Proactive Video Caching Towards User Preference Using Ensemble of Deep Learning Models in Mobile Edge Networks

The-Vi Nguyen, Nhu-Ngoc Dao, Van-Dat Tuong, Wonjong Noh and Sungrae Cho

***Abstract*—Content caching is an attractive approach in edge computing with the rapid growth of mobile data, especially movie data. Due to the capacity limit of caches, it is essential to predict the popularity of movies and cache those popular ones. In this paper, we focus on predicting the popularity of movie genres. The proposed caching scheme includes two main parts: 1) adopting deep learning model Long Short Term Memory (LSTM) to predict the popularity of movie genres on different groups of attributes of users, which are based on users' profiles, such as gender, age. Then those predictions are combined using Ensemble Learning to obtain the final prediction. We aim to predict movie genre popularity incorporating users' behavior, which contributes to the high accuracy of the LSTM models. 2) Caching those movie genres with high predicted popularity. Finally, the simulation results show that our caching scheme outperforms baseline schemes in terms of a cache hit efficiency.**

***Index Terms*—Mobile Edge Network, Proactive Caching, Popularity Prediction, Ensemble Learning**

I. Introduction

NOWADAYS, with the widespread of smart devices such as smart phone, laptop, and smart TV, we have been experiencing various entertainment applications, game shows, and movies everyday. Those activities result in huge amount of data traffic, which puts growing pressure on the core networks and backhaul links. As a result, the traffic congestion at the backhaul links and the long latency between users and remote cloud servers, which both negatively affect the user's quality of experience. To address such challenges, Mobile Edge Computing (MEC) emerge as a new paradigm to provide computing and storage capacities on the network edge, where the cloud server is in close proximity to the users [1]. With the deployment of MEC, caching at the edge is an efficient way to alleviate mobile traffic and reduce the content delivery latency [1]. It means that cache entities at the edge nodes store popular contents, which are served directly to the users without repeatedly fetching from the remote content server. In the caching phase, due to limited capacity of cache entities, it is beneficial to know popularity of contents prior and cache the most popular ones. However, in this paper, we restrict our concentration on prediction popularity of *content's type* instead, which is *movie's genre* for an instance (please see Table. I).

V. T. Nguyen, N. N. Dao, D. V. Tuong and S. Cho are with the School of Computer Science and Engineering, Chung-Ang University, Seoul 156-756, South Korea (e-mail: tvnguyen, dnngoc, vdtuong@uclab.re.kr, srcho@cau.ac.kr).

In the literature, Zipf-like distribution was used to describe the content popularity distribution [2], [3], [4], which was based on the observation that only small percentage of ranked movies are requested from the majority of users. However, using the priori knowledge of the content popularity (rank of the content) in this way is unreasonable, since in practical scenarios, the content popularity is actually unknown and change across time. For example, the authors in [5], [6] indicated that popularity of videos on demand is a function of time. On the other hand, local content popularity at the caching entities may be different from the global content popularity across entire population of users (e.g., Zipf-like distribution). Because different contents can be favored by different users and their preferences may change according to internal factors associated with users' demographic information such as age, gender, occupation [8]. Thereby, the diversity of content preferences of local users in the vicinity of the caching entities should be taken into account in caching problem. From these changing in time and user's preference on content popularity, it is difficult to measure the content popularity via a distribution or a simple prediction.

Motivated from these discussions on popularity of general content, our study focuses on a specific example that is *movie*. We propose a caching scheme that concerns *time-varying characteristics* of *popularity of movie genres* and its dependence on diversity of local users' preference.

To address the first factor, we use the Long Short Term Memory network (LSTM) to learn and predict the popularity of movies. Since LSTMs recently provide a potential and powerful tool in modeling sequence data in the long term. Furthermore, the powerful data-driven self-adaptability and model generalizability enable LSTMs to uncover complex relationships among samples and perform predictions on new observations of a population effectively.

To address the second factor, we partition users population into different local groups based on their gender and age. Since with a group of users of the same gender or age, they tend to have similar tastes of movie genres. We can see that such kind of user groups exist in practical scenarios. For instance, male and female students in two dormitories of a university. Since almost of them share the same range of age, i.e., from 18 to 25 years old. Hence, the movie-genre preferences depends on gender rather than age. On the other hand, the age feature has more effect than gender in the case that children in an apartment are interested in animation or children movies, while elderly people in a nursing home

are interested in drama or romance movie. In the Fig. 1 illustrates the distribution of genres for male and female users, where the male group prefers action movie rather than drama movie, while the female group favors drama rather than action. And one more fact is that male users watch movies more than female users. Therefore, based on this observation, we localize user population into different groups related to their gender or age, then we build LSTM models on time series of number of requests for movie genres generated by those groups. Afterward, those LSTM models are combined using an ensemble method to obtain the average results.

In the previous studies [22], [23], they suggested that removing seasonality helps neural network models to produce better forecasts. One of the versatile and robust time-series decomposition is the Seasonal and Trend decomposition using Loess (STL), as proposed by [14], which is able to decompose time series into seasonal, trend and residual component. Therefore, in this paper, we combine time series decomposition method STL and LSTM model in a way that leverage their advantages to enhance the forecasting accuracy [16]. Here, we want to exploit the characteristic of individual time series subjected to each user groups such as seasonality, while the LSTM model allows learn across time series and non-linear trends.

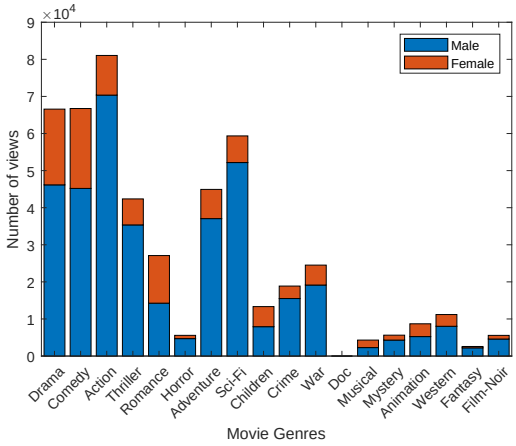

Fig. 1. Distribution of movie genres for male and female group in the MovieLens dataset from [21].

