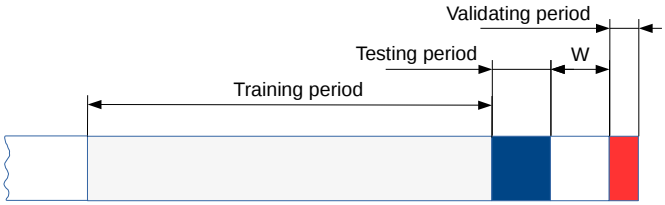


Fig. 2 Content profile period splitting

Each content profile in the selected set is divided into three periods as shown in Fig.2. The *training period*, constituted of several consecutive days, represents the training data. The *testing period* composed of consecutive days that directly follow the first one. Finally, the *validation period* composed of several consecutive days and separated from the testing period by a window W of variable length. The separating window W is used to ensure the objectivity of the validation. The training period is kept as long as possible in order to preserve the possibly seasonal relations among solicitation traces. This is the usual operating mode in any time-series prediction problem.

2.2 Prediction accuracy evaluation

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 number of total predicted samples of a time-series, and y_i and p_i denote the real value and the predicted one, respectively.

MSE and MAE provide the accuracy of the output while MAPE, slightly different, provides the relative value of the error prediction, which reveals 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 system, we mainly use MAPE through its complement, defined by:

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

2.3 Distance Measures For Time-Series

In this work, we use various kinds of distance measures for evaluating the similarity between time-series. Given two time-series $X = \{x_0, x_1, ..., x_{N-1}\}$ and $Y = \{y_0, y_1, ..., y_{N-1}\}$, the investigated distances, presented in this section, are defined in the following set: $\{d_{L_p}$, d_{MAN}, d_{EUCL}, d_{DTW}, d_{DTW_W}, d_{JAC}, d_{DIS}, d_{PDIS}, d_{CID}, d_{ACF}, d_{PACF}, d_{CORR} $\}$.

2.3.1 L_p Distance

The L_p distance (d_{L_p}) [REF] is given by:

$$d_{L_p} = \begin{cases} \sum_{i=0}^{N-1}|x_i - y_i| & p = 1 \\ \sqrt[p]{\sum_{i=0}^{N-1}(x_i - y_i)^{\frac{1}{p}}} & 1 < p < \infty \\ \max_{i=0,...,N-1}|x_i - y_i| & p = \infty \end{cases} \quad (1)$$

We consider two cases with $p = 1$ and $p = 2$ which correspond to *Manhattan Distance* (d_{MAN}) and *Euclidean Distance* (d_{EUCL}), respectively.

2.3.2 Dynamic Time Warping Distance

The Dynamic Time Warping distance (d_{DTW}) [27] is relative to time-series comparison. This allows a non-linear mapping of two vectors by minimizing the distance between them.

Given two vectors with different lengths: $X = x_0, ..., x_{n-1}$ and $Y = y_0, ..., y_{m-1}$, a cost matrix C, size $[n \times m]$, containing the distances (usually Euclidean distance) between two points $x_i, 0 \leq i < n$ and $y_j, 0 \leq j < m$ is established. A time warping path $P = \{p_0, ..., p_{K-1}\}$, where $\max(m, n) \leq K < m + n - 1$, is formulated based on the following conditions:

- Boundary: $p_1 = C(1,1)$ and $p_{K-1} = C(n,m)$;
- Monotonicity: given $p_k = C(i_k, j_k)$ and $p_{k-1} = C(i_{k-1}, j_{k-1})$, then $i \geq i_{k-1}$ and $j \geq j_{k-1}$;
- Step size: given $p_k = C(i_k, j_k)$ and $p_{k-1} = C(i_{k-1}, j_{k-1})$, then $i_k - i_{k-1} \leq 1$ and $j_k - j_{k-1} \leq 1$.

Many paths satisfying the above conditions could be found. However, only the warping path that minimizes the cost is considered as d_{DTW} distance:

$$d_{DTW} = \min\left(\sqrt{\sum_{k=0}^{K-1} p_k}\right) \tag{2}$$

The computational cost of Dynamic Time Warping algorithm dedicated to finding the path of minimal cost is the main drawback of such kind of distance. To overcome this difficulty, the "Sakoe-Chiba band" [40], which defines the set of points available for associating the warping path, is adopted. The Sakoe-Chiba band runs along the main diagonal, i.e., $i = j = \iota$, and constraints the range of warping as specified by the fixed width W, $W < \max(m,n)$. The upper and lower boundaries of the band is given by:

$$|i_k - \iota| \leq W, |j_k - \iota| \leq W$$

where ι is the corresponding coordinate at the main diagonal and k indicates the k-th point of the warping path. We denote by d_{DTW_W} the DTW derivative with Sakoe-Chiba band.

2.3.3 Jaccard Distance

Jaccard distance (d_{JAC}) [29] is given by:

$$d_{JAC} = 1 - \frac{|X \cap Y|}{|X \cup Y|} \tag{3}$$

where $|.|$ is the set cardinality.

2.3.4 Dissimilarity Distance

Canberra Dissimilarity distance (d_{DIS}), a weighted version of L_1 (Manhattan) distance, is given by:

$$d_{DIS} = \frac{1}{N} \sum_{i=0}^{N-1} \frac{|x_i - y_i|}{|x_i| - |y_i|} \tag{4}$$

A derivative of Dissimilarity distance is the Peak Dissimilarity distance (d_{PDIS}) which is defined by:

$$d_{PDIS} = \frac{1}{N} \sum_{i=0}^{N-1} \frac{|x_i - y_i|}{2\max(|x_i|, |y_i|)} \tag{5}$$

2.3.5 Complexity-Invariant Distance

Complexity-Invariant distance (d_{CID}) [4] was introduced to adjust an existing dissimilarity measure by a correction factor (CF) computed based on information about complexity difference between two series. We consider CID distance based on Euclidian distance as follows:

$$d_{CID} = d_{EUCL}(X, Y) \times CF(X, Y) \tag{6}$$

where $CF(X, Y) = \frac{\max(CE(X), CE(Y))}{\min(CE(X), CE(Y))}$ and $CE(X)$ is a complexity estimate of time-series X, given by: $CE(X) = \sqrt{\sum_{t=0}(x_{t+1} - x_t)^2}$.

2.3.6 Feature-based distances

In this distance category, the similarity between extracted features (such as Fourier, Wavelet, Correlation coefficients, etc.) of time-series is utilized instead of their raw values. For this kind of distance, we consider distance measures of features achieved by Autocorrelation function (ACF) and Partial correlation function (PACF) [34]:

$$d_{(P)ACF} = \sqrt{\sum_{l=1}^{L}(\rho_X - \rho_Y)^2} \tag{7}$$

where ρ_X and ρ_Y are the ACF/PACF coefficients of time-series X and Y, whereas L is the lag number.

We also investigate energy distances through Pearson's Correlation distance (d_{PC}) [23] and Correlation distance (d_{CORR}) [41]. The Pearson's Correlation distance is given by:

$$d_{PC} = 1 - \frac{\sum_{i=0}^{N-1}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=0}^{N-1}(x_i - \overline{x})}\sqrt{\sum_{i=0}^{N-1}(y_i - \overline{y})}} \tag{8}$$

where N is the sample size, and $\overline{x}$ is the sample mean.

The d_{PC} is sensitive to the linear relationship between two variables. However, it can be zero for dependent variables.

The d_{CORR} [41] was introduced to address the deficiency of d_{PC}. The d_{CORR} is zero if and only if the random vectors are independent. In other words, d_{CORR} measures both linear and nonlinear association between two random variables or random vectors. This is in contrast with Pearson's correlation, which can only detect linear association between two random variables. The d_{CORR} is given by:

$$d_{CORR} = \frac{\nu^2(X, Y)}{\sqrt{\nu^2(X)\,\nu^2(X)}} \tag{9}$$

where $\nu^2(X, Y)$ is the distance covariance, and $\nu^2(X)$ and $\nu^2(Y)$ are the distance variance of X and Y, respectively.

3 Individual Predictors

In this section, we present the theoretical basis of two categories of learning prediction methods which can predict without retraining requirement: tree-based regressors and RNN-based predictor. Decision tree, Random Forest, AdaBoost, Gradient Boosting regressors are the selected predictors among several ones of the former type. Long Short Term Memory (LSTM) and Gate Recurrent Unit (GRU) are the two selected units for constructing the second type.

3.1 Tree-based regressors

Tree-based learning algorithms are one of the most widely used supervised learning methods. They can be used for solving either classification or regression problems as analyzed in Classification And Regression Tree (CART) [6]. Tree-based learning uses predictive models in a tree form, i.e., flowchart-like form, called decision trees. Decision trees that are given continuous output values allow to deal with prediction problems, hence, are the prediction tools of this study. Learning a decision tree is essentially a process of splitting a training dataset based on recursive algorithms such as CHAID [26], ID3 [38], C4.5 [39] and CART.

In prediction problems, the system is composed of an input variable $\mathbf{X}$ and an output variable $\mathbf{Y}$. Given a training dataset $\mathcal{D} = \{x^{(i)}, y^{(i)}\}, i = 1, ..., m$, the objective is to find an estimation or a hypothesis function h that maps $\mathbf{X}$ to $\mathbf{Y}$. 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. Still, they cannot be generalized for an arbitrary problem with possible unseen data. For improving flexibility and diversity, Random Forest [21] uses multiple decision trees $\{t\}, t = 1, ..., T$, 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 $\{h_t\}$, the output prediction for an input x' is computed by a discriminant function given by:

$$H_T\left(x'\right) = \frac{1}{T}\sum_{t=1}^{T} h_t(x') \tag{10}$$

The discriminant function is essentially an averaging function and the final predictor generalizes the predictions aggregately.

Instead of using a simple averaging function, boosting algorithms [5, 14, 16] provide us with more complex frameworks by combining *weak* or *base learners*, e.g., decision trees. AdaBoost [14], which stands for Adaptive Boosting, performs *weak learners* repeatedly over multiple rounds $t = 1, ..., T$. The detailed AdaBoost algorithms is presented in Algorithm 1. For each round t, the *weak learner* h_t is trained, the predicting error ϵ_t is computed and used to update weights for inputs of the next round $w_{t+1}^{(i)}$ and weights of *weak learners* α_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 larger on each round. Thus, the weak learner has to concentrate on the 'hard' inputs in the training set. In contrast, the weight α_t increases as the predicting error ϵ_t decreases. This means more contribution is assigned to h_t because the final output is based on a weighted voting of *weak learners*.

Algorithm 1: The AdaBoost algorithm [13]

Data: Training set of m labeled samples $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}, i = 1, ..., m$
Initialize: $w_1^{(i)} \leftarrow 1/m \ \forall i = 1, ..., m$;
for $t = 1, ..., T$ **do**
  $h_t \leftarrow WeakLearn(\mathcal{D}, \mathbf{w}_t)$;
  $\epsilon_t \leftarrow \sum_{i=1}^{m} w_t^{(i)} \mathbb{1}[h_t(x^{(i)}) \neq y^{(i)}]$;
  for *i=1,...,m* **do**
    $w_{t+1}{}^{(i)} \leftarrow \frac{w_t^{(i)}}{2} \times \begin{cases} \frac{1}{\varepsilon_t} & \text{if } h_t\left(x^{(i)}\right) \neq y^{(i)} \\ \frac{1}{1-\varepsilon_t} & \text{if } h_t\left(x^{(i)}\right) = y^{(i)} \end{cases}$
  end
  $\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
end
Result: Final classifier: $H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

The Gradient Boosting method [15] is a more general boosting approach than AdaBoost. With the same input data set $\mathcal{D}$, the goal is to reconstruct the function H^*,

$$H^*(x) = \underset{H}{argmin}\mathbb{E}[L(y, H(x))],$$

such that the loss function L is minimized. Gradient Boosting approximates $H^*(x)$ by a weighted sum of weak learners $h(x, \rho)$:

$$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x).$$

The steepest-descent algorithm is adapted for finding this approximation as presented in Algorithm 2.

Algorithm 2: The Gradient Boosting algorithm [15]

Data: Training set of m labeled example $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}, i = 1, ..., m$,
a differentiable loss function $L(y, H(x))$. Initialize: $H_0(x) = \underset{\alpha}{argmin} \sum_{i=1}^{m} L\left(y^{(i)}, \alpha\right)$;
for $t = 1, ..., T$ **do**
  1. Compute pseudo-residuals: ;
  $r_t^{(i)} = -\left[\partial L\left(y^{(i)}, H(x^{(i)})\right)/\partial H(x^{(i)})\right]_{H(x)=H_{m-1}(x)}$, $i = 1, ..., m$;
  2. Fit weak learner $h_t(x)$ to pseudo-residuals;
  $h_t(x) \leftarrow WeakLearn\left(\left\{\left(x^{(i)}, r_t^{(i)}\right)\right\}\right), i = 1, ..., m$;
  3. $\alpha_t = \underset{\alpha}{argmin} \sum_{i=1}^{m} L\left(y^{(i)}, H_{m-1}(x^{(i)}) + \alpha h_t(x^{(i)})\right)$;
  4. $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$;
end
Result: Final classifier: $H_T(x)$

3.2 RNN-based predictors

Recurrent Neural Networks have been adopted in a wide range of machine learning tasks, particularly, when they require handling time-related data in either input or output, such as image captioning, music generation, speech recognition, handwriting recognition, machine translation and times-series prediction [31,35]. RNNs are suitable for capturing the relationship between sequential data points owing to their recurrent structures. In detail, recurrent hidden states depend on both the current input and the network states at the previous time steps, instead of only the current input as in a conventional feed-forward neural network. This structure is demonstrated by a simple RNN unit in Fig. 3a. The update of the recurrent hidden state of the simple RNN unit is given by:

$$h_t = g\left(x_t W_{xh} + h_{t-1} W_{hh} + b_h\right) \tag{11}$$

where g is the activation function, x_t is the t-th input, b_h is the bias of the simple hidden unit, W_{xh} is the input weight matrix, and W_{hh} is the recurrent weight matrix.

The simple unit outputs a distribution of the next element of the sequence data which is a composition of its previous state and its current input. Thus, this model is able to capture the sequence probability of the input time-series data. Indeed, given a sequence $x = (x_1, x_2, ..., x_T)$, the sequence probability is given by:

$$p(x_1, x_2, ..., x_T) = p(x_1)p(x_2|x_2)...p(x_T|x_1, ..., x_{T-1}). \tag{12}$$

The simple RNN model provides an output which is equivalent to:

$$h_t \sim p(x_t|x_1, ..., x_{t-1}) \tag{13}$$

Therefore, RNN is efficient in dealing with time-series data. However, due to the difficulties of training RNN for capturing long-term dependencies when gradients tend to vanish or explode, alternative units has been proposed. LSTM [22] and GRU [10] are the most popular ones, and are considered in this study.