The main goal of this paper is to maximizing the cache hits of contents by storing popular video contents at Base Stations (BSs) with the help of a deep learning-based content popularity prediction scheme. Our contributions are summarized as follows:

- We propose a caching strategy to maximize cache hits in the future by controlling cache decision in current time slot. The formulation requires the knowledge of future popularity of every movie. Therefore, we propose a movie-genre-popularity-based caching scheme to cache those common movies.
- To support the proposed caching process, we need to predict the movie-genre popularity. We first point out that the user's demographic information, such as age and gender has huge effect on the movie taste of users (i.e., preference on movie genres of users). Based on this consideration, instead of employing only one baseline LSTM model without using user-profile information, we develop LSTM-based prediction models on different groups of users (based on different demographic information) and then combining these models using ensemble learning. We call this Ensemble-LSTM based model.
- In this paper, we propose an approach that incorporates LSTM models and classical time series models. Here, for each user group, an LSTM model trained on time series of 18 movies genres can extract information across these time series (global effects), while a classical model describes characteristics (seasonal, trend) of the individual time series (local effects).
- Through simulation results, we indicate that the leveraging user demographic information can be improve the performance of the LSTM model.

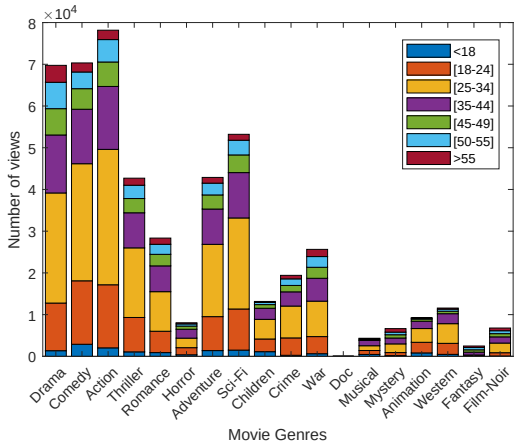

Fig. 2. Movie genre distribution on age groups in the MovieLens dataset from [21].

The rest of the paper is organized as follows. Section II reviews related works on the proactive caching for edge networks. The model system and caching optimization problem is provided in Section III. In Section IV, the LSTM-based model for forecasting popularity of movie's genre is provided. The detailed of the forecasting process is described in section V. Numerical results are illustrated in Section VI.

II. Related Work

A. Learning-based Proactive Caching

Under the consideration of unknown popularity, Bharath *et al.* [9] presented a learning-based method to have good estimate of content popularity in a required training time. Then in the improved paper [10], they additionally considered that the content popularity was not only unknown a priori, but also non-stationary and statistically dependent across time. Other studies such as Yang *et al.* [12], to improve the accuracy of the popularity prediction, the authors investigated content popularity with the location awareness, such as residential or business area. Relying on that investigation, the authors modeled of content caching and proposed two online learning algorithms that are adaptive to the change of users' demands in different

locations. Whereas Doan *et al.* [11] proposed a learning-based proactive caching with the content awareness. Here, two types of the video were considered, unpublished video (newly uploaded or new videos) and published video (old videos). In [9], [12], [11], once content popularity was estimated, it was used to determine an optimal caching placement that maximize the cache hit rate. Muller *et al.* [8] modeled the caching problem as a contextual multi-armed bandit problem that takes into account the user context information such as gender, age, occupation.

B. Time Series Forecasting and Deep-learning-based Proactive Caching

The popularity prediction problem can be formulated as a time series forecasting problem, however, there were few works that consider integrating forecasting popularity in caching. In the literature, common forecasting methods such as Autoregressive Moving Average (ARMA) [13], Autoregressive Integrated Moving Average (ARIMA) [], and Vector Auto-Regression (VAR) [] were developed for content popularity forecasting. However, for long-term temporal patterns, these models are inevitably prone to overfitting and high computational cost, especially for high-dimensional inputs. Recently, Long Short-Term Memory neural networks (LSTMs) provide us potential and powerful tool in modeling sequence data in the long term. Furthermore, the powerful data-driven self-adaptability and model generalizability enable LSTMs to uncover complex relationships among samples and perform predictions on new observations of a population effectively.

TABLE I
TABLE OF GENDER, AGE AND MOVIE'S GENRES, REFERENCED FROM MOVIELENS DATASET [21]

Gender	Age	Movie's genre
Female	under 18	Drama, Comedy, Action,
Male	[18, 24]	Thriller, Romance, Horror
	[25, 34]	Adventure, Sci-Fi, Children,
	[35, 44]	Crime, War, Documentary,
	[45, 49]	Musical, Mystery, Animation,
	[50, 55]	Western, Fantasy,
	over 56	Film-Noir

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we present the system model and formulate the caching problem in mobile edge networks.

A. Network Model

MEC edge servers provide computing and storage resources which can be utilized as edge nodes to store popular requested contents by users in their proximity. In our network scenarios, as illustrated in Fig. 3, a base station (BS) is deployed as an edge node with an adequately computing and storage capacity for the edge caching services. In the coverage of BS, a set of users can request directly movies of interest from the BS if they are already stored, otherwise they have to be taken from the remote cloud server through backhaul links.

B. Demographic-based User Partition

In the coverage of the BS, users with different interests on movie's genres may request different movies. According to the aforementioned discussion, users with the same age or gender seem to share the same interest or behaviours on their movie's genres they watch daily. Therefore the user population is partitioned into groups based on age and gender. In fact, the partition can be interpreted in real scenarios. For example, male and female students live in separate dormitories (gender group), children in apartments, elderly people in nursing homes (age group). Furthermore, this approach bring an advantage to our model. Since each group of users can share common watching behaviors, for example, watching time behaviour involving days in a week, hours in a day. Thus our model can capture these patterns better and produce more accurate forecast. According to different interests in movie's genres across user groups which change over time, our model forecasts which genres will be popular in the next day. Then these forecast result can be incorporated with the caching policy to obtain the maximum cache hit rate over a finite time horizon.