LSTM unit is illustrated in Fig. 3b and is formulated by the following equations:

$$i_t = \sigma\left(x_t W_{xi} + h_{t-1} W_{hi} + c_{t-1} W_{ci} + b_i\right) \tag{14a}$$
$$f_t = \sigma\left(x_t W_{xf} + h_{t-1} W_{hf} + c_{t-1} W_{cf} + b_f\right) \tag{14b}$$
$$\widehat{c}_t = \tanh\left(x_t W_{xc} + h_{t-1} W_{hc} + b_c\right) \tag{14c}$$
$$c_t = f_t c_{t-1} + i_t \widehat{c}_t \tag{14d}$$
$$o_t = \sigma\left(x_t W_{xo} + h_{t-1} W_{hi} + c_t W_{ci} + b_o\right) \tag{14e}$$
$$h_t = o_t \tanh\left(c_t\right) \tag{14f}$$

By introducing input gate i_t, forget gate f_t and output gates o_t, the memory content c_t, and 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 Eq. (14f). 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 Eq.(14d). 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 [11]. GRU unit is depicted in Fig. 3c and is formulated by the following equations:

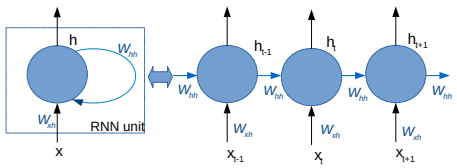
$$r_t = \sigma\left(x_t W_{xr} + h_{t-1} W_{hr} + b_r\right) \tag{15a}$$

$$u_t = \sigma\left(x_t W_{xu} + h_{t-1} W_{hu} + b_z\right) \tag{15b}$$

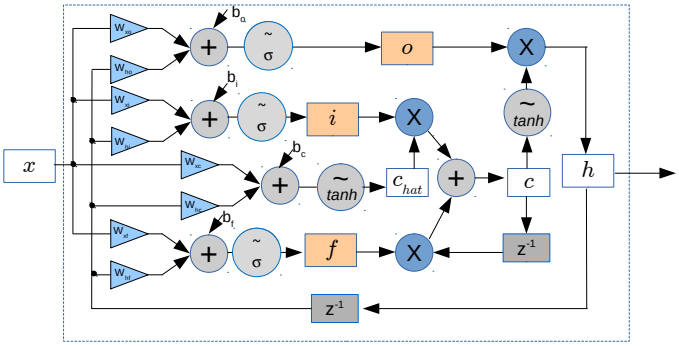
$$\widehat{h}_t = \sigma\left(x_t W_{xh} + r_t h_{t-1} W_{hh} + b_h\right) \tag{15c}$$

$$h_t = \left(1 - u_t\right) h_{t-1} + u_t \widehat{h}_t \tag{15d}$$

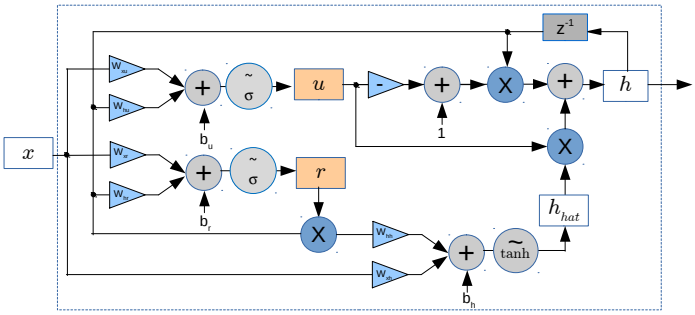
In a GRU unit, the update gate u_t adjusts the updates of its content h_t via Eq. (15d), whereas the reset gate r_t, similar to the forget gate in LSTM unit, decides to what extent the previous state h_{t-1} should be memorized in Eq. (15c).



(a) Simple RNN Unit



(b) LSTM Unit



(c) GRU Unit

Fig. 3 RNN Unit Structures

4 Recommendation systems for time-series predictions

The prediction methods may perform differently regarding the same time-series. Typically, for selecting a suitable predictor for a given time-series, 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. Hence, given a huge number of time-series and a very large number of prediction methods, finding the mechanism that optimally pins down the most efficient prediction algorithm for each time-series is a real hurdle. Hence, we propose to adopt a Recommendation System (RS) for dynamically selecting the most appropriate prediction methods.

In Section 4.1, we briefly present Imputation-Boosted Collaborative-Filtering based Recommending Prediction Method (IBCF-RPM), proposed in our previous work [36]. This recommendation method is designed for selecting the most appropriate predictor for small or medium size collection of time-series. In sections 4.2 and 4.3, we propose a new solution for recommending the best predictors and to aggregate them in order to operate on a huge collection of time-series. In section 5.5, we put forward a complexity comparison between these two proposals. We recall that, in our framework, the time-series represent the multimedia content profiles.

4.1 IBCF Recommending Prediction Method

In this section, we briefly introduce the recommendation method called IBCF-RPM proposed in [36], able to recommend an appropriate prediction method for a given profile, based on the estimation of prediction accuracy of similar contents.

For each time-series associated with a content in a catalog T, IBCF-RPM proposes to train and measure the prediction accuracy for a ratio of randomly chosen predictors during the test period. The accuracy of other predictors is estimated thanks to the CF technique. The method includes the following steps:

(i) For each time-series i from T, define P_i as a set of ratios of prediction methods, randomly selected from the initial set of predictors. For each profile i and each method m from P_i, compute the sparse accuracy matrix $ACC_{m,i}$. The non-estimated elements in the matrix correspond to unselected predictors.

(ii) For each couple of profiles u and v, a similarity matrix $sim\,(u, v)$ is computed by using the Pearson correlation coefficient. This matrix allows to identify for each time-series i, the most similar contents in the initial set, according to the prediction methods for which the accuracy was computed.

(iii) For each profile i, the set of the K most similar profiles, denoted $Top - K$, will contribute to estimate the previous non-estimated values $ACC_{m,i}$ in the accuracy matrix M_{ACC}, as thoroughly explained in [36].

(iv) For each profile i, the recommended prediction method will be the method having the highest estimated accuracy in the accuracy matrix M_{ACC} computed at the previous step.

4.2 Proposed Recommendation System Framework for Prediction Methods Selection

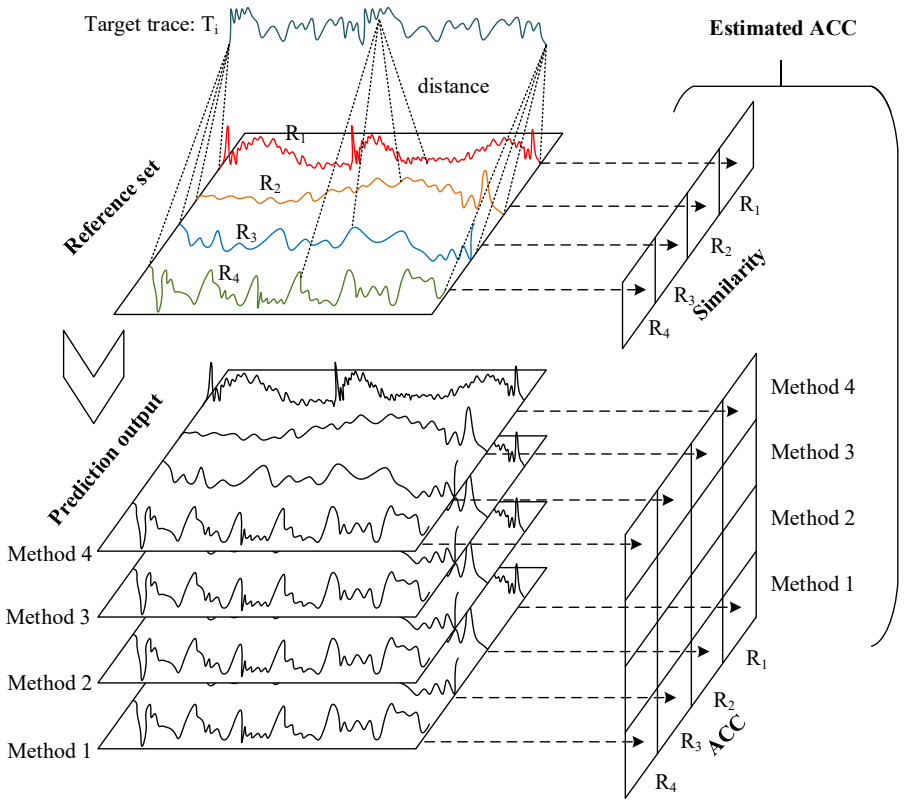


Fig. 4 Recommendation system framework for prediction method selection

Typically, some predictors could perform well when used for a given set of time-series. Determining appropriate predictors for each time-series in a large dataset like that of YouTube by investigating all available predictors is not a viable option. In this work, we propose a recommendation framework for facilitating this task. The process of the proposed framework is illustrated in Fig. 4 and have the following steps:

(i) Select a subset $\mathcal{R} = \{ R_j \}$ of reference contents profile having enough various profiles.
(ii) Investigate the predicting performances of all prediction methods on $\mathcal{R}$. Then, a reference accuracy matrix M_{ACC} relative to all methods applied to traces on $\mathcal{R}$ will be obtained.
(iii) For a new content profile T_i outside $\mathcal{R}$, compute the similarity vector V_{Sim} between T_i and reference profiles R_j in $\mathcal{R}$.

(iv) Through M_{ACC} and V_{Sim}, estimate the accuracy of all prediction methods applied to T_i. The estimated accuracy of a method m applied to T_i is calculated by what follows:

$$\widetilde{ACC}_{m,i} = \frac{\sum_{j \in \mathcal{K}} \frac{1}{d_{i,j}} ACC_{m,j}}{\sum_{j \in \mathcal{K}} \frac{1}{d_{i,j}}} \tag{16}$$

where $\mathcal{K}$ is the set of the k most similar time-series of T_i from the reference set $\mathcal{R}$, and $d_{i,j}$ is the similarity distance between T_i and R_j.

The proposed Recommendation System framework provides the estimated accuracy $\widetilde{ACC}_{m,i}$ of all prediction methods m on the profile T_i. In the next section, we explain how to select the best estimated predictors and aggregate them.

4.3 Proposed Multi-recommended Predictors Aggregation

The estimated accuracy of the investigated predictors able to estimate the solicitation evolution of a given content provide us with a descending accurate list of predictors. The best predictor or the top-N best ones could be selected for the content at hand. On the other hand, the distance that measures the similarity between a new content and the ones in the reference set impacts the value of estimated accuracy. The recommended predictors based on different distance measures could have different performances. Therefore, to enhance the proposed framework by having recourse to a diversity of recommendations, we propose to aggregate multi-recommended predictors based on the two following approaches:

- **Top-N Best Predictors Aggregation (Top-N BPA)**: The Recommendation System utilizes **only one suitable distance measure** for each given content. Accordingly, the top-N highest ACC predictors are selected. The predicting output of recommended predictors are aggregated by their mean predicted value.
- **Multi-Distances Best Predictors Aggregation (MD-BPA)**: **Multiple distance measures** are adopted to estimate the accuracy of predictors for a given content. The best predictor for each distance measure is chosen. Aggregating predicting outputs is also based on their predicted mean value.

5 Performance analysis

For evaluating the effectiveness of the proposed methods, Top-N BPA and MD-BPA, we perform the following simulations, based on the YouTube profiles dataset, presented in Section 2.1.

5.1 Context of simulation

For predicting daily access number to YouTube contents, we use the **slicing window prediction method**. The number of access to a YouTube video for the next day is predicted based on the historical access information extracted from a *look*

back window of W_{lb} preceding days. For example, with $W_{lb} = 3$, 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_{lb} is selected in the set of $\{1, 2, 3, 5, 7, 14, 30\}$ day(s).

From the huge dataset of Youtube, we selected 288 content profiles which were separated randomly into two parts: *88 profiles* for establishing the ***reference set*** and the rest (*200 contents*) for formulating the ***target set***. The ***target set*** is used for evaluating the effectiveness of the proposed RS framework.

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 for the RS that is able to recommend the most appropriate method for a content. The selected parameters given in Table 1 combined with the 7 above cases of W_{lb} give **28 tree-based predictors** and **84 RNN-based predictors**.

Table 1 Predictor parameters

RNN-based predictors			Tree-based regressors		
	No. of Unit			Max.	No. of
Config.	Layer 1	Layer 2		depth	Estimators
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			

5.2 Performance of Individual Predictors

We evaluate the performances of the individual predictors on the YouTube dataset presented in Section 2.1. For each content profile, we evaluate the performance of all investigated predicting methods by training, testing and validating on the *training*, *testing* and *validating periods*, respectively.

Accuracy The average accuracy on training, testing and validation periods of the investigated prediction methods, when $W_{lb} = 7$, are given in Fig. 5.

It can be seen that RNN-based predictors have better performances then Tree-based methods. Their accuracy at training, testing and validating periods are quite consistent while those of Tree-based predictors are not. Tree-based predictors are over-fitted when they have higher training accuracy but have worse testing and validating accuracy. For this reason, we should not perform the recommendation based on the training accuracy. Instead, resorting to *testing accuracy* is more appropriate.

History and configurations impact. Fig. 6 shows the average MAE and MSE of predictors applied on the testing content set. It can be seen that the trends of MAE and MSE are consistent. With the same W, LSTM-based and GRU-based predictors have lower errors than tree-based predictors. Among tree-based predictors, the Random Forest predictor is the best. Within the RNN-based predictors, the C3 configuration with only one layer is better than the C6 configuration

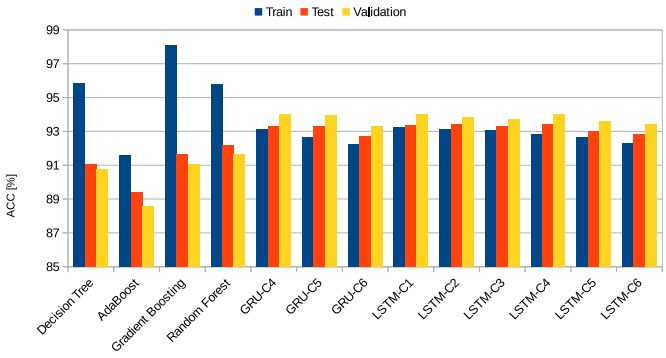


Fig. 5 Average prediction accuracy of individual predictors with $W_{lb} = 7$ days

with two-unit layers. With the same configuration and W, the predicting errors of LSTM-based and GRU-based predictors are almost similar. Decision Tree and Random Forest predictors, which are simpler, have increasing errors with larger W, whereas AdaBoost and Gradient Boosting show little difference when the error in $W = 30$ is smaller than the error in $W = 14$. For RNN-based predictor, the error in $W = 30$ is the smallest one.