Let N be the number of users, which are partitioned into different groups based on their profile features, i.e., gender and age. As shown in Table I (referenced from MovieLens Dataset [21]), gender feature includes two sub-features which are female and male. Accordingly, two groups of users are indexed as i for $i \in \mathcal{I} = \{1, 2\}$. Age feature consists of seven sub-features according to user's range of age, whose respective groups are indexed as j, for $j \in \mathcal{J} = \{3, ..., 9\}$. Let $\mathcal{H} = \mathcal{I} \cup \mathcal{J} = \{1, 2, ..., 9\}$ be the set of indices of all user groups related to gender and age features. Movie genres are specified in the last column of the Table I. There are in total 18 movie genres, which are indexed as $k \in \mathcal{K} = \{1, ..., 18\}$.

C. Problem Formulation

For formulating the caching placement problem, without loss of generality, we assume that all movies belonging to the movie library $\mathcal{F} = \{1, 2, ..., F\}$ in the cloud server have the same size and the size is normalized to 1. The cache entity at the BS have limited storage, which can cache up to α movies. The caching entity at BS monitors user's requests within a set of time periods $\mathcal{T} = \{1, 2, ..., T\}$, where T is the finite time horizon. The information of a user request is recorded including a tuple of movie ID, movie's genre, and requesting time slot. Particularly, the tuple can be expressed as $R_f^{(t)} = \langle f, k_f, t \rangle$, $\forall f \in \mathcal{F}$, $k_f \subset \mathcal{K}$, $t \in \mathcal{T}$, where f is requested movie, k_f denotes genres of the movie f (note that a movie can belong to multiple genres hence), and t is requesting time slot, $\mathcal{K}$ refers the set of the movie's genre.

Let $c_f^{(t)} \in \{0, 1\}$ be the cache decision variable of movie f during time slot t, where $c_f^{(t)} = 1$ if f is stored locally, and $c_f^{(t)} = 0$ otherwise. In case the movie f has been stored in the cache entity of BS, then a cache hit occurs. Otherwise, a cache miss occurs, this movie has to be taken from the movie's cloud server. A caching decision can be made to determine whether movie f should be stored at the local cache

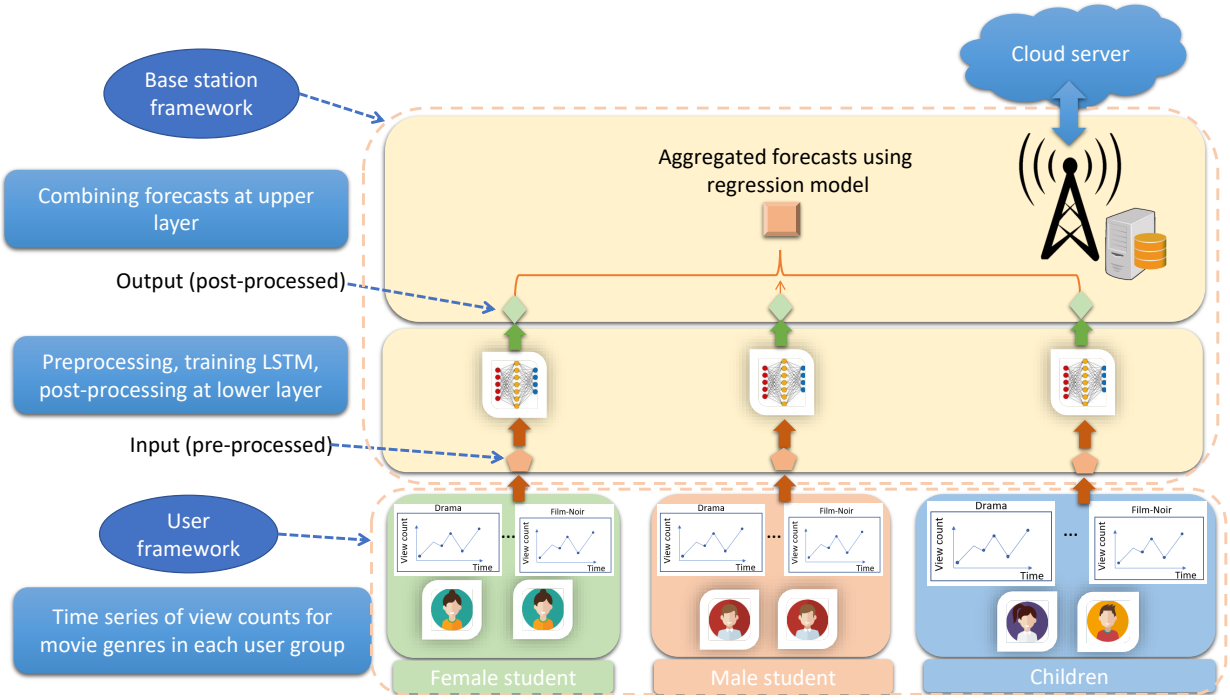

Fig. 3. System model and learning process.

and which existing movies in the cache should be removed from the cache storage. Due to the limited cache capacity, it is necessary to replace movie that have highest popularity and remove smallest popularity ones. To archive an optimal caching policy ζ, prediction the popularity of the movie based on preference on movie genre of users can be a potential approach. To measure the caching performance, we use the cache hit rate defined as the number of cache hits over the number of requests up to horizon time T as follows

$$\mathcal{R}_\zeta = \frac{\sum_{t=1}^{T} c_f^{(t)}}{\sum_{t=1}^{T} N_t},$$

where N_t is number of requests at time t. Our goal is to find the optimal policy to maximize the cache hits rate up to time slot T, which is defined as follows

$$\max_{\zeta} \ \mathcal{R}_\zeta \tag{1}$$

$$\text{s.t.} \sum_{f\in\mathcal{F}} c_f^{(t)} s_f \le \alpha, \qquad t = 1,2,...,T, \tag{2}$$

$$c_f^{(t)} \in \{0,1\}, \qquad f \in \mathcal{F}, t = 1,2,...,T. \tag{3}$$

Algorithm 1 The Proposed Edge Caching Scheme

procedure EDGECACHING
  for t = 1, 2, ..., T **do**
    Record users' information: age and gender
    Update user group profile if new members join in
    for each user group **do**
      Record request information:

$$R_f^{(t)} = \langle f, k_f, t\rangle, \forall f \in \mathcal{F}$$

      Record number of requests for all movies f up to time slot t and movie genres $k \in k_f \subset \mathcal{K}$ accordingly
      Generate time series for each movie genres
      for each movie genre $k \in \mathcal{K}$ **do**
        Forecast the one-day-ahead number of views $x_{h,k}^{(t+1)}$
      end for
    end for
    for each movie genre $k \in \mathcal{K}$ **do**
      Combine forecasts over user groups $\hat{y}_k^{(t)}$
    end for
    Obtain vector of number of views for each genre in the next day
  end for