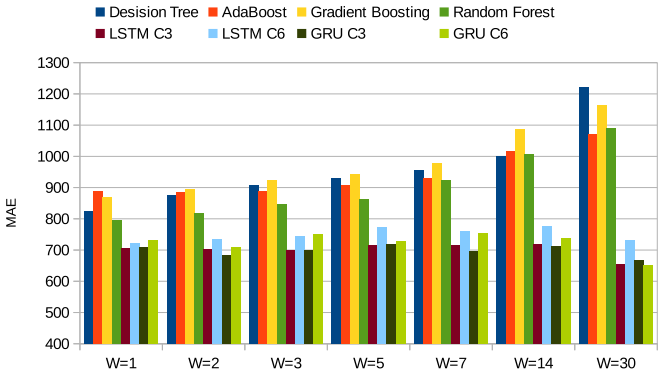


Fig. 6 Average testing errors of individual predictors

Simulation time. Fig. 7 depicts the simulation time for the selected predictors during *training and validation phases* for different values of the loop-back Window W_{lb}. The most time consuming phase is the training phase for both predictors categories. For Tree-based predictors, the simulation time of Decision Tree predictor is almost negligible (~ 0.1 s) and that of other methods is also very small (between ~ 8 and ~ 80 s). For RNN-based predictors, simulation time is an issue because it is very high compared to that of tree-based predictors: it is comprised between 600 and 1800 s. Indeed, the major time consumer is the training phase of the Neural Network, which obviously changes according to the number of recurrent units, compared to the validation time which is negligible. In our applicative con-

text, this aspect isn't a drawback because the training process is always an off-line one. The simulation time of GRU net is slightly larger than LSTM net though in principle GRU unit has less parameters than LSTM unit. This effect is related to the time-series profiles. We can also see that the general trend of the predictors simulation time is incremental with the value of W_{lb}. The increase in simulation time is the price paid for the higher accuracy that RNN-based predictors achieve compared to tree-based predictors.

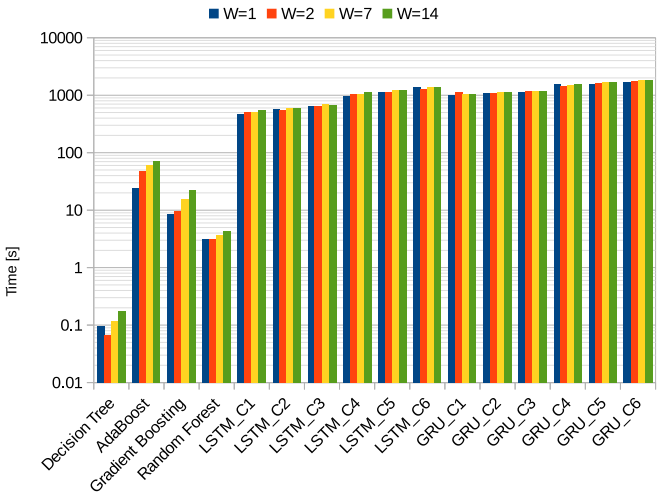


Fig. 7 Average execution time of individual predictors for different values of W_{lb}

5.3 Performance of Top-N BPA

In order to analyze the performance of the proposed recommendation framework based on the aggregation of top-N best predictors, we consider two popular distance measures including Euclidean and Dynamic Time Warping (DTW) distances. Since the original DTW requires heavy computation, we consider DTW-W with a "Sakoe-Chiba band" width that equals 30.

For each distance measure, we draw the average validating ACC of our proposed top-N best predictors aggregation according to several cases of top-N (the most accurate N methods) and top-K (the most similar K contents). The results are displayed in Fig. 8. To unravel the effect of the recommending mechanism, we investigate separately RNN-based predictors pool and Tree-based predictors pool because lower individual accuracy makes the Tree-based predictors less recommended if we consider both types of predictors (i.e., ACC $< 92\%$ and $< 94\%$ for Tree-based predictors and RNN-based predictors, respectively). There are different types of impact that can be seen here.

- *The impact of the distance measure*: The accuracy values given by Top-N BPA with both tested Euclidean and DTW-W distance measures are very similar, with an improvement of around 0.2% for Euclidean distance.
- *The impact of Top-N value*: i) For the RNN-based predictors, we can observe a general trend in which the accuracy increases along with the increase of the Top-N value. With Euclidean distance, increasing the number of aggregated predictors Top-N from 3 to 19 produces an increase of 0.2% in the estimated accuracy. With DTW-W distance, there is a little difference when Top-K = 5. However, the general increasing trend for this case is almost similar. Many good RNN-based predictors have small discrepancies in their performance. This is the main reason why the range for best predictor aggregation can be very high (up to 19 predictors) in the case of RNN-based predictors pool. ii) For tree-based predictors, the increase of Top-N value implies an increase followed by a decrease of the accuracy. The best prediction accuracy is obtained for the aggregation of $3-5$ predictors. Aggregating more than 5 predictors produces a decrease in the estimated accuracy. A lower number of good predictors with a higher gap among their accuracy is the reason behind the existence of a low optimal value of Top-N ($\sim$ 5 predictors).
- *The impact of Top-K value*: We can see in brief that Top-K value should be small (5 or 10) in the case of RNN-based predictors pool and should be high (15 or 20) in the case of Tree-based predictor pool. The reason for this effect is that RNN-based predictors are well-trained and have a higher sensitivity to the evolution of the content profiles; contrary to Tree-based predictors that are less sensitive to them because of the training over-fitting. Therefore, combining more suggestions from reference profiles in the case of Tree-based predictors provides more reliability and hence more accuracy.

5.4 Performance of MD-BPA

In this subsection, we investigate our second devised aggregation framework.

5.4.1 Distance measure analysis

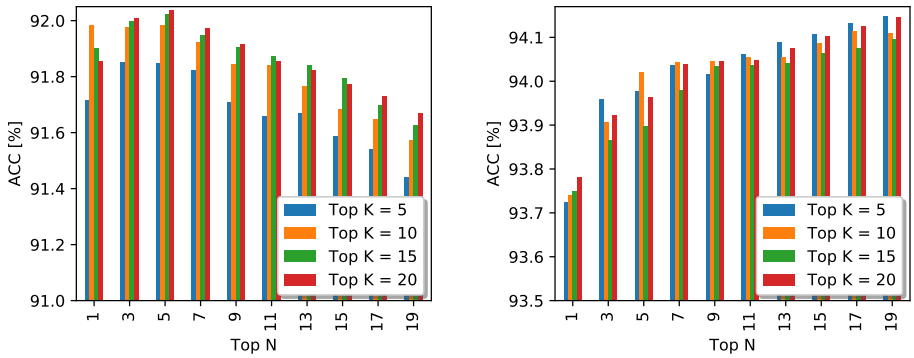
Similarity measures between target and reference profiles provide the output of the estimated ACC. Therefore, selecting a good distance measure is the key point here. To select the most appropriate measure, the analysis of different distance measures is needed.

Standardization impact for time-series values. For some distance measures such as EUCL, the computed distance values are very sensitive to the amplitude of time-series. Therefore, to remove the influence of noise and evolution trend of time-series, their value standardization is adopted. The standardization for a time-series $X = x_i$ is given by:

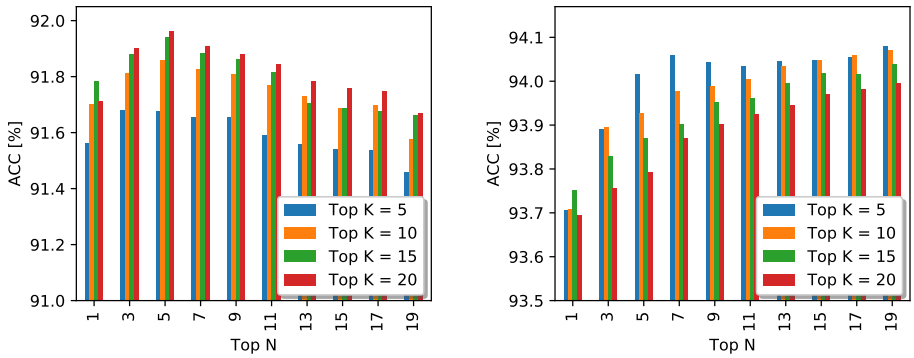
$$\hat{x}_i = \frac{x_i - \mu_X}{\sigma_X} \tag{17}$$

where μ_X and σ_X are the mean and standard deviation of X, respectively.