IV. Forecasting Problem

A. Formulation for Forecasting Problem

Our task in this section is forecasting the future value of movie genre's popularity thus history data of each genre need to be collected for a forecasting model. To do this, for each user group, we first collect movies' title that have been watched by them in a day. Then genres are extracted and number of views for each genre is recorded, and sorted by time. Formally, let $x_{k,h}^{(t)}$ be the number of views for movie genre $k \in \mathcal{K}$ in user group $h \in \mathcal{H}$ at time t. For user group $h \in \mathcal{H}$, $X_{k,h} := [x_{k,h}^{(1)}, x_{k,h}^{(2)}, ..., x_{k,h}^{(T)}] \in \mathbb{R}^T$ denotes an observed history time series of $x_{k,h}^{(t)}$. Here T is the length of the time series, which is assumed to be the same for all time series.

Our main objective is to develop a hierarchical model, see Fig.3, which is built base on hierarchical architecture including lower layer and upper layer. Here, lower layer contains a

TABLE II
TABLE OF NOTATIONS

Notation	Description
$f, \mathcal{F}$	Movie and set of movies
$c_f^t \in \{0,1\}$	Cache decision variable at time slot t
y_f^{t+1}	Number of requests for movie f at time slot $t+1$
α, s_f	Cache capacity at base station and size of each movie f
t, M, T	Time slot, total number of time slots, and length of time series
$\mathcal{I}, \mathcal{J}, \mathcal{H}, \mathcal{K}$	Set of indices of gender, age features, union of two sets of indices of gender and age features, and movie's genres
$x_{k,h}^t, \hat{x}_{k,h}^{T+m}$	Number of views at time t (in the past) and forecasted number of views $T+m$ (in the future) for movie genre $k \in \mathcal{K}$ in user group $h \in \mathcal{H}$
$X_{k,h}, g, \lambda,$	Observed history time series of $x_{k,h}^{(t)}$, LSTM model, and regularization parameter
$w_{k,h}, y_{k,h}, \hat{y}_{k,h}$	Weights in linear combinations of forecasts in ensemble process, true and predicted values for movie's genre $k \in \mathcal{K}$ over user population
X_t, S_t, T_t, R_t	time series, seasonal component, trend-cycle component, and residual or remainder component, all at time t
$f,\ i,\ o$	forget gate, input gate and output gate in an LSTM units
$\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o, \mathbf{W}$	Weight matrices according to gates, $\mathbf{W} = [\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o]^\top$
$b_f, b_i, b_o, \mathbf{b}$	Bias vectors according to gates, $\mathbf{b} = [b_f, b_i, b_o]^\top$
σ, F	Logistic sigmoid function and Rectified Linear Unit (ReLU) activation function

global LSTM model g, which is responsible for one-day-ahead forecasting of number of views for each movie's genre $k \in \mathcal{K}$ in each user group $h \in \mathcal{H}$:

$$\hat{x}_{k,h}^{(T+1)} = g(X_{k,h}, \theta), \quad k \in \mathcal{K}, h \in \mathcal{H}, \tag{4}$$

where θ is model parameter. After training on individual LSTMs, forecast estimates $\hat{x}_{k,h}^{T+1}$ aggregated at upper layer, where for each movie's genre $k \in \mathcal{K}$, a linear combination is formed to forecast the average value of number of views for this genre across entire user population:

$$\hat{y}_k^{(T)} = \sum_{h \in \mathcal{H}} w_{k,h} \hat{x}_{k,h}^{(T+1)}, \quad h \in \mathcal{H}, \tag{5}$$

where the weights $w_{k,h} \in \mathbb{R}^+$. These weights can be estimated using regression models that are standard regression, Ridge regression, and Lasso, respectively as follows [15]:

$$w_{k,h} = \arg\min_{w_{k,h}} \sum_{m=1}^{M} \left(y_m - \sum_{h \in \mathcal{H}} w_{k,h} \hat{x}_{k,h}^{(T+m)} \right)^2, \quad h \in \mathcal{H}, \tag{6}$$

$$w_{k,h} = \arg\min_{w_{k,h}} \sum_{m=1}^{M} \left(y_m - \sum_{h \in \mathcal{H}} w_{k,h} \hat{x}_{k,h}^{(T+m)} \right)^2 + \lambda \sum_{h \in \mathcal{H}} \left(w_{k,h} \right)^2, \ h \in \mathcal{H}, \tag{7}$$

and

$$w_{k,h} = \arg\min_{w_{k,h}} \sum_{m=1}^{M} \left(y_m - \sum_{h \in \mathcal{H}} w_{k,h} \hat{x}_{k,h}^{(T+m)} \right)^2 + \lambda \sum_{h \in \mathcal{H}} |w_{k,h}|, \quad h \in \mathcal{H}, \tag{8}$$

where y_m is the true value at time m, M refers to the number of data points for training the regression model, and $\lambda > 0$ is the regularization parameter controlling the size of $w_{k,h}$ such that when values of $\lambda \to \infty$, the weights $w_{k,h}$ tend to zero [15].