We consider the impact of standardization on the computed accuracy in the proposed recommendation framework. The average accuracy of several distance

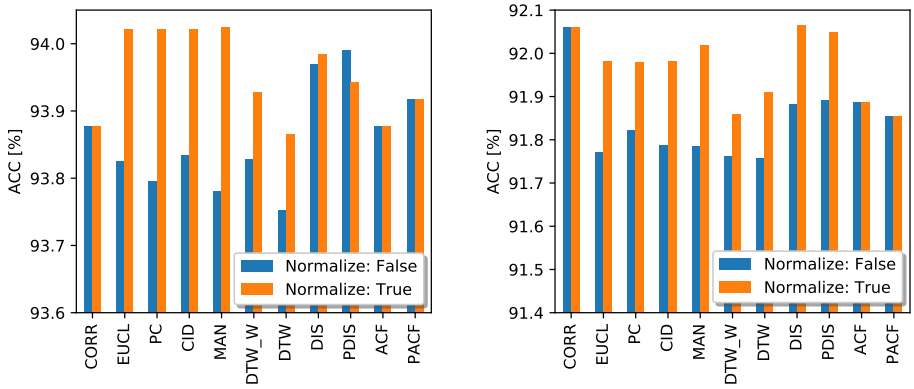


(a) Tree-based with EUCL (b) RNN-based with EUCL



(c) Tree-based with DTW-W (d) RNN-based with DTW-W

Fig. 8 Accuracy vs. Top-N predictors aggregation



(a) RNN-based predictors (b) Tree-based predictors

Fig. 9 Impact of time-series values standardization

measures in the original and standardized cases of the content profiles are given in Fig.9. Both RNN-based and Tree-based sets of individual predictors are investigated. We select Top-N=5 and Top-K=10 which are appropriate values according to the results given in Section 5.3.

We can see that for both predictors set, the standardization improves the performance of our proposed framework for all distance measures except d_{CORR}, d_{ACF} and d_{PACF}. The reason is that they are feature-based distances and standardization does not influence the feature extraction (i.e., the covariance feature for d_{CORR}, and the autocorrelation feature for d_{ACF} and d_{PACF}). Therefore, the accuracy values remain unchanged.

Distance measure correlation. As mentioned in section 2.3, there are numerous types of distance measures that could be found in the literature. The idea here is to form a set of distances so that the diversity of prediction methods is taken into account. Accordingly, the distance within the selected set should not be strongly correlated.

In order to evaluate the correlation between distance measures, the Mantel test [32] is used. The latter enables us to measure the correlation between distance matrices. We compute the distance matrix $D^d_{\mathcal{R}} = \{D^d_{i,j}\}$ of time-series in the reference subset $\mathcal{R}$ for each kind of distance measures d. Then, correlation test statistic is $r = \sum_{i<j} \sum D^d_{i,j} D^{d'}_{i,j}$ and per mutational correlation test statistic is $\{\tilde{r}\} = \left\{\sum_{i<j} \sum \tilde{D}^d_{i,j} D^{d'}_{i,j}\right\}$, where $\tilde{D}^d_{\mathcal{R}}$ are random permutation matrices of $D^d_{\mathcal{R}} = \{D^d_{i,j}\}$ that are used to derive the Mantel test score.

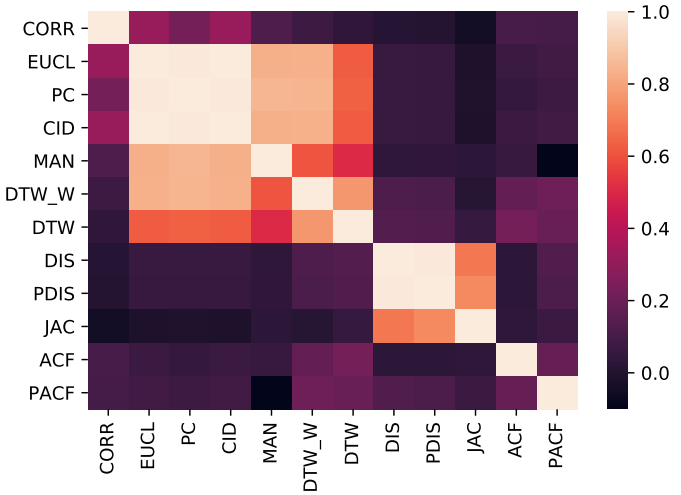


Fig. 10 Correlation Matrix of Distances

Fig. 10 shows the correlation score of the investigated distances. It can be seen that there are two highly correlated groups of distance measures including $\{d_{EUCL}, d_{PC}, d_{CID}, d_{MAN}, d_{DTW}, d_{DTW-W}\}$ and $\{d_{DIS}, d_{PDIS}, d_{JAC}\}$, the other ones $\{d_{CORR}\}$, $\{d_{ACF}\}$, $\{d_{PACF}\}$ are not correlated.

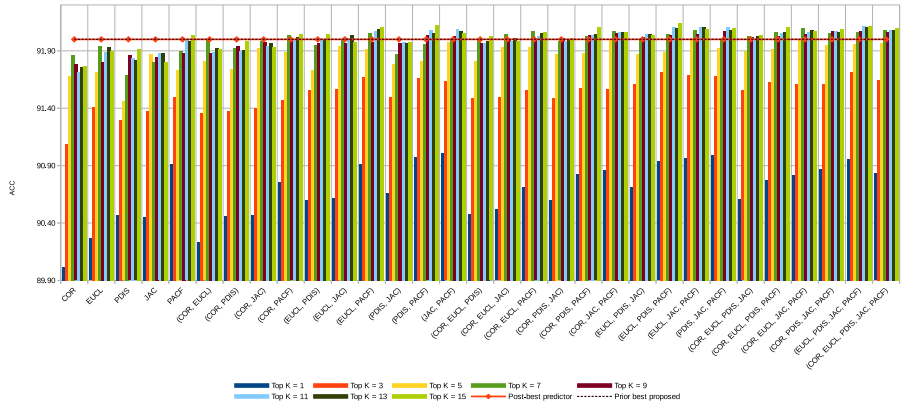


Fig. 11 Accuracy vs. Distance combination with Tree-based predictors

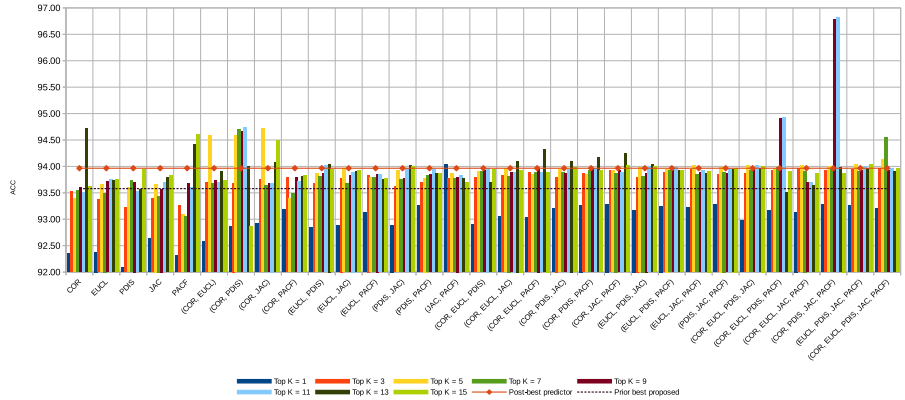


Fig. 12 Accuracy vs. Distance combination with RNN-based predictors

5.4.2 MD-BPA performance

According to the result of distance measures correlation given in the previous subsection 5.4.1, we select five representative distance measures d_{CORR}, d_{EUCL}, d_{PDIS}, d_{JAC}, and d_{PACF} for investigating the effectiveness of different combinations of multiple distance types. The other ones are ignored due to their correlation with the selected distance measures.