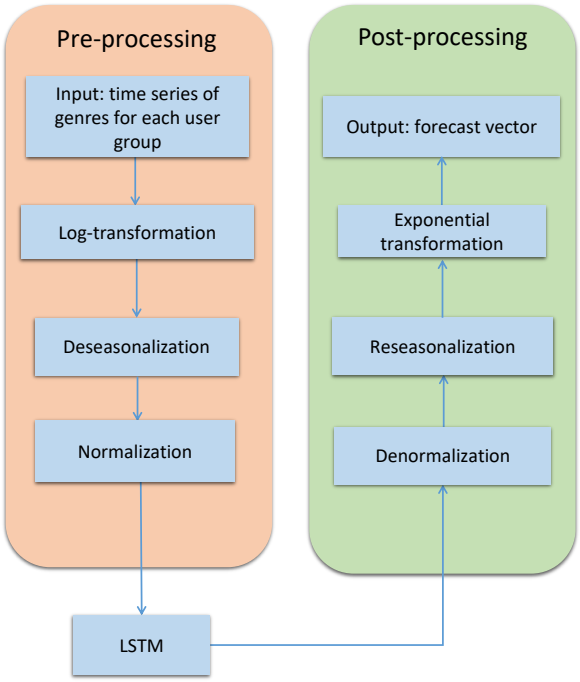

Fig. 4. Overview of pre-processing and post-processing of time series.

V. Detailed Process Using LSTM and Ensemble Model

The overall procedure is summarized as follows: we first stabilize the variance, using a log transformation. Then, the series is decomposed into trend, seasonal part, and remainder using STL decomposition method. Afterwards, the rolling window approach, along with a local normalization technique is applied to the sum of trend and reminder, to generate training data. Before going into the details, we first start with the definition of components of time series [18].

A. Data Pre-processing

Definition 1. *Time series components*

A time series is additive if

$$X_t = S_t + T_t + R_t.$$

A time series is multiplicative if

$$X_t = S_t \times T_t \times R_t,$$

where X_t is the time series, S_t is the seasonal component, T_t is the trend-cycle component, and R_t is the residual or remainder component, all at time t.

In this paper, the multiplicative time series is more appropriate since the variation of the seasonal pattern appears to vary with the level of the time series, see Fig... Interestingly, we can use multiplicative decomposition via employing additive decomposition, that is

$$y_t = S_t \times T_t \times R_t \Leftrightarrow \log(y_t) = \log(S_t) + \log(T_t) + \log(R_t).$$

This means that stabilizing the time series over time using log transformation, then using the additive decomposition. In the following, the commonly used additional decomposition, Seasonal and Trend decomposition using Loess's theory (STL) is applied based on this correspondence to obtain the multiplicative decomposition.

The process of data processing is performed as follows:

1) Variance stabilization: This step is used for stabilizing the variance in data, because the seasonality decomposition technique that we will use in the following is an additive method. Variance stabilization using power transformations: however, we stabilize the variance in our data for two reasons. As the STL decomposition method that we employ for seasonality extraction is an additive method, we need to ensure that seasonality is additive. Furthermore, we model the trend in a conservative way, and stabilizing the variance enables us to model greater dynamics in the trend. The log transform is a strongly non-linear transform and is usually used with caution, as small differences in log space may result in large differences in the original space, and therewith model fitting can yield sub-optimal results.

The time series is transformed in the natural logarithm scale as follows:

$$X_i^* = \begin{cases} \log(X_i), & \min(X) > 0 \\ \log(X_i + 1), & \min(X) = 0 \end{cases}$$

where X denotes a time series, X_i^* is the log transformed time series i.

2) Seasonal decomposition: Seasonal components in time series are repeating patterns, or cycles that repeat regularly over time that affected by the seasonal factors such as the day in the week, the time in the year. Seasonality is always of fixed and known frequency [18]. A time series can involve multiple seasonal patterns, for instance, call volume in call centres, daily hospital admissions, requests for cash at ATMs, electricity and water usage, and access to computer web sites (referred from [18]) are specific examples exhibiting more complicated seasonal patterns in practice. With the diversity and complex seasonal cycles, forecasting models such as LSTMs difficultly capture all of them. Time-series decomposition is a common strategy to handle this problem. Here, time series is decomposed into different components, including trend, seasonal, and residuals. They are modeled separately, hence the complexity of the model is reduced compared with modeling on the whole time series. In many previous studies [22], [23], they suggested that removing seasonality helps NN model to produce better forecasts. One of the versatile and robust time-series decomposition is Box-Cox, ARMA, Trend, Seasonal (BATS) and Trigonometric BATS (TBATS), as proposed by [19], which is able to decompose time series with multiple seasonal patterns.

In the next step, we refer to the concept of moving-window approach, therefore we describe its concept before going to the main step. To be specific, our forecast model is written in a general form as:

$$y_{t+1} = f(x_t, x_{t-1}, ..., x_{t-d+1})$$

where $t \in \{d, ..., N-1\}$, with N is the size of time series. The forecast problem of one step ahead can be described as a supervised learning problem, where the prediction is taken from a batch of d previous values. To learn the model, the time series should be converted to a supervised data set. This can be done by using the moving-window approach. In the Fig. 6, the time series is converted to overlapped batches of length $d + 1$. Here d refers to the size of input window and 1 indicates size of the output window. The training data is generate through sliding the window one step until the last window reach the last observation in the training set.

3) Time series normalization via moving-window approach: The features are extracted from the window of a constant size covering the recent part of the time series. Modeling trend-local normalization: Within the rolling window processing, the last value of the trend in each input window, provided by STL is used for local normalization. The trend component extracted by the STL procedure of that last value is subtracted from each data point in the corresponding input and output window. This process is applied to each input and output window combination separately.

Modelling the trend in this way has various advantages over extracting the trend using STL and modelling it separately. In contrast to prediction of a deterministic seasonality, predicting forward the extracted trend of a time series is not trivial, so if we extracted the trend and predicted it separately, we would effectively face another non-trivial prediction problem. Instead, we use the RNN directly to predict the trend.