Fig. 11 and Fig. 12 show the accuracy of the proposed MD-BPA with tree-based predictors pool and with RNN-based predictors pool, respectively. Combination of 2, 3 and 4 different distances are compared with individual distance measures, while the number of similar contents varies between 1 and 15. The aggregation performances are compared with those of the optimal predictor in term of prediction accuracy. In the case of Tree-based predictors, the post-best predictor improves the accuracy of the prior-best one by 0.1%; while in the case of the RNN-based predictors, the improvement is of 0.5%. For both predictor types, combination of at least 2 different distances improves the prediction accuracy when $K \geq 5$. For Tree-based predictors, the PACF distance improves the accuracy in aggregates of 2, 3 or 4 dis-

tances. When $K \geq 5$, these combinations realize performances that are very close or even better than those of the post-best predictor. For RNN-based predictors, when $K \geq 3$, the PDIS distance improves the accuracy of aggregates close to the accuracy of the post-best predictor. The same distance combinations increase the aggregates accuracy from 0.5% to 2.8%, e.g. (d_{CORR}, d_{EUCL}), (d_{CORR}, d_{PDIS}), (d_{CORR}, d_{JAC}), (d_{CORR}, d_{EUCL}, d_{PACF}), (d_{CORR}, d_{EUCL}, d_{PDIS}, d_{PACF}), (d_{CORR}, d_{PDIS}, d_{JAC}, d_{PACF}), etc.

Three different effects of the adopted parameters are observed as follows.

- *The effect of Top-K*: For a certain set of distance measures, the accuracy of MD-BPA increases along with the value of Top-K. However, the estimated accuracy value varies insignificantly when Top-K is larger than 5 in both predictor pools.
- *The effect of different sets of distance measures*: For the same value of Top-K, the distance measures sets which include EUCL and/or PACF in tree-based predictors pool and include PACF in RNN-based predictors pool, tend to perform better than the others, which means that these distance measures are more suitable for these kinds of predictors.
- *The effect of the number of distance measures*: There is an obvious increment of the MD-BPA accuracy with the increase of the number of combined distance measures. With approximately 2 distance measures used by at most three predictors for a content profile, we can achieve the accuracy of the optimal best predictor (i.e., the best predictor when the future is known, and indicated by "post-best predictor") on both predictor pools. It should be also noticed that, in both cases of predictor types, the MD-BPA achieves higher accuracy than the best recommended predictor (i.e., the best recommended predictor based only on historical performances is indicated by "prior-best proposed").

5.5 Complexity comparison

In this subsection, we analyze the complexity of the proposed aggregations Top-N BPA and MD-BPA, and we compare them with the complexity of IBCF-RPM. For all of the above methods, there are three computational costs:

- Training predictors
- Computing similarities
- Aggregating several predictors

First, we have the complexity of training predictors, which varies depending on their nature. If we consider the same set of predictors, *the computational cost for the training process is the same*. So, we will not consider it for the present comparison. Lastly, we have the *computational cost of the aggregation process* which is the same for Top-N BPA and MD-BPA for each content, and is equivalent to the cost of collaborative filtering in IBCF-RPM. However, this cost is dismissible compared to that of similarities computation. In other words, the complexity of the investigated methods is mainly due to the *similarity computational cost*.

Assume that M methods are considered to estimate the popularity of V contents, and each content consists of the same historical data of the p previous days. For IBCF-RPM, the cost for training predictors depends on the running ratio $r, 0 < r < 1$. rM methods are trained for all of V contents. Whereas, for proposed MD-BPA and Top-N BPA, $r'V, 0 < r' < 1$ contents are selected as the reference

set and trained with all considered methods M. Therefore, in order to identify the training costs of the considered approaches, the running must be identical with the selecting ratio r'. Given that, the number of training predictors for IBCF-RPM, Top-N BPA and MD-BPA are now identically given by rVM.

With those assumptions, the comparison of the complexity of IBCF-RPM, Top-N BPA and MD-BPA reduces to the comparison of the similarity computation costs. For performing collaborative filtering in IBCF-RPM, one needs to compute the similarity of all pairs of contents in the set of V contents on a vector of M predictors' accuracies. Thus, the complexity for computing the similarity of IBCF-RPM is $\mathcal{O}\left(V^2M\right)$. For Top-N BPA, every content in the predicting set $(1-r)V$ is compared with all contents in the reference set rV for computing similarities. Therefore, the complexity for computing the similarity of Top-N BPA is $\mathcal{O}\left(r(1-r)V^2p\right)$. For MD-BPA, a similar process to that of Top-N BPA is needed for computing the similarities for each distance type. Hence, for m selected distance types, the complexity for computing the similarity of MD-BPA is $\mathcal{O}\left(mr(1-r)V^2p\right)$.

The number of methods, M, could span from hundred to thousand which is also the range of p, i.e., some months to years, whereas $r < 1$. Therefore, the proposed Top-N BPA and MD-BPA are simpler than IBCF-RPM. For example, with $p = M$, $m = 3$ and the running ratio $r = 0.3$, the complexity of Top-N BPA and MD-BPA is now equal to 21% and 63% of the IBCF-RPM complexity, respectively.

6 Summary and Future Work

In summary, we have addressed, in this paper, the crucial issue of being able to predict with large accuracy the popularity of multimedia contents in order to ensure a proactive caching in 5G MEC, close to the end-users. As daily requests evolution of multimedia content is represented as time-series data, time-series prediction was focused on. We proposed a recommendation framework able to estimate the accuracy of a high number of prediction methods applied to a large set of time-series. The proposed framework is a general one which can be used in various applications modeled by time-series. It also enjoys very good scalability, as it adapts dynamically to analyze huge sets of time-series and large sets of predictors.

Real traces of the popularity evolution of YouTube videos were considered as the study case. From the entire catalog, we defined a reference set of time-series to which the prediction accuracy of all investigated predictors were stored. The accuracy of each predictor given to any content outside the reference set will be estimated based on the stored accuracy of the most similar subset of the reference set. The most similar reference time-series with the new content is selected according to the carefully chosen distance measures. The framework recommends the predictors having the best estimated accuracy.

Our finding is that the individual RNN-based predictors have always better average performances than tree-based predictors during the validation phase for an equivalent prediction time. The maximum performance gain is around 5% in the context of our simulation. However, the training time of RNN-based predictors is about 10 times higher than that of tree-based predictors. The complexity issue can be alleviated by performing training offline. The main concern of predictors

accuracy, which is closely related to time-series characteristics, has been considered by the proposed recommendation framework. It enables us both to select more appropriate predictors for each time-series and to aggregate them to improve the prediction performances. Two predictor aggregation methods, called Top-N BPA and MD-BPA, have been proposed.

For a given time-series, Top-N BPA aggregates the N highest accurate predictors, identified by the most K similar contents based on one distance measure, whereas MD-BPA aggregates the M most accurate predictors determined by M considered distance measures. Simulation results have shown the effectiveness of the aggregation method in term of accuracy improvement. We also showed that the complexity of the proposed recommendation methods is drastically reduced compared to our previous work deemed IBCF-RPM. In detail, the complexity is reduced by 79% and 37% for Top-N BPA and MD-BPA, respectively. This makes these recommendation frameworks appropriate for prediction problems involving a huge number of time-series and a large number of predictors.

For future work, we need to assess the effectiveness of the proposed recommendation system for content caching in comparison with classical caching algorithms. The generality of the proposed recommendation framework and its performances should be investigated in diverse contexts where data is represented as time-series.

References

1. Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T.: Mobile edge computing: A survey. IEEE Internet of Things Journal **PP**(99), 1–1 (2017). DOI 10.1109/JIOT.2017.2750180
2. Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S.: Youtube-8m: A large-scale video classification benchmark. arXiv preprint arXiv:1609.08675 (2016)
3. Al-Falahy, N., Alani, O.Y.: Technologies for 5g networks: Challenges and opportunities. IT Professional **19**(1), 12–20 (2017). DOI 10.1109/MITP.2017.9
4. Batista, G.E., Wang, X., Keogh, E.J.: A complexity-invariant distance measure for time series. In: Proceedings of the 2011 SIAM international conference on data mining, pp. 699–710. SIAM (2011)
5. Breiman, L.: Using adaptive bagging to debias regressions. Tech. rep., Technical Report 547, Statistics Dept. UCB (1999)
6. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and regression trees. CRC press (1984)
7. Bühlmann, P., Hothorn, T.: Boosting Algorithms: Regularization, Prediction and Model Fitting. Statistical Science **22**(4), 477–505 (2007). DOI 10.1214/07-STS242. URL http://projecteuclid.org/euclid.ss/1207580163
8. Cerwall, P., Möller, R., Jonsson, P., Carson, S., Svenningsson, R., Lindberg, P., Öhman, K., Sandin, T., Rangel, L., Sorlie, I., Elmgren, S., Karapantelakis, A., Wieweg, L., Halen, M., Edstam, J., Queiros, R., Muller, F., Englund, L., Kirby, R.: Ericsson Mobility Report. White paper (2018). URL https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf
9. Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y.: Recurrent neural networks for multivariate time series with missing values. arXiv preprint arXiv:1606.01865 (2016)
10. Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014)
11. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. CoRR pp. 1–9 (2014). URL http://arxiv.org/abs/1412.3555
12. Conflitti, C., Mol, C.D., Giannone, D.: Optimal combination of survey forecasts. International Journal of Forecasting **31**(4), 1096–1103 (2015). DOI https://doi.org/10.1016/j.ijforecast.2015.03.009. URL http://www.sciencedirect.com/science/article/pii/S0169207015000606

13. Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence **14**(771-780), 1612 (1999)
14. Freund, Y., Schapire, R.E.: A desicion-theoretic generalization of on-line learning and an application to boosting. In: European conference on computational learning theory, pp. 23–37. Springer (1995)
15. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
16. Friedman, J.H.: Stochastic gradient boosting. Computational Statistics & Data Analysis **38**(4), 367–378 (2002)
17. Gardner Jr, E.S., McKenzie, E.: Forecasting trends in time series. Management Science **31**(10), 1237–1246 (1985)
18. Hassine, N.B., Marinca, D., Minet, P., Barth, D.: Popularity prediction in content delivery networks. In: Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symposium on, pp. 2083–2088. IEEE (2015)
19. Hassine, N.B., Milocco, R., Minet, P.: Arma based popularity prediction for caching in content delivery networks. In: Wireless Days, 2017, pp. 113–120. IEEE (2017)
20. Hill, T., O'Connor, M., Remus, W.: Neural network models for time series forecasts. Management science **42**(7), 1082–1092 (1996)
21. Ho, T.K.: Random decision forests. In: Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on, vol. 1, pp. 278–282. IEEE (1995)
22. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
23. Immink, K.A.S., Weber, J.H.: Minimum pearson distance detection for multilevel channels with gain and/or offset mismatch. IEEE Transactions on Information Theory **60**(10), 5966–5974 (2014)
24. networking Index, C.V.: Forecast and methodology, 2016-2021, white paper. San Jose, CA, USA **1** (2016)
25. Isinkaye, F.O., Folajimi, Y.O., Ojokoh, B.A.: Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal **16**(3), 261–273 (2015). DOI https://doi.org/10.1016/j.eij.2015.06.005. URL http://www.sciencedirect.com/science/article/pii/S1110866515000341
26. Kass, G.V.: An exploratory technique for investigating large quantities of categorical data. Applied statistics pp. 119–127 (1980)
27. Keogh, E., Ratanamahatana, C.A.: Exact indexing of dynamic time warping. Knowledge and information systems **7**(3), 358–386 (2005)
28. Laptev, N., Yosinski, J., Li, L.E., Smyl, S.: Time-series extreme event forecasting with neural networks at uber. In: International Conference on Machine Learning (2017)
29. Levandowsky, M., Winter, D.: Distance between sets. Nature **234**(5323), 34 (1971)
30. Lin, T., Guo, T., Aberer, K.: Hybrid neural networks over time series for trend forecasting. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17 (2017). DOI 10.24963/ijcai.2017/316. URL https://doi.org/10.24963/ijcai.2017/316
31. Lipton, Z.C., Berkowitz, J., Elkan, C.: A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019 (2015)
32. Mantel, N.: The detection of disease clustering and a generalized regression approach. Cancer Research **27**(2 Part 1), 209–220 (1967)
33. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A Survey on Mobile Edge Computing: The Communication Perspective. IEEE Communications Surveys Tutorials **19**(4), 2322–2358 (2017). DOI 10.1109/COMST.2017.2745201
34. Montero, P., Vilar, J.A., et al.: Tsclust: An r package for time series clustering. Journal of Statistical Software **62**(1), 1–43 (2014)
35. Mulder, W.D., Bethard, S., Moens, M.F.: A survey on the application of recurrent neural networks to statistical language modeling. Computer Speech & Language **30**(1), 61–98 (2015). DOI https://doi.org/10.1016/j.csl.2014.09.005. URL http://www.sciencedirect.com/science/article/pii/S088523081400093X
36. Nguyen-Thanh, N., Marinca, D., Khawam, K., Martin, Boukhatem, L.: Multimedia content popularity: Learning and recommending a prediction method. In: IEEE Global Communications Conference (Globecom). IEEE (2018)
37. Nguyen-Thanh, N., Marinca, D., Khawam, K., Martin, S., Boukhatem, L.: Multimedia content popularity: Learning and recommending a prediction method. In: Proceedings of IEEE Global Communications Conference 2018, pp. –. IEEE (2018)

38. Quinlan, J.R.: Induction of decision trees. Machine learning **1**(1), 81–106 (1986)
39. Quinlan, J.R.: C4. 5: programs for machine learning. Elsevier (2014)
40. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing **26**(1), 43–49 (1978). DOI 10.1109/TASSP.1978.1163055
41. Székely, G.J., Rizzo, M.L., Bakirov, N.K.: Measuring and testing dependence by correlation of distances. The annals of statistics pp. 2769–2794 (2007)
42. Tatar, A., de Amorim, M.D., Fdida, S., Antoniadis, P.: A survey on predicting the popularity of web content. Journal of Internet Services and Applications **5**(1), 1–20 (2014). DOI 10.1186/s13174-014-0008-y
43. Taylor, J.W., Snyder, R.D.: Forecasting intraday time series with multiple seasonal cycles using parsimonious seasonal exponential smoothing. Omega **40**(6), 748–757 (2012). DOI 10.1016/j.omega.2010.03.004. URL http://dx.doi.org/10.1016/j.omega.2010.03.004
44. Zeydan, E., Bastug, E., Bennis, M., Kader, M.A., Karatepe, I.A., Er, A.S., Debbah, M.: Big data caching for networking: Moving from cloud to edge. IEEE Communications Magazine **54**(9), 36–42 (2016). DOI 10.1109/MCOM.2016.7565185
45. Zhang, G.P.: Time series forecasting using a hybrid arima and neural network model. Neurocomputing **50**, 159–175 (2003)