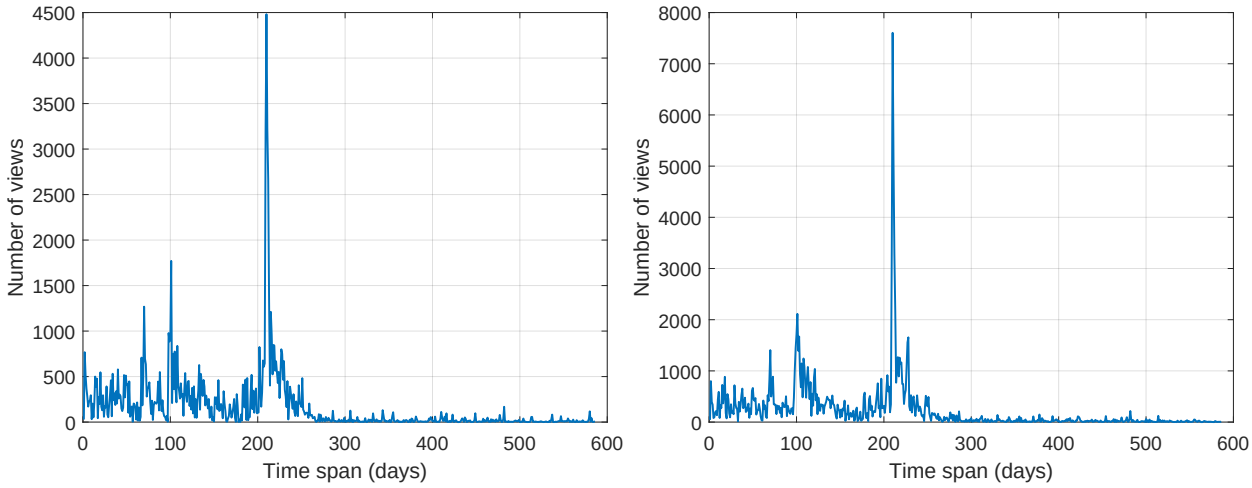

Fig. 5. Time series for drama genre subjected to two specific groups of users: (a) Female group (top), (b) Age group in range of [18, 24] (bottom).

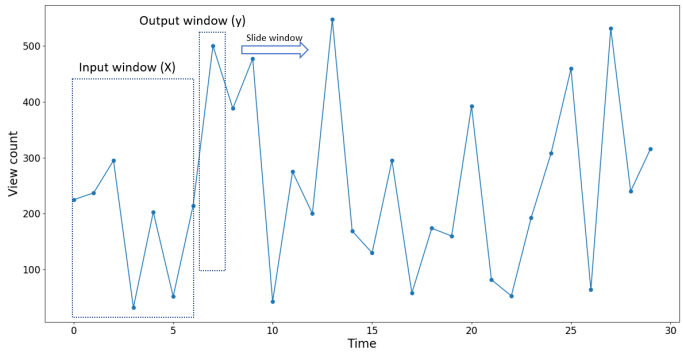

Fig. 6. Time series and sliding (moving) window.

B. Long Short-Term Memory Networks

In this paper, we utilize the Long Short-Term Memory Networks (LSTMs) to predict the genre-preference rate at each user's group. LSTM is a specific variant of Recurrent Neural Networks (RNNs) which are capable of learning long-term dependencies and overcomes the vanishing gradient problem. An LSTM block contains a memory cell and gates including forget gate, input gate, and output gate which manage the cell state and output. Specifically, forget gate conditionally decides to discard irrelevant information from the block. Input gate decides which input values will be used to update the memory state. And output gate decides what to output based on input and the memory of the block.

An LSTM network gives us a mapping from an input sequence $\{x^{(t)}\}$ to an output sequence $\{\hat{y}^{(t)}\}$ by forward passing an LSTM layer as presented in the following formulas:

- Forget gate:

$$f^{(t)} = \sigma\left(\mathbf{W}_f \begin{pmatrix} x^{(t)} \\ h^{(t-1)} \end{pmatrix} + b_f\right) \tag{9}$$

- Input gate:

$$i^{(t)} = \sigma\left(\mathbf{W}_i \begin{pmatrix} x^{(t)} \\ h^{(t-1)} \end{pmatrix} + b_i\right) \tag{10}$$

- Memory cell gate:

$$c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot tanh\left(\mathbf{W}_c \begin{pmatrix} x^{(t)} \\ h^{(t-1)} \end{pmatrix}\right) \tag{11}$$

- Memory output gate:

$$o^{(t)} = \sigma\left(\mathbf{W}_o \begin{pmatrix} x^{(t)} \\ h^{(t-1)} \end{pmatrix} + b_o\right) \tag{12}$$

- Output Computation:

$$h^{(t)} = o^{(t)} \odot tanh(c^{(t)}) \tag{13}$$

$$\hat{x}^{(t+1)} = F(h^{(t)}) \tag{14}$$

where the subscripts f, i, o denote for respectively forget gate, input gate and output gate. Following that notation, weight matrices are denoted as $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o$ and bias vectors b_f, b_i, b_o. We denote for later use $\mathbf{W} = [\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o]^\top, \mathbf{b} = [b_f, b_i, b_o]^\top$. The function $tanh$ represents hyperbolic tangent function, σ is logistic sigmoid function, and the $\odot$ function represents element-wise product. F is the rectified linear unit (ReLU) activation function which is used at the final output layer. The total loss function after M time steps is defined as follows:

$$Loss(x^{(t+1)}, \hat{x}^{(t+1)}; \mathbf{W}) \doteq \frac{1}{M} \sum_{t=1}^{M} \|x^{(t+1)} - \hat{x}^{(t+1)}\|_2^2, \tag{15}$$

where M is the total time step, $x^{(t+1)}, \hat{x}^{(t+1)}$ is the true output value and predicted output value at time $t+1$, respectively. Here $\|\cdot\|_2$ defines Euclidian norm in $\mathbb{R}$ and "$\doteq$" denotes for "is defined to be equal to".

For the problem of minimizing the loss function, we use Back Propagation Through Time (BPTT) to compute gradient of the loss function and the most basic version of Stochastic Gradient Descent (SGD) to update parameters.

C. Post-processing of Forecast Results

Once obtaining forecasts via testing the trained LSTM model on the test set, a post-processing process is carried

out to reverse the effects of previous pre-processing on these forecasts. The post-processing goes through the following steps. The reseasonalization step includes introducing the last seasonal component to the generated forecasts. Whereas in the renormalization step, the generated forecasts are back-transformed to their original scale, by adding the corresponding trend value obtained from the local normalization process, i.e, adding the last value of the trend inside an input window, and finally taking the exponent of the values.

D. Combining Forecasts

After individual LSTM forecasts on each user groups are trained, those forecasts are used as inputs for training a model which is responsible for combining all these forecasts. Our approach for this combination is averaging, regression, and ridge regression.

VI. Experimental Results

A. Data Set

In this paper, for evaluating our algorithm, we use the MovieLens 1M data set, which is available on [21]. This data set is one of the popular movie data sets used for movie recommendation publicly released by the research GroupLens [20]. The MovieLens 1M data set was collected from 2000 to 2003. It consists of 1 000 209 ratings of approximately 3900 movies made by 6040 users. Specifically, it provides anonymous user ID, movie ID, each user rates some movies by a number from 1 to 5, which is called rating at a timestamp. The data set also includes user information such as gender (2 groups), age (7 groups), occupation (21 groups), and zip-code. This additional information allows us to perform partition into specific group without any difficulty. In this paper, only two features gender and age are considered. Additionally, the movie data set also includes name of movies and their associating genres, which is indicated in the Table. I.

In our simulation, we use data collected within first two years from April 25, 2000 to December 1, 2001 (586 days), since almost ratings were provided within this time. We also assume that the caching entity updates its cache content in daily basis. Content refreshing can be performed in the off-peak hour in a day without affecting on the normal network activities. For simulating the movie requests, we assume that at each day (corresponding to timestamp of the data set), the rated movies are the requested from the users.

B. Accuracy Metrics for Time Series Forecasting Problem

In this paper, we need to use an available metric to evaluate the accuracy of the proposed forecasting method. We could use commonly used metrics, such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Symmetric Mean Absolute Percentage Error (sMAPE), or Mean Absolute Scaled Error (MASE) in [17]. However, two former metrics are scale-dependent measures, which is depend on the scale of the data; two latter ones pose infinite or undefined problems, since they involve division by a number equal or close to zero. For our data set, there are numerous zero observations, we thus not use these metrics. The final metric MASE is one of the best candidate which overcomes two above disadvantages. Two versions of this metric, which account for non-seasonality and seasonality of time series, are written as follows:

$$MASE = \frac{\frac{1}{h}\sum_{t=1}^{h}|F_t - Y_t|}{\frac{1}{n-1}\sum_{t=2}^{n}|Y_t - Y_{t-1}|} \tag{16}$$

$$MASE = \frac{\frac{1}{h}\sum_{t=1}^{h}|F_t - Y_t|}{\frac{1}{n-M}\sum_{t=M}^{n}|Y_t - Y_{t-M}|} \tag{17}$$

where Y_t represents actual observation at time t, F_t refers to the forecast at time t. The parameters h and n are the number of forecast values in the test set, number of observations in the training set, respectively. The seasonal period of a time series is represented as M. Mean Absolute Scale Error (MASE) measures the improvement possible from the proposed forecast method relative to the benchmark forecast method (e.g., the "naïve" method, where forecast value is equal to the last observation). Precisely, it is the mean absolute error between the forecast value and actual observation, divided by the mean absolute error between two consecutive observations in the training set. The MASE is a scale independent error measure, which also offers interpretability, as it measures the forecasting accuracy relative to the average one-step naïve forecast error, or to the naïve seasonal forecast error, if the time series is seasonal. When MASE < 1, the proposed method is better than the benchmark method, and when MASE > 1, the proposed method is worse than the benchmark method.

C. Hyper-parameter Setting

TABLE III
Setting of Hyperparameters

Hyperparameter	Value
Batch size	20
Number of epochs	200
Number of hidden layers	1
Number of neurons	100
Learning rate	0.0001
Dropout	0.1
L2-regularization parameter	5×10^{-4}

D. Forecasting Results

To obtain time series that serve our goal in this paper, we perform in the following way. For each user group, from the movies they have watched in daily basis, the number of views of according genres of those movies is recorded. These observations are sorted by day for each genre in each user group. Here we have 18 genres and 9 user groups, thus we have $18 \times 9 = 162$ times series in total.

In our forecasting experiment, we apply an LSTM model for one-day ahead forecasting, which uses history observations in 7 days, i.e., input window size is equal to 7 and output window

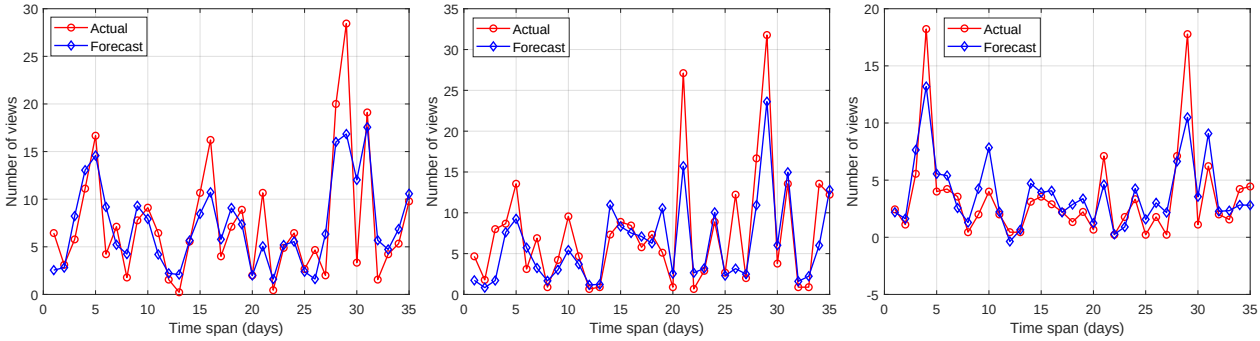

Fig. 7. Actual and forecast value within 35 days of three movie genres with highest view counts over user population: (a) Drama, (b) Comedy, and (c) Romance.

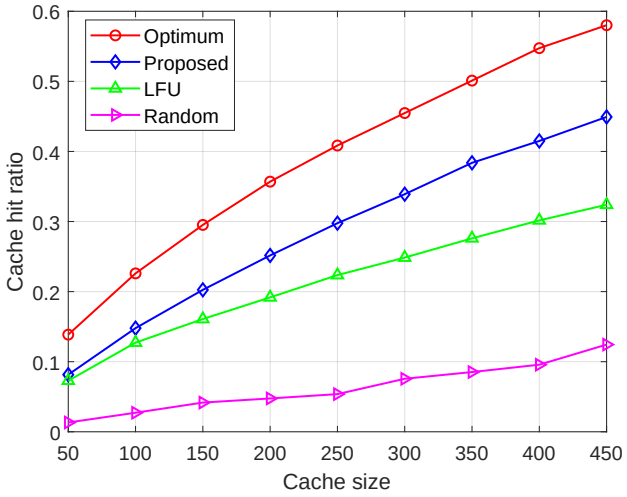

Fig. 8. Cache ratio vs cache size.

size is equal to 1. In fact, this is understandable in terms of weekly fluctuations seen in our time series.

The forecasting model is performed over time series of each movie's genre in each user group. For each time series, we use 70% of the data points as the training set for generating an LSTM model. Then for the rest of data set, we split 70% for training the ensemble model and 30%, i.e., 35 days for testing one day ahead.

E. Caching Performance Metric

Considering a base station with cache size c, which is in range of $\{50, 100, 150, 200, 250, 300, 350, 400, 450\}$. The measure for evaluating the performance of caching scheme is cache efficiency, [8] which describes the percentage of the requests that can be served locally at cache. The cache efficiency is defined as follows:

$$\text{Cache hit ratio} = \frac{\text{cache hits}}{\text{cache hits} + \text{cache misses}}$$

Once the predicted number of views for each movie's genre in the next day is obtained, a vector of percentage v of view count associated with each genre is extracted. To perform caching process at time $t+1$, a window of k previous days is presented, which is used to extract all movies viewed within these days. They are partitioned into a group of the same genre, where one movie can appear in more than one group, since a movie can belong to more than one genre. Then they are sorted in ascending order according to the view counts within k days. For each cache size, the number of movies (of each genre) need to be cached is calculated by multiplying cache size c with the genre percentage vector v.

In order to demonstrate the effectiveness of our proposed algorithm, we compare it with two benchmark algorithms that are Random Caching scheme and Least Frequency Used (LFU) scheme. Random Caching scheme caches randomly movies extracted from previous day. Reactive Caching scheme caches movies with highest view count within k previous days.

F. Caching Results

VII. Conclusions

Acknowledgment

This research was supported by the Chung-Ang University Young Scientist Scholarship (CAYSS) Program and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-...).

References

[1] Jingjing Yao, Tao Han, and Nirwan Ansari, On Mobile Edge Caching, IEEE Communications Survey & Tutorials, 2019.
[2] L. Breslau, Pei Cao, Li Fan, G. Phillips and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," IEEE INFOCOM '99.
[3] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I Tube, You Tube, everybody tubes: Analyzing the world's largest user generated content video system," in Proc. 7th ACM SIGCOMM Conf. Internet Meas., 2007, pp. 1–14.
[4] K. Wang, Z. Chen, and H. Liu, "Push-based wireless converged networks for massive multimedia content delivery," IEEE Trans. Wireless Commun., vol. 13, no. 5, pp. 2894–2905, May 2014.
[5] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," IEEE/ACM Trans. Netw., vol. 17, no. 5, pp. 1357–1370, Oct. 2009.
[6] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," Commun. ACM, vol. 53, no. 8, pp. 80–88, 2010.
[7] E. Bastug, M. Bennis, M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks", IEEE Commun. Mag., vol. 52, no. 8, pp. 82-89, Aug. 2014.
[8] S. Muller, O. Atan, M. van der Schaar, and A. Klein, "Context aware proactive content caching with service differentiation in wireless networks," IEEE Transactions on Wireless Communications, vol. 16, no. 2, pp. 1024-1036, 2017.

[9] B. N. Bharath, K. G. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogeneous small cell networks," IEEE Trans. Commun., vol. 64, no. 4, pp. 1674–1686, Apr. 2016.
[10] B. N. Bharath, K. G. Nagananda, D. Gündüz and H. V. Poor, "Caching With Time-Varying Popularity Profiles: A Learning-Theoretic Perspective," in IEEE Transactions on Communications, vol. 66, no. 9, pp. 3837-3847, Sept. 2018.
[11] K. N. Doan, T. Van Nguyen, T. Q. S. Quek and H. Shin, "Content-Aware Proactive Caching for Backhaul Offloading in Cellular Network," in IEEE Transactions on Wireless Communications, vol. 17, no. 5, pp. 3128-3140, May 2018.
[12] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang and X. Shen, "Content Popularity Prediction Towards Location-Aware Mobile Edge Caching," in IEEE Transactions on Multimedia, vol. 21, no. 4, pp. 915-929, April 2019.
[13] G. Gürsun, M. Crovella and I. Matta, "Describing and forecasting video access patterns," 2011 Proceedings IEEE INFOCOM, Shanghai, pp. 16-20, 2011.
[14] Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. J. (1990). STL: A seasonal-trend decomposition procedure based on loess. Journal of Official Statistics, 6(1), 3–33.
[15] James, G., Witten, D., Hastie, T., Tibshirani, R.: Unsupervised learning. In: An Introduction to Statistical Learning, pp. 203–264. Springer, New York (2013)
[16] Slawek Smyl, A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting.
[17] Another look at measures of forecast accuracy.
[18] Forecasting: Principles and Practice.
[19] Alysha M. De Livera, Rob J. Hyndman, Ralph D. Snyder (2011) Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing, Journal of the American Statistical Association, 106:496, 1513-1527.
[20] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.
[21] MovieLens 1M data set: https://grouplens.org/datasets/movielens/1m/.
[22] M. Nelson, T. Hill, W. Remus, and M. O'Connor, "Time series forecasting using neural networks: Should the data be deseasonalized first?" J. Forecasting, vol. 18, no. 5, pp. 359–367, Sep. 1999.
[23] G. P. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," Eur. J. Oper. Res., vol. 160, no. 2, pp. 501–514, Jan. 2005.